МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р. Е. АЛЕКСЕЕВА»

Кафедра «Вычислительные системы и технологии»

СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

Методические указания к лабораторным работам по курсу «Теоретические основы проектирования цифровых схем» для студентов высших учебных заведений по направлению 09.03.01 «Информатика и вычислительная техника» профиль «Вычислительные машины, комплексы, системы и сети» всех форм обучения

Нижний Новгород 2017

Составители: **В.В. Горбалетов, П.С. Кулясов**
УДК

Синтез цифровых автоматов: метод. указания к лаб. работам по курсу «Теоретические основы проектирования цифровых схем» для студентов высших учебных заведений по направлению 09.03.01 «Информатика и вычислительная техника» профиля «Вычислительные машины, комплексы, системы и сети» всех форм обучения / НГТУ им. Р.Е. Алексеева; сост.: В.В. Горбалетов, П.С. Кулясов – Нижний Новгород, 2017. – 40 с.

В методических указаниях приводятся понятие о цифровом автомате, типы автоматов, и способы задания их функционирования. Рассматриваются оба этапа абстрактного синтеза цифровых автоматов. Изложение сопровождается конкретными примерами синтеза автоматов. Материал предназначен для студентов высших учебных заведений по направлению 09.03.01 «Информатика и вычислительная техника» профиля «Вычислительные машины, комплексы, системы и сети».

Редактор

Подп. к печ. 25.03.2017. Формат . Печать .
Бумага . Усл. печ. л. 2. Тираж . Заказ.

Нижегородский государственный технический университет им. Р.Е. Алексеева.
Типография НГТУ, 603950, Нижний Новгород, ул. Минина, 24.

Содержание

1. ВВЕДЕНИЕ

1.1 Понятие о цифровом автомате

Среди цифровых устройств преобразования и обработки информации большое место занимают такие, у которых значения выходных сигналов в данный момент времени определяются не только значениями входных сигналов, но и внутренним состоянием устройства, зависящим от предыстории входных сигналов. Такие устройства называют цифровыми автоматами с памятью или просто автоматами, в отличие от автоматов без памяти, которые называют комбинационными автоматами, или комбинационными схемами.

Таким образом, автомат – это преобразователь дискретной информации, который под воздействием входных сигналов переходит из одного состояния в другое, сохраняет принятое состояние в период между входными сигналами и выдает выходные сигналы.

Абстрактный автомат – математическая модель дискретного преобразователя, которая задается тремя множествами:

- $Z = \{Z_1,\ldots,Z_f,\ldots,Z_F\}$ – множество входных сигналов (входной алфавит);
- $W = \{W_1,\ldots,W_g,\ldots,W_G\}$ – множество выходных сигналов (выходной алфавит);
- $A = \{a_1,\ldots,a_m,\ldots,a_M\}$ – множество внутренних состояний (алфавит состояний)

и двумя характеристическими функциями: δ – функция переходов, λ – функция выходов.

Алфавит – это конечное множество попарно различных символов, которые называются буквами данного алфавита.

Абстрактный автомат имеет один входной и один выходной каналы (рис. 1) и предназначен для реализации некоторого отображения слов входного алфавита и слова выходного алфавита.

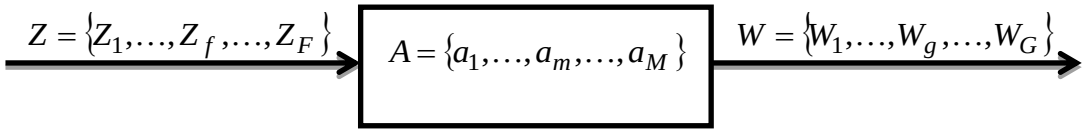


Рис.1. Абстрактный автомат

Автомат функционирует в дискретные моменты автоматного времени $t = 0,1,2,\ldots$.

В каждый момент времени t автомат находится в определенном состоянии $a(t) = a_m \in A$, воспринимает на входе некоторую букву $Z(t) = Z_f \in Z$, выдает на выход некоторую букву $W(t) = W_g \in W$, которая определяется функцией выходов λ как $W(t) = \lambda(a(t), Z(t))$, или, короче, $W_g = \lambda(a_m, Z_f)$, и переключается в новое состояние $a(t+1) = a_s \in A$, которое определяется функцией переходов δ как $a(t+1) = \delta(a(t), Z(t))$, или $a_s = \delta(a_m, Z_f)$.

Автомат называется конечным, если конечны множества *A*, *Z*, *W*. Автомат будет полностью определенным, если для любой пары $(a_m, Z_f) \in A \times Z$ определены функции δ и λ.

Автомат называется частичным, если функции δ и λ определены не для всех пар $(a_m, Z_f) \in A \times Z$.

Состояние автомата в момент времени $t = 0$ называется начальным и обозначается $a(0) = a_1$.

Автомат называется детерминированным, если любой паре $(a_m, Z_f) \in A \times Z$ однозначно соответствует пара (a_s, W_g). В противном случае автомат называется вероятностным.

Для детерминированных автоматов справедливо следующее:

1. Любому входному слову из L букв (L – длина слова) соответствует выходное слово той же длины.
2. Двум входным словам, в которых первые L_1 букв совпадают, соответствуют выходные слова с совпадающими L_1 первыми буквами, если автомат перед подачей входных слов находился в одном и том же состоянии.

1.2 Типы автоматов и способы их задания

1.2.1 Типы автоматов

В зависимости от способа определения выходных сигналов различают два типа автоматов: автоматы Мили и автоматы Мура.

Закон функционирования автомата Мили задается системой уравнений вида:

$$\begin{cases} a(t+1) = \delta(a(t), Z(t)), \\ W(t) = \lambda(a(t), Z(t)) \end{cases}$$

У автомата Мили выходной сигнал в момент времени *t* зависит как от внутреннего состояния, так и от входного сигнала в момент времени *t*.

Закон функционирования автомата Мура задается системой уравнений вида:

$$\begin{cases} a(t+1) = \delta(a(t), Z(t)), \\ W(t) = \lambda(a(t)) \end{cases}$$

У автомата Мура выходной сигнал в момент времени *t* зависит только от внутреннего состояния в момент *t*.

Конечный автомат считается заданным, если заданы начальное состояние, входной, выходной алфавиты, алфавит внутренних состояний, функции переходов и выходов. Обычно автомат задается либо с помощью таблиц переходов и выходов, либо в виде направленного графа.

1.2.2 Таблицы переходов и выходов

Пусть имеется автомат Мили, у которого $Z = \{Z_1, Z_2, Z_3\}$, $W = \{W_1, W_2, W_3\}$, $A = \{a_1, a_2, a_3, a_4\}$, $a(0) = a_1$, а функции δ и λ задаются в виде таблиц переходов (табл. 1) и выходов (табл. 2). Или в виде совмещенной таблицы переходов и выходов (табл. 3).

Таблица 1

a(t) / z(t)	a_1	a_2	a_3	a_4
Z_1	a_1	a_3	a_3	a_3
Z_2	a_4	a_1	a_4	a_2
Z_3	a_2	a_3	a_4	a_1

Таблица 2

a(t) / z(t)	a_1	a_2	a_3	a_4
Z_1	W_1	W_1	W_2	W_1
Z_2	W_2	W_2	W_3	W_1
Z_3	W_1	W_1	W_2	W_3

Таблица 3

$a(t)$ / $Z(t)$	a_1	a_2	a_3	a_4
Z_1	a_1 / W_1	a_3 / W_1	a_3 / W_2	a_3 / W_1
Z_2	a_4 / W_2	a_1 / W_2	a_4 / W_3	a_2 / W_1
Z_3	a_2 / W_1	a_3 / W_1	a_4 / W_2	a_1 / W_3

Строки таблиц соответствуют входным сигналам в момент времени *t*, столбцы – состояниям в момент *t*. На пересечении *i*-ой строки и *j*-ого столбца проставляются состояния для момента времени $t+1$ и выходные сигналы в момент *t*. Первый столбец отличается начальным состоянием автомата a_1.

Проследим последовательность работы автомата:

$t=0$	1	2	3	4	5	
$Z(t)=$	Z_1	Z_2	Z_2	Z_3	Z_1	
$a(t)=$	a_1	a_1	a_4	a_2	a_3	a_3
$W(t)=$	W_1	W_2	W_1	W_1	W_2	

Функции переходов и выходов автомата Мура задаются в виде таблицы переходов (табл.4).

Таблица 4

$Z(t)$	W_3	W_1	W_2	W_3
	a_1	a_2	a_3	a_4
Z_1	a_2	a_4	a_4	a_3
Z_2	a_3	a_2	a_3	a_3
Z_3	a_4	a_1	a_2	a_2

Строки таблицы отмечены входными сигналами для момента времени *t*, столбцы – состояниями и выходными сигналами для момента *t*. На пересечении *i*-ой строки и *j*-ого столбца проставляется соответствующее состояние для момента *t+1*.

