

УЧЕБНОЕ ПОСОБИЕ

Н. Г. Захаров, В. Н. Рогов

**СИНТЕЗ
ЦИФРОВЫХ АВТОМАТОВ**

Ульяновск 2003

Министерство образования Российской Федерации
Государственное образовательное учреждение высшего профессионального образования
Ульяновский государственный технический университет

Н. Г. Захаров, В. Н. Рогов

СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

Учебное пособие

Ульяновск 2003

УДК 519. 713 (075)
ББК 32. 972 я 7
3-38

Утверждено редакционно-издательским советом университета в качестве учебного пособия.

Рецензенты: кафедра математической кибернетики и информации Ульяновского государственного университета, заведующий кафедрой доктор технических наук, профессор И. В. Семушин;
директор Ульяновского филиала института радиотехники и электроники Академии Наук Российской Федерации А. А. Широков.

Захаров Н. Г. , Рогов В. Н.

3-38 Синтез цифровых автоматов: Учебное пособие / Н. Г. Захаров, В. Н. Рогов. – Ульяновск: УлГТУ, 2003.

ISBN 5-89146-300-0

Изложены основные понятия формальных грамматик, приведены синтез абстрактного и структурного конечных цифровых автоматов. Рассмотрены работы машин Тьюринга и сетей Петри. Предназначена для студентов специальности 200700 «Радиотехника», 220100 «Вычислительные машины, комплексы и системы связи», 200900 «Сети связи и системы коммутации».

УДК 519.713(075)
ББК 32.927 я 7

© Н. Г. Захаров, В. Н. Рогов, 2003

ISBN 5-89146-300-0

© Оформление. УлГТУ, 2003

ОГЛАВЛЕНИЕ

Введение	5
1. Основы теории формальных грамматик	6
1.1. Основные понятия теории автоматов	6
1.2. Основные понятия теории формальных грамматик	8
1.3. Классификация языков по Хомскому.....	11
1.4. Распознающие устройства и автоматы.....	13
1.4.1. Концепция порождения и распознавания	14
1.4.2. Распознающие, порождающие и преобразующие формальные грамматики..	16
1.5. Автоматы и формальные языки	18
1.5.1. Понятие об информации и ее преобразованиях	18
1.5.2. Преобразование алфавитной информации	20
1.5.3. Способы задания автоматов	21
Контрольные вопросы	25
2. Машины Тьюринга	25
2.1. Основные понятия	25
2.2. Машины Тьюринга с двумя выходами	27
2.3. Машины Тьюринга и линейно-ограниченные автоматы	29
2.4. Автоматы с магазинной памятью и бесконтекстные языки	30
2.4.1. Автоматы с магазинной памятью	30
2.4.2. Бесконтекстные (контекстно-свободные) языки	32
Контрольные вопросы	33
3. Абстрактный конечный автомат	33
3.1. Абстрактная теория автоматов	33
3.1.1. Модель дискретного преобразователя Глушкова В.М	33
3.1.2. Понятие об абстрактном автомате и индуцируемом им отображении	34
3.2. Представление событий в автоматах.....	36
3.2.1. Автоматные отображения и события	36
3.2.2. Представление событий в автоматах	38
3.2.3. Регулярные языки и конечные автоматы	39
3.3. Алгоритм синтеза конечных автоматов	40
3.3.1. Основной алгоритм синтеза конечных автоматов	40
3.3.2. Усовершенствованный основной алгоритм синтеза конечных автоматов..	45
3.4. Автоматы Мили и Мура	49
3.4.1. Автомат Мили	49
3.4.2. Автомат Мура	52
3.4.3. Получение неполностью определенных (частичных) автоматов	53
3.5. Синтез автоматов по индуцируемым ими отображениям	53
3.5.1. Общий метод решения задачи	53
3.5.2. Синтез автомата Мили	55
3.5.3. Синтез автомата Мура.....	57
Контрольные вопросы.....	58
4. Структурный конечный автомат	58
4.1. Основные понятия структурной теории автоматов	58
4.2. Композиция автоматов и структурные схемы	61
4.3. Условия корректности и правильности построения схем	63
4.4. Канонический метод структурного синтеза автомата	67

4.4.1. Основная задача теории структурного синтеза автоматов	67
4.4.2. Теорема о структурной полноте	69
4.4.3. Комбинационная часть автомата. Синтез схемы автомата	71
4.5. Кодирование состояний. Гонки в автомате	73
4.6. Построение комбинационной схемы автомата	75
Контрольные вопросы.....	79
5. Микропрограммирование	79
5.1. Принципы микропрограммного управления	80
5.2. Система команд автоматов, реализующих выполнение алгоритма	81
5.3. Набор операций автомата	82
5.4. Состав и назначение элементов блок-схемы	83
5.5. Общий алгоритм функционирования	84
5.6. Основные характеристики автоматов	84
5.7. Устройство управления микропрограммным автоматом	85
5.8. Формирование адреса микрокоманд	87
Контрольные вопросы.....	91
6. Проблемы отображения времени при проектировании	91
6.1. Модель тактируемого дискретного автомата	91
6.2. Выбор параметров тактирующих сигналов	93
6.3. Сравнение способов тактирования автоматов	94
6.4. Абсолютная и относительная шкала времени	95
6.5. Характеристики сигналов в абсолютной шкале времени	96
6.6. Характеристики сигналов в относительной шкале времени	99
Контрольные вопросы.....	101
7. Сети Петри	101
7.1. Назначение и общая характеристика сетей Петри	101
7.2. Структура и способы представления сетей Петри	103
7.2.1. Структура сетей Петри	104
7.2.2. Графы сетей Петри	105
7.2.3. Маркировка сетей Петри	106
7.2.4. Работа сетей Петри.....	107
7.3. Анализ сетей Петри	108
7.3.1. Безопасность сети Петри.....	108
7.3.2. Анализ живучести (сохранения) сетей Петри	109
7.3.3. Активность сети Петри	111
7.3.4. Достижимость и покрываемость	112
7.4. Моделирование алгоритмов с помощью сетей Петри	114
7.5. Расширенные сети Петри	118
7.6. Сети Петри и регулярные языки	120
7.7. Преобразование конечного автомата в сеть Петри	121
7.8. Разработка модели сложения двух чисел с плавающей запятой	124
Контрольные вопросы.....	126
Заключение.....	126
Приложение.....	127
Предметный указатель.....	133
Библиографический список.....	134

Введение

Учебное пособие «Синтез цифровых автоматов» предназначено для более углубленного изучения студентами специальности 200700 «Радиотехника» дисциплины «Цифровые устройства и микропроцессоры».

В результате изучения дисциплины студент должен изучить: цифровые автоматы, представляемые как математические модели дискретных систем; связь автоматов с формальными языками и грамматиками; способы задания и принципы построения цифровых автоматов, а также методы и средства их разработки.

Создание современных автоматизированных средств и систем управления основывается на следующих двух научных направлениях: теории преобразования информации и теории построения различного рода преобразователей информации.

Первое направление включает в себя общую теорию алгоритмов, абстрактную теорию автоматов. В нем рассматриваются общие вопросы формализации процессов обработки информации.

Второе направление включает в себя элементы математической логики, структурную теорию автоматов. В нем рассматриваются принципы и методы построения автоматов, которые реализуют алгоритмы обработки информации, полученные на этапе абстрактного синтеза автоматов.

В соответствии с этими направлениями в теории автоматов выделяют два больших раздела: *абстрактную* теорию автоматов и *структурную* теорию автоматов.

В первом разделе пособия даны основные понятия теории формальных грамматик, классификация языков, понятие об информации и её преобразованиях.

Второй раздел посвящён изучению машин Тьюринга как абстрактных устройств, представляющих наиболее близко математическую модель вычислительных машин.

В третьем разделе дано понятие об абстрактном аппарате и индуцируемом им отображении. Рассмотрено представление событий в автоматах, приведены модели автоматов Мили и Мура.

В четвертом разделе даны основные понятия структурной теории автоматов, рассмотрены вопросы композиции автоматов и построения структурных схем. Приведен канонический метод структурного синтеза автоматов, рассмотрены задачи кодирования состояний и способы устранения состязаний элементов памяти при изменении состояний автоматов. Дано построение комбинационной схемы автомата.

Пятый раздел посвящен вопросам микропрограммирования. Рассмотрены принципы микропрограммного управления, набор операций автомата, состав и назначение элементов блок-схемы микропрограммного автомата. Рассмотрена работа устройства управления микропрограммным автоматом.

Шестой раздел посвящен проблемам отображения времени при проектировании дискретных автоматов. Рассмотрена модель тактируемого автомата, выбор и сравнение способов тактирования автомата. Дана характеристика сигналов в абсолютной и относительной шкале времени.

В седьмом разделе рассматриваются задачи моделирования синхронных процессов с помощью сетей Петри. Приведены графы сетей Петри, структура, маркировка и работа сетей Петри. Рассмотрены вопросы преобразования конечного автомата в сеть Петри.

1. ОСНОВЫ ТЕОРИИ ФОРМАЛЬНЫХ ГРАММАТИК

1.1. Основные понятия теории автоматов

Термин «автомат», как правило, используется в двух аспектах. С одной стороны, автомат - это устройство, выполняющее некоторые функции без непосредственного участия человека. В этом смысле мы говорим, что ЭВМ – автомат, так как после загрузки программы и исходных данных ЭВМ решает заданную задачу без участия человека. С другой стороны, термин «автомат» как математическое понятие обозначает математическую модель реальных технических автоматов. В этом смысле автомат представляется как «черный ящик», имеющий конечное число входов и выходов и некоторое множество внутренних состояний $Q = \{q_1(t), q_2(t), \dots, q_n(t)\}$, в которые он под действием входных сигналов переходит скачкообразно, т. е. практически мгновенно, минуя промежуточное состояние. Конечно, это условие не выполняется в реальности, так как любой переходный процесс длится конечное время.

Автомат называется **конечным**, если множество его внутренних состояний и множество значений входных сигналов – конечные множества.

На практике часто используется понятие цифрового автомата, под которым понимают устройство, предназначенное для преобразования информации. С общей точки зрения, процесс получения информации есть ни что иное, как процесс снятия неопределенности в результате того, что из некоторой совокупности возможных в данной конкретной ситуации явлений выделяется явление, фактически имевшее место. Таким образом, в понятии информации существенно не само произошедшее явление, а лишь его отношение к совокупности явлений, которые могли произойти.

Устройства, служащие для преобразования дискретной информации, называются **дискретными автоматами**.

В современных дискретных автоматах принято обычно отождествлять буквы используемого стандартного алфавита с цифрами той или иной системы счисления.

В состав цифровых автоматов обязательно входят запоминающие элементы (элементы памяти). Выходные сигналы в таких автоматах формируются в зависимости от входных сигналов и состояний, в которых находятся элементы памяти. Поэтому дискретные автоматы принято называть также **цифровыми автоматами**.

Основным качеством, выделяющим дискретные автоматы из числа всех других преобразователей информации, является наличие дискретного множества внутренних состояний и свойства **скачкообразного перехода** автомата из одного состояния в другое. Скачкообразность перехода означает возможность трактовать этот переход как мгновенный, хотя для любого реально существующего автомата имеет место конечная длительность переходных процессов, так что требование скачкообразности перехода не удовлетворяется.

Второе допущение состоит в том, что после перехода автомата в произвольное состояние переход в следующее состояние оказывается возможным не ранее, чем через некоторый фиксированный для данного автомата промежуток времени $\tau > 0$, так называемый интервал дискретности автомата. Это допущение дает возможность рассматривать функционирование цифрового автомата в **дискретном времени**. При построении автоматов с дискретным автоматным временем различают синхронные и асинхронные автоматы.

В **синхронных автоматах** моменты времени, в которые оказывается возможным изменение состояния автомата, определяются специальным устройством – **генератором синхронизирующих импульсов**. Соседние моменты времени оказываются при этом обычно разделенными равными временными промежутками.

В **асинхронных автоматах** моменты переходов из одного состояния в другое заранее не определены и могут совершаться через неравные между собой промежутки времени.

Изменения состояний цифрового автомата вызываются **входными сигналами**, которые возникают вне автомата и передаются в автомат по конечному числу входных каналов. В отношении входных сигналов цифровых автоматов принимаются два допущения: во-первых, для любого цифрового автомата число различных входных сигналов обязательно **конечно**, а, во-вторых, входные сигналы рассматриваются как **причина** перехода автомата из одного состояния в другое и относятся к моментам времени, определяемым соответствующими им переходами.

Отметим, что при таком допущении входной сигнал рассматривается как мгновенный, хотя в действительности он имеет конечную длительность. Особо следует подчеркнуть, что реальный физический входной сигнал, вызывающий изменение состояния автомата в момент времени t , может *кончиться до наступления этого момента*, однако, тем не менее, он относится именно к текущему моменту времени t , а не к предыдущему $(t-1)$.

Результатом работы цифрового автомата является выдача **выходных сигналов**, передаваемых из автомата во внешние цепи по конечному числу **выходных каналов**. В отношении выходных сигналов вводятся допущения, аналогичные допущениям для входных сигналов. Во-первых, число различных выходных сигналов для любого цифрового автомата всегда конечно. Во-вторых, каждому отличному от нуля моменту автоматного времени относится соответствующий ему входной сигнал. Реальный физический выходной сигнал $y(t)$, отнесенный к моменту времени t , появляется всегда после соответствующего этому же моменту времени входного сигнала $x(t)$. Что же касается момента времени t перехода автомата из состояния $q(t-1)$ в состояние $q(t)$, то сигнал $y(t)$ может фактически появиться либо раньше, либо позже этого момента.

В первом случае принимается, что выходной сигнал $y(t)$ однозначно определяется входным сигналом $x(t)$ и состоянием $q(t-1)$ автомата в предыдущий момент времени, во втором случае сигнал $y(t)$ однозначно определяется парой $(x(t), q(t))$. Будем считать, что для любого момента времени всегда имеет место лишь одна из этих возможностей (одновременно для всех переходов).

Цифровые автоматы, в которых выходной сигнал $y(t)$ определяется парой $(x(t), q(t-1))$, будем называть **автоматами первого рода**, а автоматы, в которых сигнал $y(t)$ определяется парой $(x(t), q(t))$, – **автоматами второго рода**.

Цифровой автомат (первого или второго рода) называется правильным, если выходной сигнал $y(t)$ определяется одним лишь его состоянием $(q(t-1)$ или $q(t))$ и не зависит явно от входного сигнала $x(t)$.

Автоматы первого рода обычно называют **автоматами Мили**, а автоматы второго рода – **автоматами Мура**.

Общая теория автоматов при сделанных выше допущениях разбивается на две большие части, которым присвоены названия **абстрактной теории автоматов** и **структурной теории автоматов**. Различие между ними заключается в том, что в абстрактной теории не учитываются структура как самого автомата, так и структуры

его входных и выходных сигналов. Входные и выходные сигналы рассматриваются при этом просто как буквы двух фиксированных для данного автомата алфавитов: входного и выходного. Не интересуясь способом построения автомата, абстрактная теория изучает лишь те переходы, которые претерпевает автомат под воздействием входных сигналов, и те выходные сигналы, которые он при этом выдает.