Последовательность работы автомата:

$t=$	0	1	2	3	4	5
$Z(t)=$	Z_1	Z_2	Z_2	Z_3	Z_1	
$a(t)=$	a_1	a_2	a_2	a_2	a_1	a_2
$W(t)=$	W_3	W_1	W_1	W_1	W_3	W_1

Для частичных автоматов, у которых функции δ и λ определены не для всех пар (a_m, Z_f), на месте неопределенных состояний и выходных сигналов ставится прочерк.

1.2.3 Графы автоматов

Граф автомата Мили – ориентированный граф, вершины которого соответствуют состояниям, а дуги – переходам между ними. Две вершины, a_m и a_s, соединяются дугой (a_m, a_s), если в автомате имеется переход из a_m в a_s при некотором входном сигнале Z_f.

Дуге (a_m, a_s) приписывается входной сигнал Z_f и выходной сигнал $W_g = \lambda(a_m, Z_f)$, если он определен, и ставится прочерк в противном случае.

Если переход автомата из состояния a_m в состояние a_s происходит под действием нескольких входных сигналов, то дуге (a_m, a_s) приписывают все эти входные и соответствующие им выходные сигналы.

При задании автомата Мура в виде графа выходной сигнал $W_g = \lambda(a_m)$ записывается внутри или рядом с вершиной a_m.

На рис. 2 представлен граф автомата Мили, заданного табл. 3.

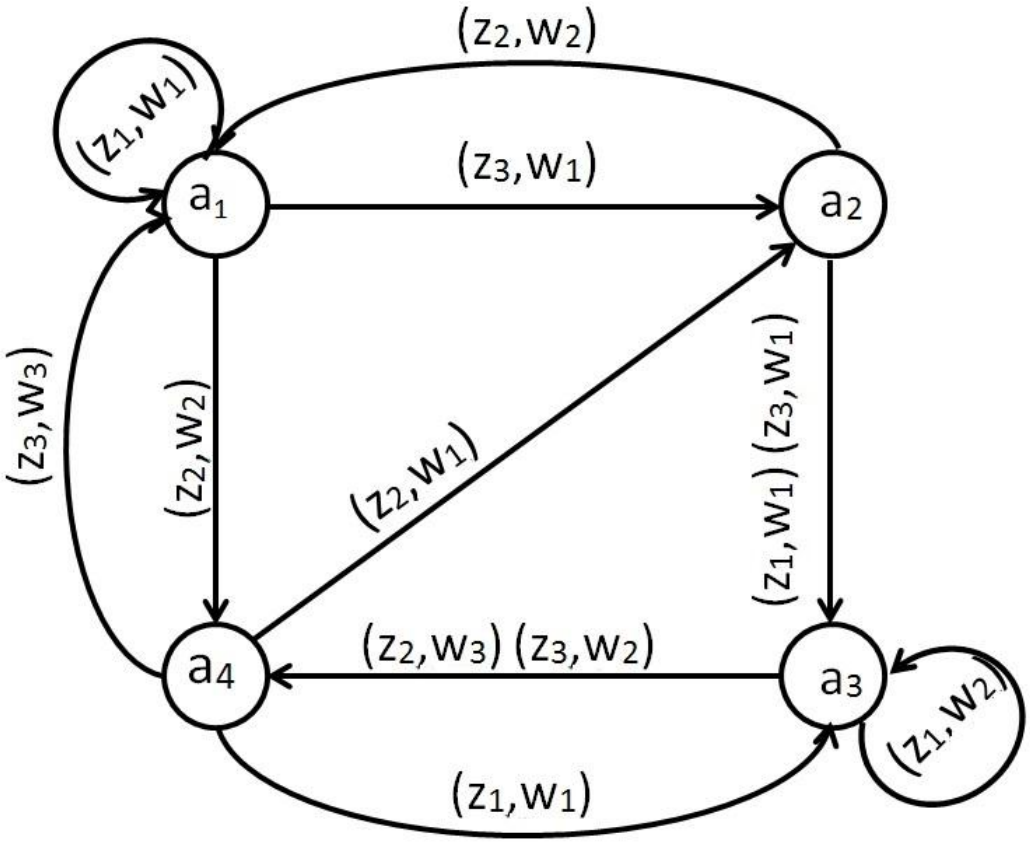


Рис. 2. Граф автомата Мили

На рис. 3 изображен граф автомата Мура, заданного табл. 4.

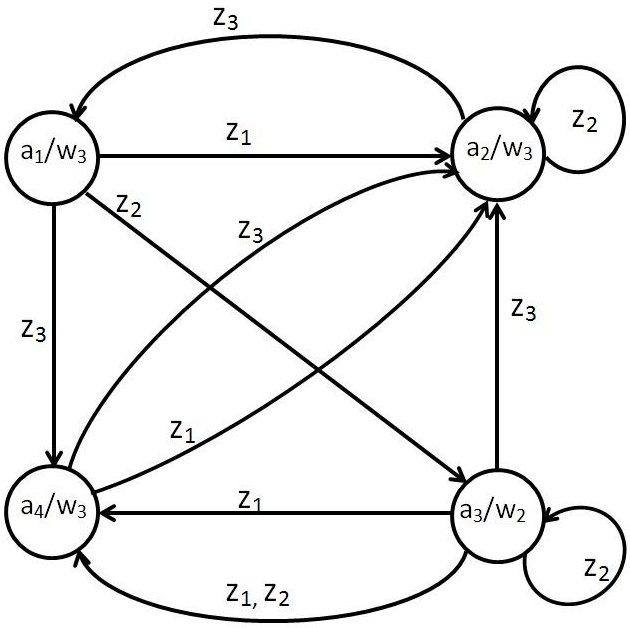


Рис. 3. Граф автомата Мура

Между автоматами обоих типов имеется связь, позволяющая преобразовывать любой автомат Мили в эквивалентный (ниже будет рассмотрено, что значит эквивалентный) ему автомат Мура, и наоборот.

1.3 Постановка задачи синтеза цифровых автоматов с памятью

Задача синтеза цифровых автоматов с памятью может быть сформулирована аналогично задаче синтеза автоматов без памяти, а именно: имеется алгоритм преобразования информации и некоторая элементная база, необходимо построить оптимальную (например, в смысле максимума быстродействия или минимума аппаратных затрат) схему, реализующую данный алгоритм.

Однако, в отличие от автоматов без памяти, эта задача более сложная, так как составить таблицу зависимости значений выходных сигналов от входных, как это делалось при синтезе комбинационных автоматов, невозможно вследствие бесконечности множества допустимых входных слов.

Поэтому необходимо выбрать такую форму задания автомата, которая позволяла бы представлять соответствие между бесконечными множествами входных и выходных слов конечными формулами или таблицами. Этому условию удовлетворяют рассмотренные выше стандартные способы задания автоматов.

Заметим, что задачи, связанные с получением таблиц переходов и выходов, а также их оптимизацией, решаются на этапе абстрактного синтеза автоматов. Задачи же, связанные с получением конкретной схемы по одной из форм задания автоматов (таблица, граф) решаются на этапе структурного синтеза автоматов.

2. АБСТРАКТНЫЙ СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

Абстрактный синтез автомата включает в себя два этапа:

1. Синтез автомата, реализующего заданный алгоритм.
2. Минимизация числа состояний.

Рассмотрим выполнение первого этапа применительно к автоматам, заданным в виде допустимых входных и выходных слов.

2.1 Автоматные отображения

Часто закон функционирования автомата задается в виде списка допустимых входных слов, для каждого из которых указывается выходное слово, которое должен сформировать автомат. То есть автомат задается в виде детерминированного отображения Φ:

$$\Phi : Z_{доп} \to W^*,$$

где $Z_{доп}$ – список допустимых входных слов, W^* – список выходных слов.

Отображение Φ представляется обычно с помощью таблицы соответствия или информативно нагруженного дерева.

Например, табл. 5 представляет собой таблицу соответствия некоторого автомата, который при подаче на вход последовательности $Z_0Z_2Z_1$ выдает на выход последовательность $W_1W_2W_1$ при подаче на вход $Z_1Z_2Z_0Z_1 - W_0W_1W_2W_1$ и т.д.

Таблица 5

$Z_{доп}$	W^*
$Z_0Z_2Z_1$	$W_1W_2W_1$
$Z_1Z_2Z_0Z_1$	$W_0W_1W_2W_1$
$Z_1Z_0Z_0Z_2Z_1Z_0$	$W_0W_2W_1W_0W_1W_2$

На рис. 4 этот же автомат задан с помощью информативно-нагруженного дерева.