В противоположность абстрактной теории, структурная теория автоматов учитывает структуры автомата и его входных и выходных сигналов. В структурной теории изучаются способы построения автоматов из нескольких элементарных автоматов, способы кодирования входных и выходных сигналов элементарными сигналами, передаваемыми по реальным входным и выходным каналам.

Таким образом, структурная теория автоматов является продолжением и дальнейшим развитием абстрактной теории. В частности, задача синтеза идеализированного (без учета переходных процессов) цифрового автомата естественным образом подразделяется на этапы абстрактного и структурного синтеза.

Частным случаем дискретных автоматов являются автоматы, обладающие лишь одним внутренним состоянием. Такие автоматы **называются комбинационными схемами** или **автоматами без памяти**. Работа таких автоматов состоит в том, что они сопоставляют каждому входному сигналу $x(t)$ выходной сигнал $y(t)$.

Абстрактная теория автоматов без памяти совершенно тривиальна, а структурная теория таких автоматов много легче, чем теория произвольных автоматов с памятью. Основная идея излагаемой методики синтеза автоматов состоит в том, чтобы еще на уровне абстрактной теории преодолеть основные затруднения, вызванные наличием памяти, а на уровне структурной теории свести задачу синтеза автомата к задаче синтеза комбинационных схем.

1.2. Основные понятия теории формальных грамматик

В общем случае язык представляет собой бесконечное множество, а бесконечные объекты даже задать трудно: их невозможно задать простым перечислением элементов. Любой конечный механизм задания языка называется грамматикой.

Формальный язык представляет собой множество цепочек в некотором конечном алфавите. К формальным языкам можно отнести искусственные языки для общения человека с машиной – языки программирования.

Для задания описания формального языка необходимо, во-первых, указать алфавит, т. е. совокупность объектов, называемых символами (или буквами), каждый из которых можно воспроизводить в неограниченном количестве экземпляров (подобно обычным печатным буквам или цифрам), и, во-вторых, задать формальную грамматику языка, т. е. перечислить правила, по которым из символов строятся их последовательности, принадлежащие определяемому языку, – **правильные цепочки**.

Правила формальной грамматики можно рассматривать как продукции (правила вывода), то есть элементарные операции, которые, будучи применены в определенной последовательности к исходной цепочке (аксиоме), порождают лишь правильные цепочки. Сама последовательность правил, использованных в процессе порождения некоторой цепочки, является ее выводом. Определенный таким образом язык представляет собой **формальную систему**.

По способу задания правильных цепочек формальные грамматики разделяются на **порождающие** и **распознающие**. К порождающим относятся грамматики языка L ,

по которым можно построить любую «правильную» цепочку с указанием ее структуры и нельзя построить ни одной неправильной цепочки. Распознающая грамматика языка L – это грамматика, позволяющая установить, правильна ли произвольно выбранная цепочка и, если она правильна, то выяснить ее строение. Распознающая грамматика задает критерий принадлежности произвольной цепочки данному языку.

Формальные грамматики широко применяются в лингвистике и программировании в связи с изучением естественных языков и языков программирования.

Автоматные и лингвистические модели строятся на базе теории формальных грамматик, основы которой были заложены в работах Н. Хомского. Основными объектами, с которыми имеет дело эта теория, являются символы, представляющие собой базовые элементы какого-либо непустого множества A любой природы, а также цепочки, построенные из этих элементов. Множество A называют также **алфавитом**.

Символы будем обозначать строчными буквами латинского алфавита, а цепочки – в виде $ffghhh$, которые будем считать ориентированными слева направо. Цепочки будем обозначать также специальными символами – прописными буквами латинского алфавита или греческими буквами, например: $\gamma = ffg$, $B = abba$. Введем в рассмотрение пустую цепочку ε , не содержащую ни одного символа.

Длиной цепочки будем называть число символов, входящих в эту цепочку. Длина цепочки обозначается следующим образом:

$$\begin{aligned} |\gamma| &= |ffg| = 3; \\ |B| &= |abba| = 4; \\ |\varepsilon| &= 0. \end{aligned}$$

Конкатенацией двух цепочек X и Y называется такая цепочка Z , которая получается непосредственным слиянием цепочки X , стоящей слева, и цепочки Y , стоящей справа. Например, если $X = ffg$, $Y = ghh$, то конкатенация X и Y – это цепочка $Z = ffgghh$. Обозначим операцию конкатенации символом \circ . Свойства этой операции можно записать следующим образом:

- 1) свойство замкнутости:
 $\circ: A^* \times A^* \rightarrow A^*$;
- 2) свойство ассоциативности:
 $(\forall X \in A^*, Y \in A^*, Z \in A^*)$
 $[(X \circ Y) \circ Z = X \circ (Y \circ Z)],$

где через A^* обозначено множество всех возможных цепочек (разумеется, бесконечное), составленных из конечного множества A базовых элементов (символов) словаря, включая пустую цепочку ε ; символ \circ обозначает операцию декартова произведения двух множеств; а X, Y, Z – произвольные цепочки, принадлежащие A^* .

Рассмотрим пару (A^*, \circ) . С учетом перечисленных свойств операции \circ эта пара представляет собой полугруппу с единичным элементом ε или моноид. Полугруппой в алгебре называют только множество (в данном случае A^*), снабженное всюду определенной ассоциативной операцией.

Цепочка может принадлежать или не принадлежать языку L . Любое множество цепочек $L \leq A^*$ (где A^* – моноид), называется **формальным языком**, если это множество цепочек определено на алфавите A .

Пример 1. Пусть A – множество букв русского алфавита. Тогда множество цепочек, составленных из пяти букв, представляет собой формальный язык L_1 . Другой пример языка, определенного на том же алфавите – множество L_2 пятибуквенных

слов русского языка, которые можно разыскать в орфографическом словаре. Очевидно $L_2 \subset L_1$, так как многие цепочки языка L_1 не являются русскими словами.

Пусть B и C – некоторые подмножества множества A^* .

Произведением множеств B и C называется множество D цепочек, являющихся конкатенацией цепочек из B и C , т. е.

$$D = \{ X \circ Y \mid X \in B, Y \in C \}.$$

Обозначается произведение следующим образом:

$$D = BC.$$

Рассмотрим алфавит A . Обозначим множество, состоящее из ε , через A^0 . Определим степень алфавита как $A^n = A^{n-1} A$ для каждого $n \geq 1$.

Нетрудно показать, что множество всех возможных цепочек алфавита

$$A^* = \bigcup_{n=0}^{\infty} A^n.$$

Такое множество называют **итерацией** алфавита A . **Усеченной итерацией** алфавита A называют

$$A^+ = \bigcup_{n=1}^{\infty} A^n.$$

Если X и Y – цепочки множества A^* , то цепочку X называют **подцепочкой** цепочки Y , когда существуют такие цепочки U и V из A^* , что

$$Y = U \circ X \circ V.$$

При этом, если U – пустая цепочка, то подцепочку X называют **головой** цепочки Y , а если V – пустая цепочка, то X называют **хвостом** цепочки Y .

Конкатенация двух цепочек X и Y обозначается $X \circ Y$ или XY .

Рассмотрим пары цепочек $(P_1, Q_1), (P_2, Q_2), \dots, (P_n, Q_n)$ из $A^* \times A^*$.

Соотношениями Туэ будем называть правила, согласно которым любой цепочке $X = U P_i V$ из множества A^* будет ставиться в соответствие цепочка $Y = U Q_i V$, из того же множества A^* ($i = 1, 2, \dots, n$) и наоборот. Эти соотношения приводят к так называемым **ассоциативным исчислениям**.