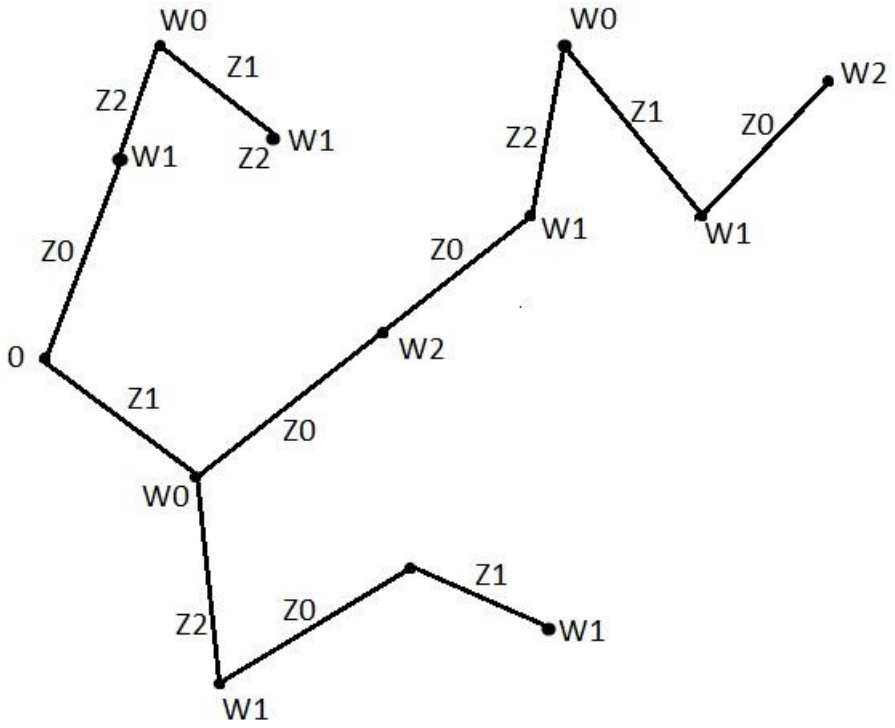


Рис. 4. Информативно-нагруженное дерево

Как следует из рис. 4, информативно-нагруженное дерево представляет собой граф, построенный по следующим правилам:

1. Фиксируется корень дерева (вершина первого ранга) – в виде точки 0.
2. Из него проводится $n_1 \le F$ ребер первого ранга, где n_1 – число различных первых букв из алфавита *Z*, которыми начинаются слова из множества $Z_{доп}$, $F = |Z|$.
3. Каждому ребру ставится в соответствие та буква входного алфавита, с которой начинаются слова из $Z_{доп}$. Ребра заканчиваются вершинами второго ранга.
4. Из любой вершины второго ранга проводится $n_2 \le F$ ребер второго ранга, где n_2 – число различных вторых букв из в словах из множества $Z_{доп}$. Описанный процесс продолжается до тех пор, пока не будет изображена последняя буква самого длинного слова из $Z_{доп}$. Любой путь в дереве, составленный из его ребер с общими вершинами, будет изображать определенное слово из $Z_{доп}$. Корень дерева является началом пути.
5. Приписав каждому ребру соответствующую букву из выходного алфавита, получим информативно-нагруженное дерево, в котором каждому пути соответствует свое входное и выходное слово.

Обратим внимание на то, что не всякое отображение является автоматным (реализуемым с помощью конечного автомата). Для того, чтобы это стало возможным, необходимо, чтобы отображение удовлетворяло определенным условиям.

2.2 Условия автоматности отображения

Условие 1. Автоматное отображение Φ является однозначным.

Условие 2. Автоматное отображение Φ сохраняет длину слова.

Условие 3. Автоматное отображение Φ однозначно переводит любой начальный отрезок входного слова в соответствующий той же длине начальный отрезок выходного слова.

Чтобы лучше уяснить смысл данных условий приведем примеры их невыполнения.

1) Пусть задано отображение Φ в виде табл.6

Таблица 6

$Z_{доп}$	W^*
$Z_1Z_0Z_1$	$W_1W_0W_1$
$Z_1Z_0Z_1$	$W_1W_0W_0$
Z_1Z_2	W_1W_2

Реакция автомата на одно и то ж входное слово $Z_1Z_0Z_1$ неоднозначна, поэтому здесь не выполняется условие 1.

2) Пусть имеется отображение Φ, заданное табл.7

Таблица 7

$Z_{доп}$	W^*
$Z_1Z_2Z_1$	$W_1W_2W_1$
Z_1Z_2	$W_1W_2W_2$
$Z_2Z_1Z_1$	W_1W_2

Здесь выходные и входные слова второй и третьей строчек таблицы имеют разную длину, поэтому не выполняется условие 2.

3) Пусть имеется отображение Φ, заданное табл.8

Таблица 8

$Z_{доп}$	W^*
$Z_1Z_2Z_1$	$W_2W_1W_1$
$Z_1Z_2Z_2$	$W_2W_2W_1$
$Z_1Z_1Z_2$	$W_2W_2W_2$

Здесь не выполняется условие 3, так как начальному отрезку входных слов Z_1Z_2 соответствуют разные начальные отрезки выходных слов W_2W_1 и W_2W_2.

Во всех рассмотренные примерах отображения не автоматные.

Приведение отображений к автоматному виду часто производится путем дополнения входных и выходных слов пустыми буквами.

Пустая буква (эта буква обозначается как C) в выходном алфавите означает, что в месте ее постановки выходной сигнал не считывается, то есть он может быть любым.

На месте пустой буквы входного слова также может находиться любая буква входного алфавита. Введение пустых букв во входные слова можно представить как предварительное запоминание входных слов.

Рассмотрим примеры приведения отображения к автоматному виду при нарушении условия 2 (табл. 9) и условия 3 (табл. 10)

Таблица 9

$Z_{доп}$	W^*		$Z_{доп}$	W^*
$Z_0Z_1Z_0$	CCW_1		Z_0Z_0C	$W_0W_2W_0$
$Z_0Z_1Z_1$	CCW_2		Z_0Z_1C	$W_0W_1W_2$
$Z_1Z_2Z_0$	CCW_2		Z_1Z_0C	$W_2W_1W_3$
			Z_1Z_1C	$W_2W_1W_1$

Таблица 10

$Z_{доп}$	W^*
Z_0Z_0C	CW_0W_1
Z_0Z_1C	CW_1W_0
Z_1Z_0C	CW_1W_1
Z_1Z_1C	CW_0W_0

1.1. Синтез автомата по таблице соответствия

1.1.1. Синтез автомата по информативно нагруженному дереву

Рассмотрим методику синтеза на конкретном примере. Пусть задано автоматное отображение в виде следующей таблицы соответствия (табл. 11).

Таблица 11

$Z_{доп}$	W^*
Z_0Z_0C	CW_1W_0
Z_0Z_1C	CW_0W_1
Z_1Z_0C	CW_0W_0
Z_1Z_1C	CW_1W_1

Для нашего примера информативно-нагруженное дерево выглядит так, как изображено на рис. 5.

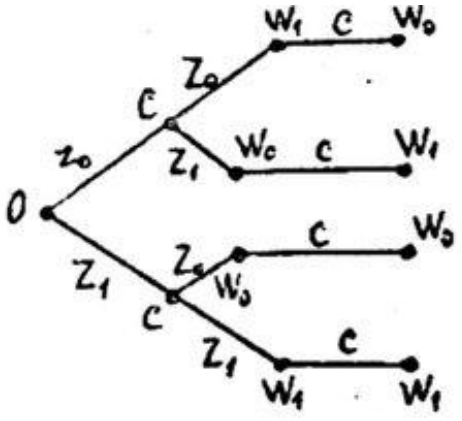


Рис. 5

Заменив вершины дерева состояниями автомата, а ребра – соответствующими дугами, можно от информативно-нагруженного дерева перейти к графу автомата.

Граф автомата Мура для данного дерева изображен на рис. 6.