Если цепочка Y получается из цепочки X однократным применением одного соотношения Туэ (т. е. заменой подцепочки P_i на подцепочку Q_i), будем говорить, что X и Y являются **смежными цепочками**.

Цепочка X_n соотносима с цепочкой X_0 , если существует последовательность цепочек

$$X_0, X_1, \dots, X_n,$$

такая, что X_{i-1} и X_i являются смежными цепочками.

Пример 2. Пусть A – множество букв русского алфавита, на котором определим соотношение Туэ, заключающееся в праве замены любой одной буквы слова на любую другую. Тогда в последовательности цепочек МУКА, МУЗА, ЛУЗА, ЛОЗА, ПОЗА, ПОРА, ПОРТ, ТОРТ, две любые соседние цепочки являются смежными, а цепочки МУКА и ТОРТ являются соотносимыми в смысле заданных соотношений.

Введение соотношений Туэ позволяет выделить среди множества языков определенные их классы, которые используются при построении автоматно-лингвистических моделей самого различного типа.

Соотношения Туэ являются двусторонними, если цепочка X является смежной по отношению к цепочке Y , и наоборот, цепочка Y является смежной по отношению к

цепочке X . Более интересными, с точки зрения теории формальных грамматик, являются соотношения, в которых введено направление.

В этом случае их называют полусоотношениями Туэ или продукциями и обозначают следующим образом:

$$(P_1 \rightarrow Q_1), (P_2 \rightarrow Q_2), \dots, (P_n \rightarrow Q_n).$$

В том случае, когда имеется набор продукций, говорят, что цепочка Y непосредственно порождается из цепочки X , и обозначается как $X \Rightarrow Y$, если существуют такие цепочки U и V , что

$$X = U P_i V, \quad Y = U Q_i V,$$

а $(P_i \rightarrow Q_i)$ – продукция из данного набора.

Говорят также, что X порождает Y .

Если существует последовательность цепочек X_0, X_1, \dots, X_n такая, что для каждого $i = 1, 2, \dots, n$

$$X_{i-1} \Rightarrow X_i,$$

то говорят, что X_n порождается из X_0 (X_0 порождает X_n), и обозначают как $X_0 \Rightarrow^* X_n$.

Грамматики Хомского соответствуют формальным комбинаторным схемам, являющимся полусистемами Туэ, в основу которых положены полусоотношения Туэ (продукции).

1.3. Классификация языков по Хомскому

Теория формальных языков (формальных грамматик) занимается описанием, распознаванием и переработкой языков. Описание любого языка должно быть конечным, хотя сам язык может содержать бесконечное множество цепочек. Полезно иметь возможность описания отдельных типов языков, имеющих те или иные свойства, т. е. иметь различные типы конечных описаний.

Предположим, что имеется некоторый класс языков L , который задается определенным типом описаний. Теория формальных языков позволяет ответить на ряд вопросов, возникающих во многих прикладных задачах, в которых используются автоматически-лингвистические модели. Например, могут ли языки из класса L распознаваться быстро и просто; принадлежит ли данный язык классу L и т. д. Важной проблемой является построение алгоритмов, которые давали бы ответы на определенные вопросы о языках из класса L , например: «Принадлежит цепочка X языку L или не принадлежит?»

Существуют два основных способа описания отдельных классов языков. Первый из них основан на ограничениях, которые налагаются на систему полусоотношений Туэ (продукций), на базе которых определяются **грамматики** как механизмы, порождающие цепочки символов. Другим способом является определение языка в терминах множества цепочек, с помощью некоторого распознающего устройства. Такие устройства будем называть **автоматами** (автоматами-распознавателями).

Н. Хомский определил четыре типа грамматик, на основе которых оцениваются возможности других способов описания языков.

Пусть алфавит символов (непустое конечное множество), из которых строятся цепочки языка L , представляет собой алфавит терминальных символов V_T . Очевидно, что $L \leq V_T^*$.

Определение формальной грамматики требует наличия еще одного алфавита V_N – непустого конечного множества нетерминальных символов ($V_N \cap V_T = \emptyset$). Объединение этих алфавитов назовем словарем формального языка L :

$$V = V_N \cup V_T.$$

Условимся обозначать элементы алфавита V_T строчными латинскими буквами, элементы множества V_N – прописными латинскими буквами, элементы словаря V^* (цепочки символов словаря) – греческими буквами.

Определим также множество упорядоченных пар (полутуэвских соотношений) следующим образом:

$$\Pi = \{(\alpha, \beta) \mid \alpha \in V^* V_N V^* \wedge \beta \in V^+\}.$$

Каждая пара (α, β) называется продукцией и обозначается как $\alpha \rightarrow \beta$.

Заметим, что β является элементом усеченной итерации словаря, поэтому среди продукций нет пар вида $\alpha \rightarrow \varepsilon$, где ε – пустая цепочка.

Формальная грамматика G – это совокупность четырех объектов:

$$G = (V_T, V_N, P, S),$$

где P – непустое конечное подмножество Π , а $S \in V_N$ – начальный символ.

Типы грамматик по Хомскому обозначают: тип 0, тип 1, тип 2 и тип 3. Соответствующий тип грамматики определяется теми ограничениями, которые налагаются на продукцию P .

Если таких ограничений нет, грамматика принадлежит к типу 0.

Единственное ограничение, налагаемое на длину цепочек α и β :

$$|\alpha| \leq |\beta|,$$

относит грамматики к типу 1. Такие грамматики также называют контекстно-зависимыми, то есть грамматиками непосредственных составляющих (НС-грамматиками).

В том случае, когда цепочка α состоит из одного символа, т. е. $\alpha \in V_N$, грамматики относят к типу 2. В этом случае их называют бесконтекстными (контекстно-свободными или КС-грамматиками).

Наконец, регулярными грамматиками (типа 3) называют такие, для которых $\alpha \in V_N$, а $\beta \in V_T V_N$, либо $\beta \in V_T$. Иными словами, правые части продукций регулярных грамматик состоят либо из одного терминального и одного нетерминального символов, либо из одного терминального символа.

Языком $L(G)$, порождаемым грамматикой G , будем называть множество цепочек $\alpha \in V_T^*$, каждая из которых порождается из начального символа S в смысле полутуэвских соотношений P данной грамматики. Другими словами,

$$L(G) = \{\alpha \mid \alpha \in V_T^* \wedge S \Rightarrow^* \alpha\}.$$

Нетрудно видеть, что каждая регулярная грамматика является бесконтекстной, а каждая бесконтекстная грамматика является контекстно-зависимой. В свою очередь, каждая контекстно-зависимая грамматика – это грамматика типа 0. Обратное утверждение неверно. Очевидно, что имеется некоторая иерархия грамматик, которой соответствует иерархия формальных языков, каждый из них может быть порожден некоторой формальной грамматикой. При этом тип языка соответствует типу той грамматики, с помощью которой он может быть порожден.

С другой стороны, типы языков могут быть определены типами абстрактных распознающих устройств (автоматов). При этом язык определяется как множество цепочек, допускаемых распознающим устройством определенного типа. На рис. 1.1

приведена иерархия языков и соответствующие ей иерархии грамматик и автоматов как распознающих устройств.

Любое множество, порождаемое автоматическим устройством произвольного вида, порождается некоторой грамматикой типа 0 по Хомскому. Заметим, что для любого естественного языка, в принципе, возможно построить математическую модель, использующую такую грамматику.

Таким образом, грамматики типа 0 представляют собой порождающие устройства очень общего характера. А те формальные языки, с которыми имеют дело автомато-лингвистические модели (язык программирования, ограниченные естественные языки), как показывает практика, всегда описываются языками типа 1 или 2.

Языки типа 3, которые называют автоматными языками, языками с конечным числом состояний, нашли широкое применение в исследовании электронных схем, а также в ряде других областей (например, исследование цепей Маркова).

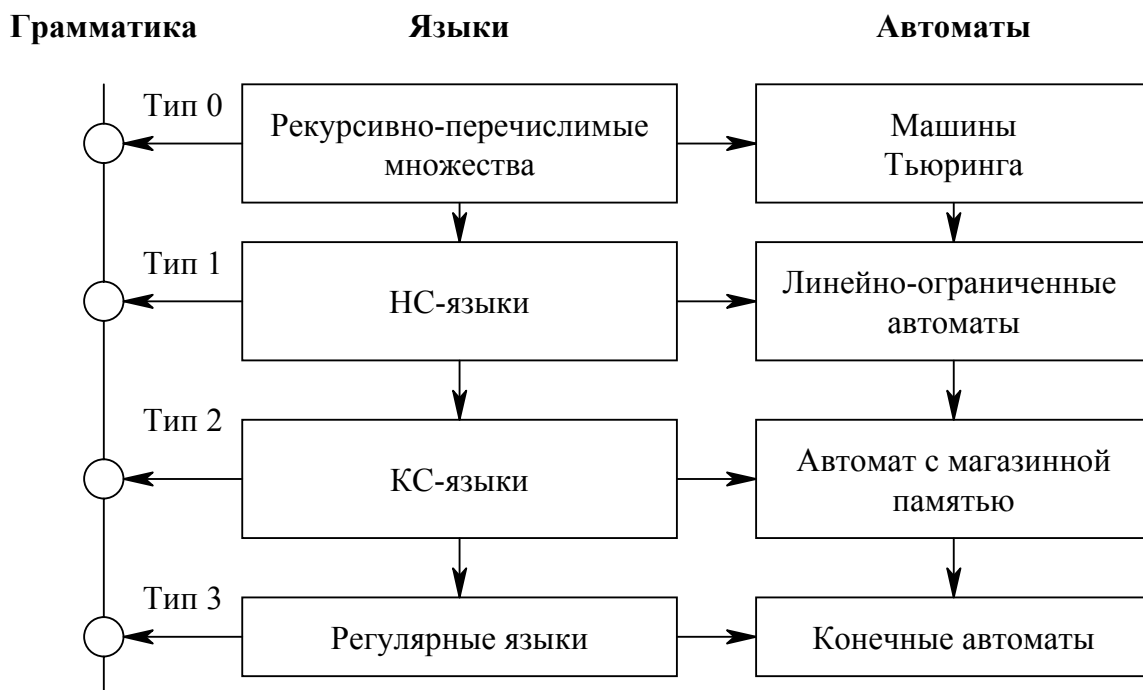


Рис. 1.1. Иерархия грамматик, языков и автоматов

1.4. Распознающие устройства и автоматы

Распознавание образов (объектов, сигналов, ситуаций или процессов) – едва ли не самая распространенная задача, которую человеку приходится решать практически каждую секунду от первого до последнего дня своего существования. Для решения этой задачи человек использует огромные ресурсы своего мозга, включая одновременно 10 – 12 миллиардов нейронов. Именно это дает возможность людям мгновенно узнавать друг друга, с большой скоростью читать печатные и рукописные тексты, безошибочно водить автомобили в сложном потоке уличного движения, осуществлять обработку деталей на конвейере, дешифровать аэро- и космические фотоснимки, разгадывать коды, древнюю египетскую клинопись и т. д.

Потребности в комплексной механизации и автоматизации, создании роботов, решении задач технической и медицинской диагностики, метеорологического прогноза, формализованной оценки общественных, экономических и социологических

явлений и процессов, в определении наиболее вероятных направлений их трансформаций и предопределили значительные усилия научной и инженерной мысли, направленные на решение теоретических и прикладных вопросов проблем распознавания.

Распознавание представляет собой задачу преобразования входной информации в выходную, представляющую собой заключение о том, к какому классу относится распознаваемый образ. В качестве входной информации используются некоторые параметры или признаки распознаваемых образов.

1.4.1. Концепция порождения и распознавания

На основе теории распознавания образов можно построить модель процесса распознавания и применительно к ней рассмотреть основные теоретические положения и понятия.

К одному из основных понятий относят понятие **класс** или **образ**.

Известно, что отдельные предметы или явления обладают общими друг с другом свойствами и имеют некоторые отличительные свойства. **Классом** (образом) можно назвать множество предметов или объектов, обладающих (объединенных) некоторыми общими свойствами.

Например, к классу «карандаш» относятся карандаши всех размеров и цветов, к классу «красный карандаш» относятся карандаши с красным грифелем, к классу «пишущие устройства» относятся карандаши, ручки, мел и пр.

Как правило, имеется набор классов или алфавит классов (образов):

$$A = \{A_1, A_2, \dots, A_i, \dots, A_m\},$$

где A_i – отдельный i -класс; m – общее число классов.

Если $m = 1$, то никакого распознавания не нужно. Очень часто рассматривается задача отнесения объекта к одному из двух ($m = 2$) классов. Случай $m = \infty$ практически нереальный и рассматриваться не будет.

Следующее важное понятие – это **объект**, или **реализация**, или **образ**.

Каждый класс в алфавите образов может быть представлен некоторым количеством объектов или реализацией. Например, имеется по 100 штук красных, желтых и зеленых карандашей разных оттенков и размеров. Совокупность различных реализаций для всех классов образует множество возможных реализаций:

$$B = \{b_1, b_2, \dots, b_j, \dots, b_T\}.$$

В большинстве случаев T – конечно и $T \gg m$.

Определение понятия **признак класса** вызывало и вызывает большие споры. При введении понятия класса указывается, что в класс объединяются образы, имеющие общие свойства. Эти свойства и составляют признаки данного класса. Обычно признаки задаются своими количественными значениями.

Для простоты будем считать, что все классы характеризуются одним и тем же количеством признаков N . Обозначим совокупность признаков для данного алфавита A следующим образом:

$$X = \{x_1, x_2, \dots, x_k, \dots, x_N\}.$$

В качестве примера можно привести диагностику какого-либо заболевания (например, гриппом). Тогда условно можно считать, что

x_1 – это температура;

x_2 – кашель (сильный, слабый, средний, отсутствует);

x_3 – покраснение горла и т. д.

Практически числовые значения признаков изменяются в некоторых пределах. Каждый признак x_k может принимать одно значение из совокупности:

$$x_k = \{x_k^1, x_k^2, \dots, x_k^p, \dots, x_k^R\}.$$

Каждая конкретная реализация b_j задается совокупностью значений признаков:

$$b_j = \{x_1^{j1}, x_2^{j2}, \dots, x_k^{jk}, \dots, x_N^{jN}\},$$

которая называется *описанием реализации*.

В теории распознавания часто прибегают к геометрической интерпретации. Значения каждого признака откладываются по соответствующим осям прямоугольной системы координат. Если взять случай двух признаков ($N = 2$), то для случая трех градаций числовых значений признаков пространство признаков будет иметь вид (рис. 1.2).

Каждая из реализаций задается двумя значениями признаков:

$$b_2 = \{x_1^3, x_2^2\}; \quad b_3 = \{x_1^3, x_2^3\} \text{ и т. д.}$$

Количество возможных реализаций при N признаках, R градациях может быть определено как $T = R^N$.