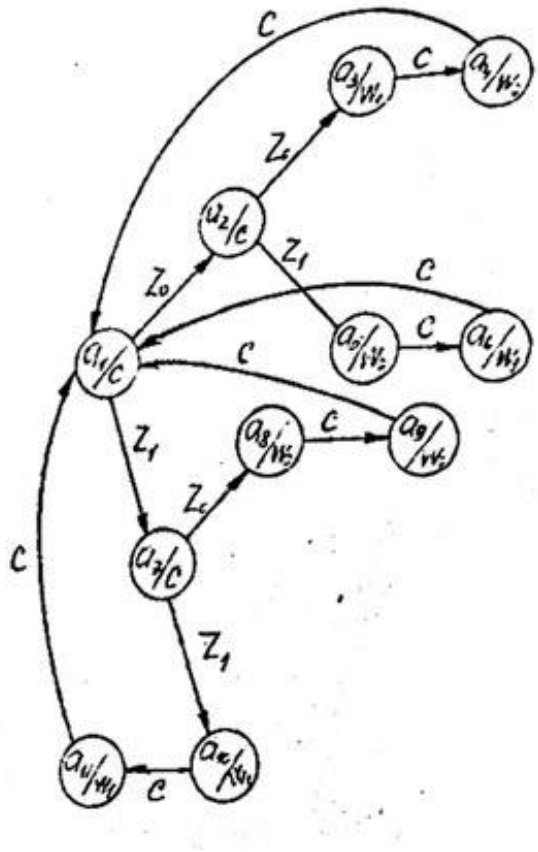


Рис. 6

Из анализа этого графа видно, что состояния a_4, a_6, a_9 и a_{11}, соответствующие в дереве вершинам четвертого ранга, сообщаются с начальным состоянием a_1. Это сделано для того, чтобы автомат каждый раз (в дополнительном четвертом такте) после выработки соответствующего выходного слова возвращался в начальное состояние и был тем самым готов для последующей работы.

Граф автомата Мили для данного примера приведен на рис. 7. Поскольку у автомата Мили выходной сигнал зависит от состояния и входного сигнала, то это делает излишними состояния, соответствующие в дереве вершинам максимального ранга.

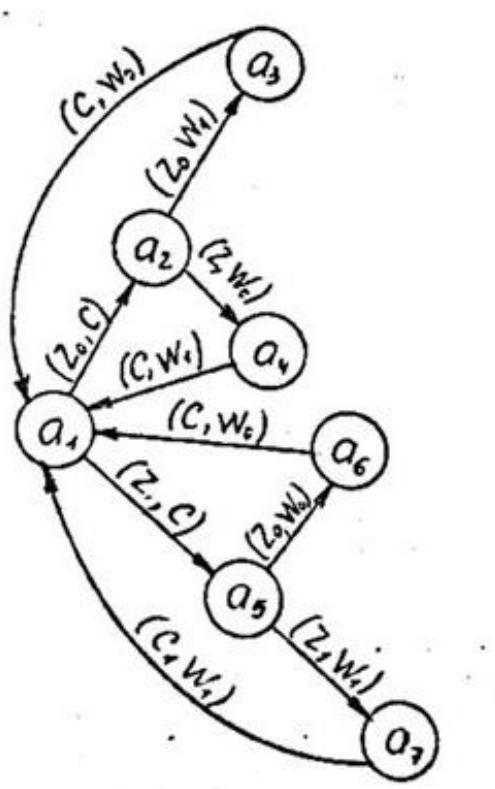


Рис. 7

От построенных графов легко можно перейти к соответствующим таблицам переходов и выходов. Отмеченная таблица переходов автомата Мура приведена в табл. 12, а совмещенная таблица переходов и выходов автомата Мили – в табл.13.

Таблица 12

$Z(t)$	C	C	W_1	W_0	W_0	W_1	C	W_0	W_0	W_1	W_1
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
Z_0	a_2	a_3	a_4	a_1	a_6	a_1	a_8	a_9	a_1	a_{11}	a_1
Z_1	a_7	a_5	a_4	a_1	a_6	a_1	a_{10}	a_9	a_1	a_{11}	a_1

Таблица 13

$a(t)$ / $z(t)$	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Z_0	a_2 / C	a_3 / W_1	a_1 / W_0	a_1 / W_1	a_6 / W_0	a_1 / W_0	a_1 / W_1
Z_1	a_5 / C	a_4 / W_0	a_1 / W_0	a_1 / W_1	a_7 / W_1	a_1 / W_0	a_1 / W_1

Символ C в этих таблицах можно изменить прочерком и рассматривать пустую букву как неопределенный сигнал.

Недостаток рассмотренного метода синтеза автоматов по информативно-нагруженному дереву состоит в том, что он приводит к автоматам со сравнительно большим числом состояний. Трудоемкость же второго этапа абстрактного синтеза (этапа минимизации числа состояний) во многом определяется числом состояний исходного автомата. Поэтому уже на первом этапе надо стремиться получить автомат с возможно меньшим числом состояний.

1.1.2. Синтез автоматов по разметке вход-выходных слов

Остановимся еще на одном методе, который по сравнению с уже рассмотренным позволяет строить автоматы со значительно меньшим числом состояний.

Пусть задано автоматное отображение, представленное в табл. 11. Требуется построить совмещенную таблицу переходов и выходов автомата Мили, реализующего заданное отображение.

Синтез автомата выполняется в такой последовательности:

1) Выписываются попарно друг под другом все входные и соответствующие им выходные слова. Буквы всех слов определяются вертикальными линиями, которым в дальнейшем (при разметке) присваиваются определенные номера состояний автомата.
2) Первая и последняя линии каждой пары вход-выходного слова отмечаются индексом 1 начального состояния a_1 автомата. Например, для первой пары будем иметь:

Z_0	Z_0	C
C	W_1	W_0
1		1

3) Выполняется разметка путем приписывания последовательно слева направо (внутри каждой пары вход-выходного слова) и сверху вниз (относительно различных пар) всем линиям индексов состояний. Основная идея метода состоит в том, что разметку надо продвигать по возможности без введения новых состояний, соблюдая, однако, чтобы она не была противоречивой.
4) В соответствии с полученной разметкой заполняется совмещенная таблица переходов и выходов автомата Мили.

Остановимся более подробно на выполнении пункта 3. Присвоение номера одной вертикальной линии будем называть шагом разметки. Для автомата Мили выполненный шаг разметки, например

Z_n
W_k
i $\quad$ j

означает, что автомат, находясь в момент времени t в состоянии a_i, под воздействием входной буквы Z_n переключается в состояние a_j и выдает на выход сигнал W_k.

На каждом шаге разметки возможен один из трех случаев:

1.	2.	3.	
Z_n	C	Z_n	(1)
W_k	W_k	C	
i	i	i	

В первом случае возможен один из результатов:

1.1. Если на более ранних шагах уже встречались разметки вида (1), то неотмеченной линии присваивается номер j.

1.2. Если же к выполнению данного шага ни одна из разметок вида (1) не встречалась, то мы вправе неотмеченной линии приписать любой номер. Однако, в соответствии с основной идеей метода, необходимо попытаться обойтись уже имеющимися состояниями, т.е. приписать неотмеченной линии один из имеющихся к моменту выполнения данного шага номеров состояний автомата.

Заметим, что для некоторых номеров состояний дальнейшая разметка может оказаться противоречивой. В этом случае надо вернуться к данному шагу и попытаться использовать другой номер.

Перебор состояний удобно выполнять в порядке возрастания номеров, всегда начиная с номера 1.

Поскольку в этой ситуации метод предусматривает возвращение к данному шагу, то размеченная линия должна быть помечена особым образом, например, знаком «+», указывающим на то, что здесь мы сэкономили одно состояние автомата.

1.3. Возникает противоречие, если на более ранних шагах встречались разметки вида

$$\begin{array}{c|c|c|c|c|c} & Z_n & & & C & \\ & W_l & & \text{или} & W_l & \text{, где } W_l \neq W_k \\ i & j & & i & j & \end{array} \qquad (2)$$

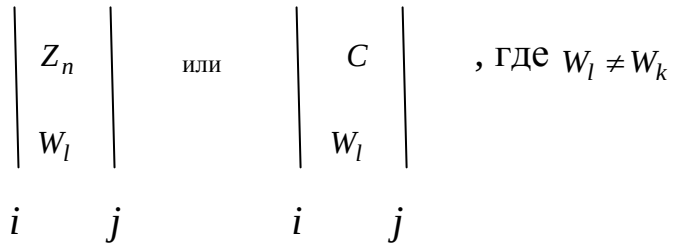
В случае возникновения противоречия необходимо в обратном порядке (т.е. справа налево и снизу вверх) вернуться до ближайшего шага, помеченного знаком + и, если этот шаг не последний (т.е. если выше есть еще знаки +), то изменить номер соответствующей линии на +1 и продолжить разметку дальше с учетом этого номера в том случае, если вновь полученный номер по-прежнему не является новым для данного шага разметки. Если же вновь полученный номер оказывается новым для данного шага разметки, то надо,

убрав знак + для данного шага, без введения нового состояния продолжить движение в обратном направлении до следующего ближайшего шага со знаком +. И, теперь уже для этого шага, попытаться, не вводя новых состояний, получить непротиворечивую разметку.