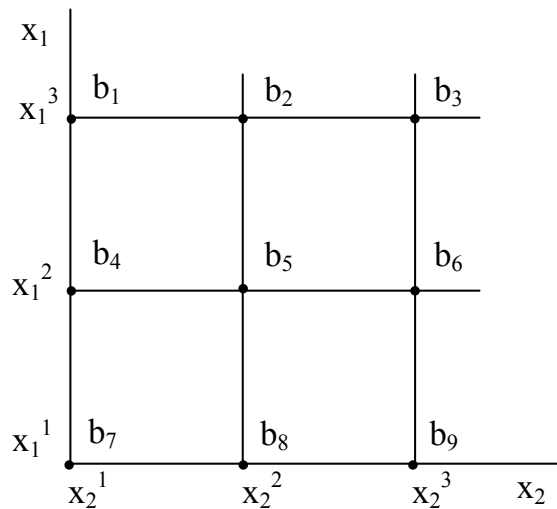


Рис. 1.2. Пространство признаков ($N = 2$) для случая трех градаций

В большинстве задач распознавания имеется 2 этапа: обучение распознаванию на заданном количестве эталонных образов, принадлежность которых к определенному классу известна, и собственно распознавание, когда предъявляется реализация (объект) с неизвестной принадлежностью и требуется определить, к какому классу относится объект.

При рассмотрении моделей распознавания образов очень часто применяется понятие *распознающей машины*, под которой понимается некоторая система. На вход этой системы поступают объекты для распознавания, а на выходе появляются распознанные объекты, отнесенные к определенному классу.

Для каждого класса имеется определенное количество объектов, на которых происходит обучение. Геометрически в случае двух классов эти объекты будут представлены в виде двух многообразий точек (рис. 1.3).

Эти две области могут не перекрываться (рис. 1.3, а) или перекрываться (рис. 1.3, б). Очень часто в области перекрытия распознающая машина дает отказ от распознавания.

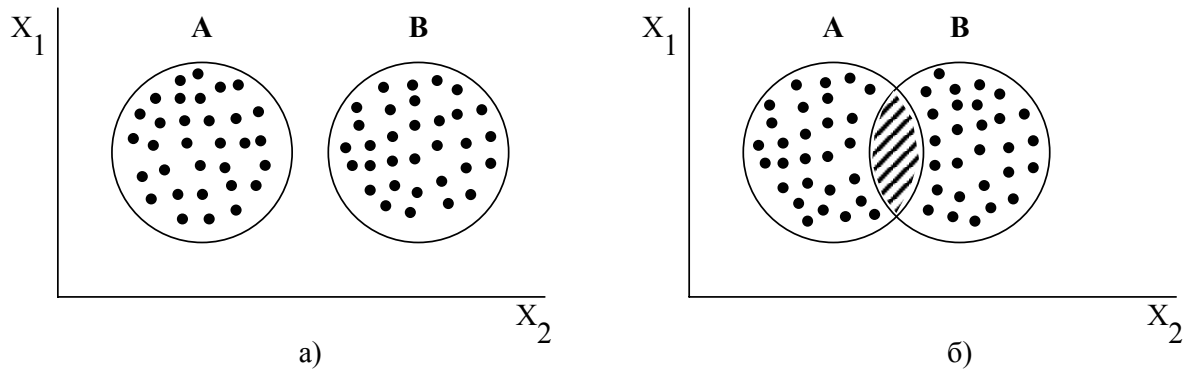


Рис. 1.3. Геометрическая интерпретация объектов двух различных классов.
а) области объектов не перекрываются; б) перекрываются

Одна из интерпретаций процесса обучения состоит в том, что необходимо максимально сжать множество эталонов каждого образа, максимально удалив их друг от друга. После деформации пространства признаков (в результате обучения) начинается собственно процесс распознавания. Обученной машине предъявляется образ, новая точка в деформированном пространстве, и требуется отнести ее к одному из двух образов.

Одним из способов распознавания, критерием распознавания, является классификация объектов по расстоянию от центральной точки (центров тяжести или рассеяния) множества данного образа или по сумме расстояния до всех эталонов данного образа. Если точка (объект) ближе к множеству первого образа, то его отождествляют с этим образом (рис. 1.4).

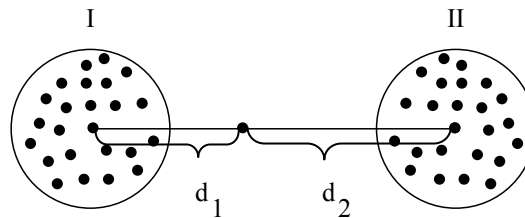


Рис. 1.4. Объект относится к I классу ($d_1 < d_2$)

1.4.2. Распознающие, порождающие и преобразующие формальные грамматики

Трудности, возникающие при решении задач распознавания образов (классов) привели к разработке нового метода распознавания – структурного, получившего также название лингвистического или синтаксического. Его особенность заключается в том, что априорными описаниями классов являются структурные описания – формальные конструкции, при получении которых последовательно реализуется вначале учет иерархичности структуры объекта, а затем учет отношений, существующих между отдельными элементами этой иерархии, в пределах одних и тех же уровней между ними.

Поскольку практическая реализация структурных методов распознавания образов основана на использовании некоторых идей и методов математической лингвистики, рассмотрим ее основные понятия.

Термин *формальная грамматика* представляет собой общее название нескольких типов исчислений, используемых в математической лингвистике для описания строения естественных языков, а также некоторых искусственных языков, в частности, языков программирования.

Под грамматиками в математической лингвистике понимают некоторые специальные системы правил, задающие (или характеризующие) множества цепочек (конечных последовательностей) символов. Эти объекты могут интерпретироваться как языковые объекты различных уровней, например, как словоформы, словосочетания и предложения (цепочки словоформ) и т. п.

Следовательно, формальные грамматики имеют дело с абстракциями, возникающими в результате обобщения таких стандартных лингвистических понятий как словоформа, словосочетание и предложение. Из определенного набора символов (обозначающих, например, все словоформы русского языка) можно строить произвольные цепочки; одни из них естественно считать правильными или допустимыми, а другие – неправильными или недоступными.

Формальные грамматики задают правильные цепочки, если для любой предъявленной цепочки грамматика позволяет установить, является или нет эта цепочка правильной, и в случае положительного ответа дается указание о строении этой цепочки, либо грамматика позволяет построить любую правильную цепочку, давая при этом указания о ее строении, и не строит ни одной неправильной цепочки. В первом случае формальная грамматика называется *распознающей*, во втором – *порождающей*.

Формальные грамматики обладают двумя существенными особенностями. Во-первых, существующие формальные грамматики описывают только совокупность возможных результатов, не давая прямых указаний, как именно можно получить результат, соответствующий определенной исходной задаче. Во-вторых, в формальных грамматиках все утверждения формируются исключительно в терминах небольшого числа четко определенных и весьма элементарных символов и операций. Это делает формальные грамматики очень простыми с точки зрения их логического строения и обеспечивает изучение их свойств дедуктивными методами.

Структурные методы распознавания базируются на *порождающей грамматике* – системе, состоящей из четырех частей: основной, или терминальный словарь; вспомогательный словарь; начальный символ; набор правил подстановки исходных элементов, из которых строят цепочки, порождаемые грамматикой. Элементы основного словаря называют основными (терминальными) символами.

Вспомогательный (нетерминальный) словарь. Это набор символов, которыми обозначаются классы исходных элементов или цепочек исходных элементов, а также в отдельных случаях некоторые специальные элементы (вспомогательные или нетерминальные).