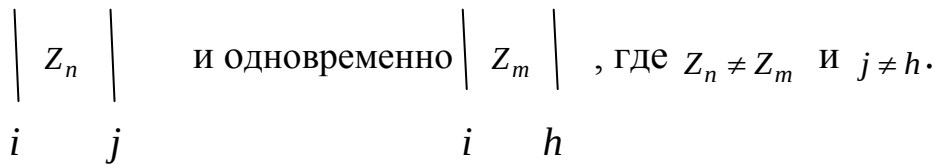
Вводить новое состояние можно только при условии, что для всех имеющихся к моменту выполнения данного шага номеров состояний дальнейшая разметка оказывается противоречивой и выше нет ни одного шага, помеченного знаком +. При введении нового состояния знак + для соответствующего шага исключается.

Для второго и третьего случая ситуации (2.1 и 3.1) и (2.2, 3.2) те же, что и 1.1 и 1.2 соответственно. Противоречий в третьем случае не возникает, а во втором случае имеем:

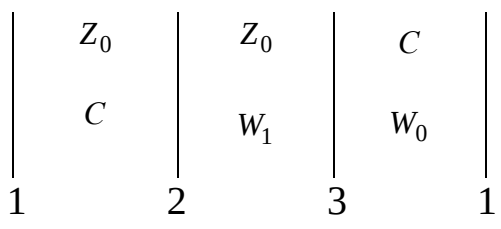
1.2. Возникает противоречие, если на более ранних шагах встречались разметки вида

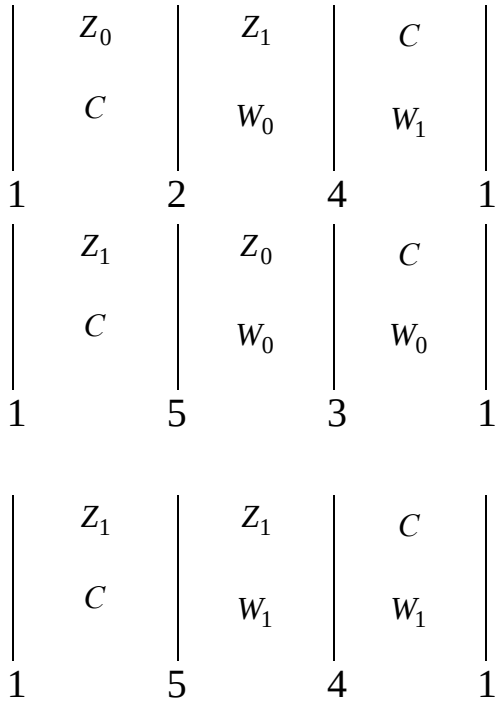


или



В результате применения описанного алгоритма для нашего примера разметка в окончательной форме имеет вид:



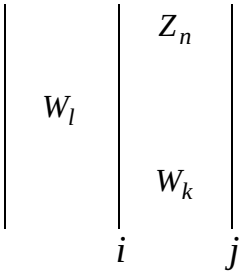


Совмещенная таблица переходов и выходов для данной разметки:

Таблица 14

$a(t)$ / $z(t)$	a_1	a_2	a_3	a_4	a_5
Z_0	a_2 / C	a_3 / W_1	a_1 / W_0	a_1 / W_1	a_3 / W_0
Z_1	a_5 / C	a_4 / W_0	a_1 / W_0	a_1 / W_1	a_4 / W_1

Описанный метод синтеза может быть использован при построении автомата Мура. Однако при этом появляются отличия, которые обусловлены в основном тем, что для автомата Мура выполненный шаг разметки, например



означает, что автомат, находясь в момент времени t в состоянии a_i, выдает на выход сигнал W_l и под воздействием входного сигнала Z_n переключается в состояние a_j.

Отличия возникают только при фиксации противоречия и при переборе состояний, помеченных знаком +, в остальном процедура синтеза остается без изменений.

С учетом сказанного получим разметку для автомата Мура:

Z_0	Z_0	C	C	
C	W_1	W_0	C	
1	2	3	4	1

Z_0	Z_1	C	C	
C	W_0	W_1	C	
1	2	5	6	1

Z_1	Z_0	C	C	
C	W_0	W_0	C	
1	7	8	9	1

Z_1	Z_1	C	C	
C	W_1	W_1	C	
1	7	10	6	1

По данной разметке построим таблицу переходов автомата Мура (табл.15)

Таблица 15

$Z(t)$	C	C	W_1	W_0	W_0	W_1	C	W_0	W_0	W_1
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
Z_0	a_2	a_3	a_4	a_1	a_6	a_1	a_8	a_9	a_1	a_6

Z_1	a_7	a_5	a_4	a_1	a_6	a_1	a_{10}	a_9	a_1	a_6

2.4. Минимизация автоматов

2.4.1. Постановка задачи

Сначала сформулируем задачу минимизации полностью определенных автоматов. Состояния a_i автомата M и a_l' автомата M' будем называть эквивалентными, если оба автомата, получив одну и ту же (любую) входную последовательность в состояниях a_i и a_l' соответственно, перерабатывают ее в одинаковую выходную последовательность. Автоматы M и M' будем называть эквивалентными, если для каждого состояния автомата M существует эквивалентное состояние автомата M', и наоборот. Автомат, эквивалентный заданному и имеющий наименьшее возможное число состояний, называется минимальным. Задача построения минимального автомата и есть задача минимизации автомата.

В случае частичных автоматов задача минимизации формулируется иначе. Пусть на вход частичного автомата, находящегося в состоянии a_m, подается последовательность $Z_{i_1} \ldots Z_{i_p}$. Если на некотором шаге окажется, что следующее состояние не определено, то дальнейшее поведение автомата непредсказуемо. Это можно допустить лишь в том случае, когда неопределенное состояние возникает на последнем шаге (после подачи Z_{i_p}), ибо дальнейшее поведение автомата нас не интересует. Входную последовательность $Z_{i_1} Z_{i_2} \ldots Z_{i_p}$ будем называть допустимой к состоянию a_m автомата M, если определены состояния:

$$a_{m_1} = \delta(Z_{i_1}, a_m), a_{m_2} = \delta(Z_{i_2}, a_{m_1}), a_{m_{p-1}} = \delta(Z_{i_2}, a_{m_{p-2}}).$$

Пусть $\tilde{V}$ и $\tilde{W}$ – последовательности одинаковой длины, составленные из символов некоторого алфавита и неопределенного символа (пустой буквы). Будем говорить, что последовательность $\tilde{V}$ покрывает $\tilde{W}$, если некоторые неопределенные символы последовательности $\tilde{W}$ можно заменить так, что получится $\tilde{V}$. Например, последовательность 2-102-10 покрывает 2- -02- -0.

Состояние a_l' частичного автомата M' покрывает состояние a_i частичного автомата M, если любая входная последовательность, допустимая к состоянию a_i автомата M, допустима к состоянию a_l' автомата M' и

соответствующая выходная последовательность автомата M' покрывает выходную последовательность автомата M. Если для каждого состояния a_i автомата M найдется покрывающее его состояние a_i' автомата M', будем говорить, что автомат M' покрывает M. Автомат, покрывающий M и имеющий наименьшее число состояний, называется минимальным. Задача нахождения такого автомата и представляет собой задачу минимизации частичного автомата.

2.4.2. Максимальные классы совместимости.

Последовательности $\tilde{V}$ и $\tilde{W}$, составленные из символов некоторого алфавита и неопределенного символа, назовем совместимыми, если существует общая для них покрывающая последовательность. Это означает, что если в некоторой позиции последовательностей $\tilde{V}$ и $\tilde{W}$ расположены значащие символы, то эти символы должны совпадать. Например, последовательности 2 - 01- -012 и - 10- -10- - являются совместимыми.

Два состояния am и as автомата M называются совместимыми (обозначение $a_m \sim a_s$), если при подаче любой допустимой в a_m и a_s входной последовательности получаемые выходные последовательности совместимы. Состояния a_m и a_s называются несовместимыми, если они не являются совместимыми.

Приведем более конструктивное определение совместимости. Два состояния a_m и a_s одного и того же автомата M совместимы, если:

1. Для каждого входа $Z_f \in Z$, для которого оба выхода определены, $\lambda(a_m, Z_f) = \lambda(a_s, Z_f)$. Это условие называется совместимостью по выходу.
2. Для каждого входа $Z_f \in Z$, для которого оба выхода определены, $\delta(a_m, Z_f) \sim \delta(a_s, Z_f)$.

Данное определение относится к автоматам Мили. В случае автомата Мура условие 1 принимает другой вид: если оба выхода ($\lambda(a_m)$ и $\lambda(a_s)$) определены, то $\lambda(a_m) = \lambda(a_s)$.

Заметим, что отношение совместимости состояний не транзитивно: два состояния, совместимые с третьим, могут оказаться несовместимыми друг с другом. Это связано с тем, что две последовательности, совместимые с третьей,

могут быть несовместимыми между собой. Например, последовательности 001 и 011, несовместимые между собой, являются совместимыми с последовательностью 0-1.