Начальный символ. Это выделенный нетерминальный символ, обозначающий совокупность (класс) всех тех языковых объектов, для описания которых предназначается данная грамматика. Так в грамматике, порождающей предложения, начальным будет символ, означающий предложение; в грамматике, порождающей допустимые слоги, начальный символ означает слог, и т. п.

Правила подстановки. Это выражения вида « $X \rightarrow Y$ », « X вместо Y », где X и Y – цепочки, содержащие любые терминальные или нетерминальные символы.

Для описания собственно процесса порождения, т. е. как грамматика применяется, необходимо введение таких понятий, как **непосредственная выводимость; язык, порожденный грамматикой.**

Непосредственная выводимость. Если имеются цепочки X и Y , которые можно представить в виде $X = Z_1 a Z_2$ и $Y = Z_1 b Z_2$, где $a \rightarrow b$ – одно из правил грамматики G , то говорят, что цепочка Y непосредственно выводима из цепочки X в грамматике G . Другими словами, цепочка X может быть переработана в цепочку Y за один шаг применением одной подстановки: X получается из Y подстановкой b на место некоторого вхождения цепочки a . Это обозначается как $X / G = Y$.

Выводимость. Если имеется последовательность цепочек X_0, X_1, \dots, X_n , в которой каждая следующая цепочка непосредственно выводима из предыдущей, то цепочка X_n выводима из цепочки X_0 . Последовательность цепочек X_0, X_1, \dots, X_n называется **выводом X_n из X_0 – в грамматике G .**

Существенно, что порождающая грамматика не есть алгоритм, поскольку правила подстановки представляют собой не последовательность предписаний, а совокупность решений. Это означает, что, во-первых, правило вида $a \rightarrow b$ понимается в грамматике как « a можно заменить на b » (но можно и не заменять); в алгоритме же $a \rightarrow b$ означало бы « a следует заменить на b » (нельзя не заменять); во-вторых, порядок применения правил в грамматике произволен: любое правило, в принципе, разрешается применять после любого.

Язык, порожденный грамматикой. Совокупность всех терминальных цепочек, т. е. цепочек, состоящих только из терминальных символов, выводимых из начального символа в грамматике G , называется языком, порожденным грамматикой G , и обозначается $L(G)$.

Следовательно, применение грамматики – это построение полных выводов, последние цепочки которых и образуют язык, порожденный грамматикой.

Две различные грамматики могут порождать один и тот же язык, т. е. одно и то же множество терминальных цепочек. Такие грамматики называются **эквивалентными грамматиками.**

1.5. Автоматы и формальные языки

1.5.1. Понятие об информации и ее преобразованиях

Понятие информации принадлежит к числу важнейших понятий современной науки. Важность этого понятия обуславливается его всеобщностью: с понятием информации мы сталкиваемся при изучении любого явления, происходящего в природе или обществе.

Существуют два различных подхода к изучению явлений с информационной точки зрения: непрерывный и дискретный. При непрерывном подходе все изучаемые явления рассматриваются как переменные векторные поля. Задание информации состоит в выборе какого-либо определенного (переменного) поля из фиксированной за-

ранее совокупности таких полей. Характерным для непрерывного подхода является то, что все описывающие явление величины (компоненты векторов, пространственные и временные координаты) являются вещественными числами и могут изменяться непрерывно.

При дискретном подходе также имеют дело с переменными векторными полями. Однако, в отличие от предыдущего случая, компоненты векторов, а также пространственные и временные координаты принимают дискретные ряды значений. Наиболее употребительным является случай, когда число значений принимаемых компонентами векторов и пространственными координатами конечно (поле задано в конечном числе точек).

Задание информации при дискретном подходе сводится, таким образом, к заданию конечных последовательностей конечнозначных (постоянных) векторных полей. Если дискретная информационная задача фиксирована, то количество различных постоянных векторных полей, возможных в этой задаче, в соответствии с принятыми условиями конечно. Вводя для каждого такого поля специальное буквенное обозначение, мы получаем возможность задавать информацию конечными последовательностями букв. Подобный способ задания дискретной информации условимся называть алфавитным, совокупность элементарных символов (букв), из которых составляется информация – *алфавитом*, а конечные последовательности букв алфавита – *словами в данном алфавите*.

Понятие *слова в алфавите* существенно отличается от понятия слова в обычном языке даже в том случае, когда исходный алфавит совпадает, например, с русским алфавитом. Различие заключается в том, что в силу принятых нами определений, мы будем называть словами не только все фактически существующие слова русского языка, но также и любые бессмысленные сочетания букв.

Алфавитный способ задания информации является достаточно универсальным. Действительно, этим способом можно осуществить представление любой дискретной информации. Что же касается информации, задаваемой в непрерывной форме, то на практике, применяя методы аппроксимации, ее всегда можно представить с любой степенью точности в дискретной форме. С точки зрения устройства, воспринимающего информацию, любая непрерывная информация сводится, фактически, к дискретной, а следовательно, в конечном счете, к алфавитной информации. Отсюда следует универсальность алфавитного способа задания информации.

Роль дискретных (алфавитных) методов задания информации особенно возросла после того, как появились мощные автоматы для преобразования дискретной информации (ЭВМ с программным управлением).

Подобно тому как со всяким явлением в природе или в обществе связана несущая этим явлением информация, взаимосвязь явлений приводит к понятию о *преобразовании информации*.

Предположим, что любое явление α из некоторого класса A явлений влечет за собой некоторое определенное явление β из того же самого или любого другого класса B явлений. В таком случае говорят, что нам задано преобразование информации $\alpha \rightarrow \beta = f(A)$. Следовательно, с абстрактной точки зрения преобразование информации есть ни что иное, как *отображение* одного класса явлений в другой класс явлений.

Стрелка « \rightarrow » здесь служит в качестве знака преобразования информации, или, говоря иначе, в качестве обозначения отображения явления α в явление β .

Если подобное отображение осуществляется каким-либо определенным объектом, то этот объект называется преобразователем информации. Например, обыкновенный радиоприемник, с информационной точки зрения, представляет собой преобразователь информации, осуществляющий преобразование информации, заданной в виде радиоволн, в информацию, задаваемую с помощью звуковых колебаний. Человек, решающий какую-либо задачу, также может рассматриваться как преобразователь информации: входной информацией в этом случае является условие задачи, а выходной – ответ.

1.5.2. Преобразование алфавитной информации

В соответствии с высказанными ранее соображениями будем рассматривать исключительно дискретно заданную алфавитную информацию и ее преобразования.

В наиболее общем виде преобразование алфавитной информации может быть задано следующим образом. Пусть нам даны два конечных алфавита $X = (x_1, x_2, \dots, x_m)$ и $Y = (y_1, y_2, \dots, y_n)$. Обозначим через $F = F(x_1, \dots, x_m)$ совокупность всех слов конечной длины в алфавите X , а через $G = G(y_1, \dots, y_n)$ – совокупность всех слов конечной длины в алфавите Y . Если исходная информация записывается в алфавите X , а конечная информация – в алфавите Y , то произвольное преобразование информации φ будет представлять собой отображение множества F в множество G . В дальнейшем мы будем рассматривать лишь детерминированные преобразования информации, при которых входное слово полностью определяет слово на выходе преобразователя. Это требование есть ни что иное, как требование *однозначности* отображения φ .

В самом общем случае целесообразно считать преобразование информации φ частичным отображением. Иначе говоря, задавать отображение φ не обязательно на всем множестве F , а лишь на части этого множества (не исключая, впрочем, случай совпадения этой части со всем множеством F). Введение частичных отображений позволяет вместо отображений одного множества слов в другое рассматривать лишь отображения таких множеств в себя. Для этой цели достаточно ввести объединенный алфавит $Z = (x_1, \dots, x_m, y_1, \dots, y_n)$ и множество $H = H(x_1, \dots, x_m, y_1, \dots, y_n)$ слов над этим алфавитом. Теперь вместо отображения множества F в множество G можно рассматривать частичное отображение множества H в себя. Это частичное отображение будет определено для слов, состоящих только из букв x_1, x_2, \dots, x_m .