Некоторое множество состояний частичного автомата образует класс совместимости, если все входящие в него состояния попарно совместимы. Класс совместимости называется максимальным, если при добавлении к нему любого состояния он перестает быть совместимым. Совокупность некоторых классов совместимости представляет собой покрытие (группировку), если всякое состояние автомата входит хотя бы в один из них. Покрытие, составленное из всех максимальных классов совместимости, называется максимальным.

Построение минимального автомата начинается с нахождения максимального покрытия, то есть всех максимально совместимых классов (МС-классов).

Множество МС-классов формируется на основе пар совместимых состояний с помощью треугольной таблицы, называемой диаграммой пар. Строки и столбцы диаграммы пар сопоставляются с состояниями автомата. Таблица 17 – треугольная таблица для автомата, заданного таблицей 16. Для упрощения записи в табл. 17 вместо a_i записано i.

Таблица 16

a(t) / z(t)	a_1	a_2	a_3	a_4	a_5
Z_1	a_2 / W_1	a_3 / W_1	a_3 / W_1	- / -	- / -
Z_2	- / W_2	a_5 / W_2	a_4 / W_2	a_1 / W_2	- / -
Z_3	a_1 / -	a_2 / W_1	- / -	a_2 / -	a_1 / W_2
Z_4	a_2 / W_1	- / -	a_5 / W_1	- / -	- / -

Таблица 17

2	2 3			
3	2 3 2 5	4 5		
4	2 3	1 5	1 4	
5	1 3	X	V	1 2
	1	2	3	4

На пересечении *m*-й строки и *s*-го столбца в исходной треугольной таблице ставятся:

1. X (крест), если состояния a_m и a_s несовместимы по выходу, например, a_5 и a_2 (в табл. 16 в строке Z_3 в столбцах a_2 и a_5 стоят разные выходные сигналы).

2. Множество всех пар состояний, следующих за парой (a_m, a_s) и отличных от нее. То есть все те пары, совместимость, которых необходима для совместимости пары (a_m, a_s). Например, для совместимости (a_3, a_1) необходима совместимость (a_2, a_3) и (a_2, a_5).

3. V (птичка), если за (a_m, a_s) не следует пары отличной от (a_m, a_s). То есть пара (a_m, a_s) совместима без дополнительных условий. Например, пара (a_5, a_3).

Далее вычеркиваются все клетки, содержащие вычеркнутые пары, и так далее, пока это возможно. В итоге получаем результирующую треугольную таблицу, которая в нашем примере представлена в табл. 18.

Таблица 18

2	2 3		
3	X	4 5	
4	2 3	X	1 4

5	X	X	V	1 2
	1	2	3	4

Невычеркнутые клетки результирующей таблицы соответствуют всем парам совместимых состояний. Действительно, если состояния a_m и a_s несовместимы, то найдется допустимая в a_m и a_s входная последовательность $\xi = Z_{i_1} Z_{i_2} ... Z_{i_p}$, на который автомат, запущенный в состояниях a_m и a_s, выдаст несовместимые выходные последовательности. Можно считать, что значащие значения выходов различаются лишь на шаге P (после подачи Z_{i_p}), иначе последовательность ξ можно укоротить (до первого различия значащих выходов). Пусть автомат при подаче ξ, начав функционировать из состояния a_m, проходит последовательность состояний $a_{m_1} a_{m_2} ... a_{m_p}$, а из a_s – последовательность $a_{s_1} a_{s_2} ... a_{s_p}$. В состояниях $a_{m_{p-1}}$ и $a_{s_{p-1}}$ подача Z_{i_p} приводит к разным значащим значениям выхода, поэтому пара $(a_{m_{p-1}}, a_{s_{p-1}})$ будет вычеркнута в треугольной таблице на первом шаге (при построении исходной треугольной таблицы). Поскольку сигнал $Z_{i_{p-1}}$ превращает пару $(a_{m_{p-2}}, a_{s_{p-2}})$ в $(a_{m_{p-1}}, a_{s_{p-1}})$, то на следующем шаге $(a_{m_{p-2}}, a_{s_{p-2}})$ также окажется вычеркнутой и так далее. Не позже, чем через p шагов исходная пара (a_m, a_s) будет вычеркнута.

Из табл. 18 заключаем, что всеми парами совместимых состояний для нашего примера будут (a_1, a_2), (a_1, a_4), (a_2, a_3), (a_3, a_4), (a_3, a_5) и (a_4, a_5).

Рассмотрение этих пар показывает, что отношение совместимости не транзитивно.

Результирующая треугольная таблица позволяет построить все МС-классы. Для этого таблица просматривается слева направо, и образуются некоторые системы множеств. Первоначально берется система из одного множества, в которое входят все состояния. Допустим, что после рассмотрения i-1 столбцов построена система множеств $A_1, A_2, ..., A_k$. При переходе к столбцу i выделяются все состояния, несовместимые с a_i (им соответствуют зачеркнутые клетки столбца). Если множество A_j одновременно не содержит a_i и несовместимых с ним состояний, оно не изменяется. В противном случае из него образуются два множества: одно – путем удаления состояния a_i, другое –

путем удаления всех состояний, несовместимых с a_i. Проделав это для всех случаев A_j и устранив множества, целиком содержащиеся в других, получим систему (результат) шага *i*. После просмотра последнего столбца будем иметь максимальное покрытие. Это следует из того, что каждое из полученных множеств является совместимым классом состояний и что никакой класс совместимости (в частности МС-класс) в течение процедуры не дробится, входя в некоторые множества целиком.

Для рассматриваемого примера результаты каждого шага отражены в табл. 19.

Таблица 19

№ шага	Система множеств
0	$\{a_1, a_2, a_3, a_4, a_5\}$
1	$\{a_2, a_3, a_4, a_5\}, \{a_1, a_2, a_4\}$
2	$\{a_3, a_4, a_5\}, \{a_2, a_3\}, \{a_1, a_4\}, \{a_1, a_2\}$
3	$\{a_3, a_4, a_5\}, \{a_2, a_3\}, \{a_1, a_4\}, \{a_1, a_2\}$
4	$\{a_3, a_4, a_5\}, \{a_2, a_3\}, \{a_1, a_4\}, \{a_1, a_2\}$

Последняя строка таблицы содержит все МС-классы. Заметим, что МС-классы для частичных автоматов могут пересекаться, а число их (в отличие от нашего примера) может значительно превышать число состояний исходного автомата.

2.4.3. Минимальное замкнутое покрытие

Пусть C_i – множество совместимых состояний, а C_{ij} – множество состояний, в которые может перейти автомат под действием входного сигнала Z_j, будучи установленным в одном из состояний множества C_i, то есть $C_{ij} = \{ a_k \mid a_k = \delta(a_m, Z_j) \}$ для всех $a_m \in C_i \}$. Будем говорить, что C_{ij} – множество состояний, следующих за множеством C_i по входу Z_j, или совместимых множеств $C = \{ C_1, C_2, ..., C_l \}$ является замкнутым, если для каждого $C_i \in C$ все предопределенные множества C_{ij} содержатся в некотором элементе из *C*.

Имея замкнутое покрытие $C=\{C_1,C_2,...,C_l\}$ (то есть замкнутое множество таких совместимых множеств $C_1,C_2,...,C_l$, которые образуют покрытие) состояний частичного автомата M, можно построить покрывающий его автомат M'. С этой целью каждому множеству $C_i(i=1,...,l)$ необходимо сопоставить в автомате M' состояние a'_i. Состояние a'_k, в которое переходит a'_i под действием сигнала Z_j, определяется множеством C_k, содержащим все состояния, в которые входной сигнал Z_j переводит состояния из C_i (если таких множеств C_k несколько, то можно взять любое из них). Поскольку C_i образует совместимое множество, то подача входного сигнала Z_j приводит в каждом состоянии из C_i либо к неопределенному символу на выходе, либо к символу, общему для всего множества C_i. Этот символ и принимается в качестве выходного сигнала, соответствующего переходу в автомате M' из состояния a'_i под действием Z_j. Если же в каждом состоянии из C_i при подаче Z_j возникает только неопределенный символ, то значение выхода автомата M' может быть назначено произвольно или может считаться неопределенным.

Таким образом, построение покрывающего автомата сводится к нахождению замкнутого покрытия, а построение минимального автомата – к построению минимального замкнутого покрытия, содержащего минимально возможное число множеств совместимых состояний.

Заметим, что множество всех МС-классов замкнуто, но не обязательно является минимальным для произвольного автомата. Замкнутое покрытие, состоящее только из МС-классов, может содержать больше множеств, чем замкнутое покрытие, в котором некоторые (или все) совместимые множества являются подмножествами МС-классов.

Сказанное означает, что для получения минимального замкнутого покрытия частичного автомата может оказаться необходимым анализ всех совместимых множеств, а не только МС-классов.

Однако задача построения минимального автомата значительно упрощается при использовании концепции доминирующих совместимых множеств, позволяющей исключить некоторые совместимые множества из рассмотрения.

Пусть по-прежнему C_i – множество (класс) совместимых состояний, а C_{ij} – множество (класс) состояний, следующих за классом C_i по входу Z_j. Назовем порожденным множеством P_i класса C_i множество всех классов C_{ij}, таких, что

1. $C_{ij} \not\subset C_i$.
2. $C_{ij} \not\subset C_{ik}$, если $C_{ik} \in P_i$.
3. C_{ij} содержит более одного элемента.

Будем говорить, что множество C_i доминирует над совместимым множеством C_j, если $C_i \supset C_j$ и $P_i \subseteq P_j$, где P_i и P_j представляют собой порожденные множества классов C_i и C_j соответственно. Если C_i доминируют над C_j, то C_i покрывает все состояния, покрытые C_j, и условия того, что покрытия, содержащие C_i или C_j, замкнуты, одинаковы.

Таким образом, при определении минимального замкнутого покрытия нет необходимости рассматривать множество C_j.

Совместимое множество состояний, не имеющее доминирующего над собой совместимого множества, назовем простым классом совместимых состояний (ПС-классом). Следующая ниже теорема демонстрирует полезность понятия ПС-класса.

Теорема 1. Для любого автомата M существует покрывающий M минимальный автомат M' все состояния которого соответствуют ПС-классам автомата M.

Доказательство. Предположим, что существует минимальный автомат M'', показывающий M, и что некоторое состояние M'' соответствует совместимому множеству C_j, не являющемуся ПС-классом. Тогда существует ПС-класс C_j', который доминирует над C_j. Замена каждого элемента C_j в M'' на C_j' не приведет к увеличению числа состояний. Повторение такой замены для всех состояний, соответствующих ПС-классам, позволит получить результирующий автомат с числом состояний, не большим, чем в автомате M''.

ПС-классы получаются разложением каждого МС- класса на подклассы и последующей проверкой того, какие из этих классов простые.

Начинается процесс с включения МС-классов, наибольших по числу элементов, в список простых классов. Пусть они содержат по *n* элементов. В нашем примере это $C_1 = \overline{3.4.5}$ из табл. 20. Запись $\overline{3.4.5}$ следует понимать как $\{a_3, a_4, a_5\}$.

Таблица 20

Максимальные классы	Порожденные множества
$\overline{3.4.5}$	$\overline{1.4}, \overline{1.2}$
$\overline{2.3}$	$\overline{4.5}$
$\overline{1.4}$	$\overline{2.3}$
$\overline{1.2}$	$\overline{2.3}$

На первом шаге генерируются все подклассы порядка $n-1$ и вычисляются порожденные ими множества. К списку ПС-классов добавляются только те из них, которые не исключаются каким-либо ПС-классом, уже вошедшим в список. У нас это

$C_2 = \overline{3.4}, \quad P_2 = \overline{1.4};$

$C_3 = \overline{3.5}, \quad P_3 = \varnothing;$

$C_4 = \overline{4.5}, \quad P_4 = \overline{1.2}.$

Здесь $\varnothing$ – пустое множество. Затем к списку добавляются все МС-классы порядка $n-1$:

$C_5 = \overline{2.3}, \quad P_5 = \overline{4.5};$

$C_6 = \overline{1.4}, \quad P_6 = \overline{2.3};$

$C_7 = \overline{1.2}, \quad P_7 = \overline{2.3}.$

На следующем шаге генерируются все подклассы порядка $n-2$. Генерирующими классами при этом являются МС-классы порядка n и $n-1$. Процесс продолжается до тех пор, пока не будут сгенерированы подклассы всех порядков. На каждом шаге в список добавляются только подклассы, не исключаемые уже вошедшими в список, а также МС-классы того же порядка.

Список всех ПС-классов для рассматриваемого примера приведен в табл. 21.

Таблица 21

Простые классы	Порожденные множества
$C_1 = \overline{3.4.5}$	$\overline{1.4}, \overline{1.2}$
$C_2 = \overline{3.4}$	$\overline{1.4}$
$C_3 = \overline{3.5}$	$\varnothing$
$C_4 = \overline{4.5}$	$\overline{1.2}$
$C_5 = \overline{2.3}$	$\overline{4.5}$
$C_6 = \overline{1.2}$	$\overline{2.3}$
$C_7 = \overline{1.4}$	$\overline{2.3}$
$C_8 = \bar{4}$	$\varnothing$
$C_9 = \bar{2}$	$\varnothing$
$C_{10} = \bar{1}$	$\varnothing$

После того, как найдены все ПС-классы, строится специальная таблица покрытия и замкнутости, состоящая из двух частей (для нашего примера это табл. 22). Строки таблицы соответствуют ПС-классам, а столбцы – либо состояниям исходного автомата (для левой части таблицы), либо всем различным порожденным множествам, содержащим хотя бы два состояния и соответствующим всем ПС-классам (для правой части таблицы). На пересечении *i*-й строки и *j*-го столбца левой части таблицы ставится знак X (крест), если $a_j \in C_i$. Эта часть аналогична импликантной таблице, используемой при минимизации булевых функций. Совокупность простых классов, имеющая крест в каждом столбце левой части таблицы, является покрытием.

Таблица 22

	1	2	3	4	5	$\overline{1.4}$	$\overline{1.2}$	$\overline{4.5}$	$\overline{2.3}$
$C_1 = \overline{3.4.5}$			X	X	X	0	0	X	
$C_2 = \overline{3.4}$			X	X		0			
$C_3 = \overline{3.5}$			X		X				
$C_4 = \overline{4.5}$				X	X		0	X	
$C_5 = \overline{2.3}$		X	X					0	X
$C_6 = \overline{1.2}$	X	X					X		0
$C_7 = \overline{1.4}$	X			X		X			0

$C_8 = \bar{4}$				X					
$C_9 = \bar{2}$		X							
$C_{10} = \bar{1}$	X								

Для замкнутости покрытия необходимо, чтобы эта совокупность строк удовлетворила условиям, выраженным в правой части таблицы.

Обозначим столбцы правой части таблицы через C_j'. На пересечении *i*-й строки и *j*-го столбца этой части таблицы проставляется либо 0 (кружок), либо X (крест). Крест ставится в случае, если $C_j' \subseteq C_i$, кружок – когда $C_j' \in P_i$. Очевидно, что если какая-либо строка из совокупности строк, образующих замкнутое покрытие, имеет знак 0 в каком-либо столбце, то этой совокупности должна принадлежать хотя бы одна строка, имеющая в этом столбце знак X. В этом случае условия замкнутости покрытия будут выполнены.

Известен ряд методов построения минимального замкнутого покрытия на основе рассмотренной таблицы. Однако, они отличаются большой трудоёмкостью. Рассмотрим менее трудоёмкий способ, базирующийся на построении деревьев, вершинами которых приписаны простые классы.

Правила построения таких деревьев:

1. Выбирается состояние автомата, которое содержится в минимальном числе ПС-классов. Если есть несколько таких состояний, выбирается любое из них. Все ПС-классы, содержащие это состояние, выбираются в качестве начальных и помещаются слева в корнях деревьев.

В табл. 21 каждое из состояний a_1, a_2, a_5 содержится только в трех ПС-классах. Выбираем a_1. Тогда начальными элементами будут ПС-классы $\overline{1.2}$, $\overline{1.4}$, $\bar{1}$ (рис. 8).

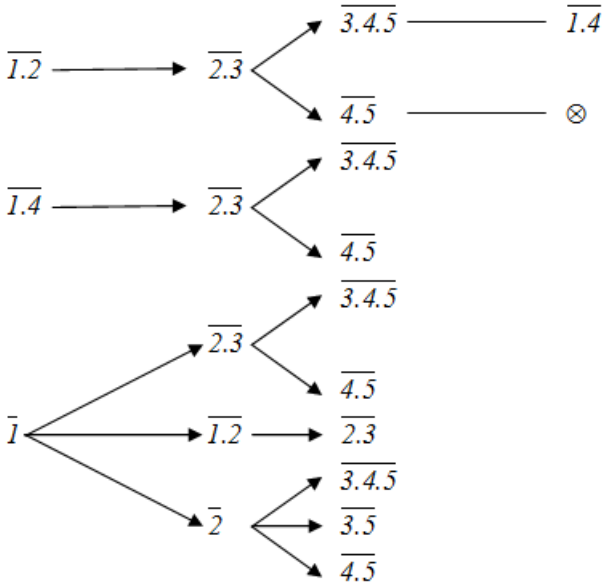


Рис. 8

2. Дерево строится из каждой вершины следующим образом:

А) Пусть в качестве вершины в дерево перенесен ПС-класс C_i, такой, что его порожденное множество P_i не пусто и не покрыто множеством $D(C_i)$ простых классов, лежащих на пути, проходящем через корень дерева и вершину, соответствующую ПС-классу C_i (то есть C_i не замкнут в этом множестве). Тогда из C_i строятся различные пути так, чтобы с помощью всевозможных комбинаций ПС-классов удовлетворить условие замыкания C_i.

Например, рассмотрим путь $(\overline{1.2}, \overline{2.3})$ на рис. 8. Когда $C_5 = \overline{2.3}$ вносится в дерево, то $D(C_5) = \{\overline{1.2}, \overline{2.3}\}$, а $P_5 = \overline{4.5}$. Условие замыкания для C_5 не выполнено. Множества $\overline{3.4.5}$ и $\overline{4.5}$ будут удовлетворять его условиям замыкания. Поэтому они использованы для формирования двух путей, выходящих из $\overline{2.3}$.

Б) Допустим, что в качестве вершины в дерево внесен ПС-класс C_i, порожденное множество P_i которого пусто или покрывается множеством $D(C_i)$ простых классов, лежащих на пути из начального элемента в C_i. Если каждое состояние содержится в некотором классе $D(C_i)$, то $D(C_i)$ – замкнутое покрытие, причем минимальное, если соответствующий ему путь самый короткий. Если существуют состояния, которые не содержатся в простых классах $D(C_i)$, то выбираем состояние, которое содержится в минимальном числе простых классов. Каждый ПС-класс, содержащий это состояние, определяет некоторую ветвь, выходящую из C_i.

Рассмотрим начальную вершину $C_{10} = \bar{1}$ на рис. 8. Порожденное множество $P_{10} = \varnothing$. Ни одно из состояний a_2, a_3, a_4, a_5 не содержится в классе $\bar{1}$. Состояния a_2 и a_5 содержатся только в трех ПС-классах из множества всех ПС-классов, тогда как a_3 – в четырех, а a_4 – в пяти. Выбираем состояние a_2 и три ветви $\overline{2.3}$, $\overline{1.2}$, $\bar{2}$ (все ПС-классы, содержащие a_2) заносим вслед за $\bar{1}$ в дерево на рис. 8.

В) Все пути должны строится параллельно до получения одного минимального замкнутого покрытия. Это позволит избежать построения путей избыточной длины.

Так как для рассматриваемого примера путь ($\overline{1.2}, \overline{2.3}, \overline{4.5}$) определяет минимальное замкнутое покрытие, то пути длиной больше трёх не строятся.

Обоснование этих правил очевидно. Ни один ПС-класс не включается в некоторый путь без удовлетворения условий покрытия и замыкания. Хотя бы один начальный элемент должен входить в минимальное замкнутое покрытие, чтобы удовлетворялись условия покрытия.

Таким образом, минимальным замкнутым покрытием для автомата, заданного табл. 16, будет $A' = \{a_1', a_2', a_3'\}$, где a_1' соответствует ПС-классу $\overline{1.2}$, $a_2' - \overline{2.3}$, $a_3' - \overline{4.5}$. В соответствии с этим минимальным покрытием в табл. 23 представлен минимальный автомат M'.

Таблица 23

$Z(t)$ / $a'(t)$	a_1'	a_2'	a_3'
Z_1	a_2' / W_1	a_2' / W_1	– / –
Z_2	a_3' / W_2	a_3' / W_2	a_1' / W_2
Z_3	a_2' / W_1	a_1' / W_1	a_1' / W_2

Z_4	a_1' / W_1	a_3' / W_1	– / –

2.4.4 Минимизация полностью определенных автоматов.

Рассмотренная процедура минимизации частичных автоматов со значительными упрощениями может быть использована и при минимизации полностью определенных автоматов. Докажем сначала две леммы.

Лемма 1. Если для полностью определенного автомата $a_i \sim a_j$ и $a_j \sim a_m$, то $a_i \sim a_m$.

Доказательство. Предположим, что $a_i \sim a_j$, $a_j \sim a_m$, а $a_i \sim a_m$ не выполняется. Тогда должна существовать входная последовательность ξ, которая, будучи приложенной к автомату в состояниях a_i и a_m, формирует определенные выходные последовательности φ_i и φ_m соответственно, причем $\varphi_i \neq \varphi_m$. Если ξ, приложенная к автомату в состоянии a_j, формирует выходную последовательность φ_j, то по предположению $\varphi_i = \varphi_j$ и $\varphi_j = \varphi_m$. Так как все выходные последовательности полностью определены, $\varphi_i = \varphi_m$, что противоречит предположению о том, что $a_i \sim a_m$ не выполняется.

Лемма 2. МС-классы полностью определенного автомата не пересекаются (то есть для каждого состояния a_i имеется только один МС-класс, содержащий a_i).

Доказательство. Предположим, что состояние a_i содержится в двух МС-классах, C_1 и C_2, $C_1 \neq C_2$. Так как C_1 и C_2 представляют собой МС-классы, $C_1 \not\subset C_2$ и $C_2 \not\subset C_1$. Следовательно, каждое из множеств C_1 и C_2 должно содержать некоторое состояние, не содержащееся в другом множестве. Пусть $a_j \in C_1$, $a_j \notin C_2$ и $a_k \in C_2$, но $a_k \notin C_1$. Так как $a_i, a_j \in C_1$, то $a_i \sim a_j$, а так как $a_i, a_k \in C_2$, то $a_i \sim a_k$. Учитывая, что по предположению a_j и a_k находятся в разных МС-классах, $a_j \sim a_k$ не выполняется, что противоречит лемме 1.

В основе процедуры минимизации полностью определенного автомата лежит следующая теорема.

Теорема 2. Множество всех МС-классов полностью определенного автомата является минимальным замкнутым покрытием.

Доказательство. Вытекает непосредственно из леммы 2.

Таким образом, для минимизации полностью определенного автомата достаточно построить множество всех МС-классов.

В качестве примера в табл. 25 показана результирующая треугольная таблица для автомата, заданного табл. 24. Множество МС-классов $\{\overline{1.2.3}, \overline{4.5}, \overline{6}\}$ – замкнутое и минимальное, так как автомат полностью определен. Упрощенный автомат, представленный табл. 26, эквивалентен исходному (состояние a_1' соответствует МС-классу $\overline{1.2.3}$, $a_2' - \overline{4.5}$, $a_3' - \overline{6}$).

Таблица 24

$Z(t)$ / $a(t)$	a_1	a_2	a_3	a_4	a_5	a_6
Z_1	a_3 / W_0	a_2 / W_0	a_2 / W_0	a_1 / W_1	a_2 / W_1	a_1 / W_1
Z_2	a_4 / W_1	a_4 / W_1	a_5 / W_1	a_6 / W_0	a_6 / W_0	a_5 / W_1
Z_3	a_3 / W_0	a_2 / W_0	a_1 / W_0	a_6 / W_0	a_6 / W_0	a_4 / W_1

Таблица 25

2	2 3				
3	2 3 4 5 1 3	4 5 1 2			
4	X	X	X		
5	X	X	X	1 2	
6	X	X	X	X	X
	1	2	3	4	5

Таблица 26

$a'(t)$ / $Z(t)$	a_1'	a_2'	a_3'
Z_1	a_1' / W_0	a_1' / W_1	a_1' / W_1
Z_2	a_2' / W_1	a_3' / W_0	a_2' / W_1
Z_3	a_1' / W_0	a_3' / W_0	a_2' / W_1

Список литературы

1. Прикладная теория цифровых автоматов / А.Я. Савельев – М.: Высшая школа, 1987.

2. Синтез микропрограммных автоматов (граф – схемы и автоматы) / С.И. Баранов – Л.: Энергия, 1979.

3. Теория и проектирование переключательных схем / А. Фридман, П. Менон – М.: Мир, 1978.

4. Основы общей теории конечных автоматов / М.К. Чирков – Л.: Издательство Ленинградского университета, 1975.

5. Основы теории дискретных логических и вычислительных устройств / Л.А. Шоломов – М.: Наука, 1980.

6. Синтез цифровых автоматов / В.М. Глушков – М.: Физматгиз, 1962.

7. Асинхронные исследовательские схемы / С. Ангер – М.: Наука, 1977.

8. Проектирование импульсных и цифровых устройств радиотехнических систем / Под редакцией Ю.М. Казаринова – М.: Высшая школа, 1985.

9. Структура электронных вычислительных машин / С.А. Майоров, Г.И. Новиков – Л.: Машиностроение, 1979.