Однозначность отображения φ множества F в множество G не означает, вообще говоря, однозначности обратного отображения φ^{-1} . Если такая однозначность имеет место, то отображение φ называется взаимно однозначным. В этом случае отображение φ осуществляет эквивалентное преобразование информации. В случае эквивалентного преобразования заключительная (выходная) информация полностью определяет исходную (входную) информацию.

Пример 1. Преобразование информации в произвольном конечном алфавите заключается в зеркальном отображении слов: это преобразование эквивалентно.

Пример 2. В алфавите, состоящем из всех десятичных цифр и знака суммы, задается отображение слов вида $a + b$, где a и b – целые числа, преобразуемые в сумму этих двух чисел. Например, слово $2 + 5$ преобразуется в слово 7.

Легко видеть, что это преобразование не эквивалентно, поскольку задание суммы не определяет слагаемых.

Со всяким преобразованием информации связывается представление о его *сложности*.

По критерию сложности целесообразно выделить *простейшие преобразования* информации. Простейшими или побуквенными, будем называть преобразования, заключающиеся в замене каждой буквы исходного алфавита некоторой определенной комбинацией букв нового алфавита, имеющего заранее фиксированную, одинаковую для всех букв длину.

Пример 3. Исходная информация – произвольное слово в русском алфавите. Преобразование информации состоит в замене каждой буквы алфавита порядковым номером, записанным в десятичной системе счисления. При этом для соблюдения условия равенства длин комбинаций буквенного алфавита, представляющих буквы старого алфавита, необходимо первую букву обозначать комбинацией 01, вторую – 02 и т. д. В результате такого преобразования слово «дом» преобразуется в слово «051412». Это преобразование является не только простейшим, но и эквивалентным.

Пример 4. Исходная информация – произвольное целое десятичное число (слово в алфавите, состоящем из десяти цифр 0, 1, 2, ..., 9). Преобразование информации состоит в замене каждой четной цифры нулем, а нечетной – единицей. Слово «125» при этом преобразуется в слово «101», слово «0342» – в слово 0100 и т. д.

Отсюда видно, что с помощью простейших преобразований, информацию, заданную в любом конечном алфавите, можно записать в алфавите, содержащем только две буквы. Такой алфавит называют «двоичным алфавитом», а две его буквы будем обозначать нулем и единицей. Эти преобразования носят специальное название *двоичного кодирования* исходной информации.

Таким образом, при различных преобразованиях информации можно, не нарушая общности, предполагать, что как исходная, так и заключительная информация задана в некотором стандартном (например, двоичном) алфавите.

1.5.3. Способы задания автоматов

Абстрактные автоматы задаются с помощью таблиц переходов (выходов), графов, матриц переходов.

Таблица переходов (таблица выходов) автомата Мили представляет собой таблицу, в которой левый столбец обозначается входными сигналами, а верхняя строка – состояниями, начиная с начального состояния. На пересечении строки и столбца указывается следующее состояние, в которое переходит автомат (в таблице переходов) или выходной сигнал, выдаваемый им (в таблице выходов).

Представим автоматы Мили и Мура табличным способом.

Описание работы автомата Мили таблицами переходов и выходов иллюстрируется на примере автомата A1 (рис. 1.5): X – алфавит входных сигналов; Y – алфавит выходных сигналов; Q – алфавит состояний; δ – функция переходов; λ – функция выходов.

$\delta : Q \times X \rightarrow Q^*$				$\lambda : Q \times X \rightarrow Y^*$			
	q_1	q_2	q_3		q_1	q_2	q_3
x_1	q_2	q_3	q_2	x_1	y_1	y_3	y_3
x_2	q_3	q_2	q_1	x_2	y_2	y_1	y_1

а)
б)

Рис. 1.5. Таблицы переходов (а) и выходов (б) автомата А1

Автомат, имея один вход и один выход, работает в дискретном времени, принимая значения $t = 1, 2, 3, \dots$. На вход автомата поступают входные сигналы x_f (например, сигналы x_1 и x_2). В каждый момент t автомат находится в некотором состоянии $q(t)$, начиная с начального состояния q_1 .

На пересечении столбца q_m и строки x_f в таблице переходов ставится состояние $q_s = \delta(q_m, x_f)$, в которое автомат переходит из состояния q_m под действием сигнала x_f , а в таблице выходов – соответствующий этому переходу выходной сигнал $y_g = \lambda(q_m, x_f)$.

Иногда при задании автоматов Мили используют одну совмещенную таблицу переходов и выходов, в которой на пересечении столбца q_m и строки x_f записываются в виде q_s / y_g следующее состояние и выдаваемый выходной сигнал. На рис. 1.6 представлена совмещенная таблица автомата А1.

$\delta : Q \times X \rightarrow Q; \quad \lambda : Q \times X \rightarrow Y$			
	q_1	q_2	q_3
x_1	q_2 / y_1	q_3 / y_3	q_2 / y_3
x_2	q_3 / y_2	q_2 / y_1	q_1 / y_1

Рис. 1.6. Совмещенная таблица автомата А1

Так как в автомате Мура выходной сигнал зависит только от внутреннего состояния и не зависит от входного сигнала, то он задается одной *отмеченной* таблицей переходов, в которой каждому ее столбцу приписан, кроме состояния q_m еще и выходной сигнал $y_g = \lambda(q_m)$, соответствующий этому состоянию. Пример табличного описания автомата Мура А2 (рис. 1.7).

Для частичных автоматов, у которых функции δ или λ определены не для всех пар $(q_m, x_f) \in Q \times X$, на месте неопределенных состояний и выходных сигналов ставится прочерк.

$\delta : Q \times X \rightarrow Q; \quad \lambda : Q \times X \rightarrow Y$					
	y_1	y_1	y_3	y_2	y_3
	q_1	q_2	q_3	q_4	q_5
x_1	q_2	q_5	q_5	q_3	q_3
x_2	q_4	q_2	q_2	q_1	q_1

Рис. 1.7. Таблица переходов автомата Мура

Часто автомат задают с помощью графа автомата. Граф автомата – ориентированный граф, вершины которого соответствуют состояниям, а дуги – переходам меж-

ду ними. Две вершины графа автомата q_m и q_s (исходное состояние и состояние перехода) соединяются дугой, направленной от q_m к q_s , если в автомате имеется переход из q_m в q_s , т. е. если $q_s = \delta(q_m, x_f)$ при некотором $x_f \in X$. Данной дуге (q_m, q_s) приписывается входной сигнал x_f и выходной сигнал $y_g = \lambda(q_m, x_f)$. При этом выходной сигнал y_g записывается внутри вершины q_m или рядом с ней. На рис. 1.8 и 1.9 приведены графы автоматов A1 и A2, описанных ранее табличным способом.

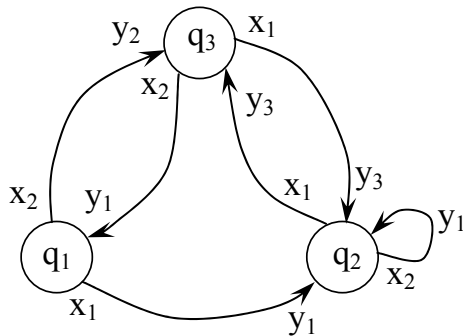


Рис. 1.8. Граф автомата A1

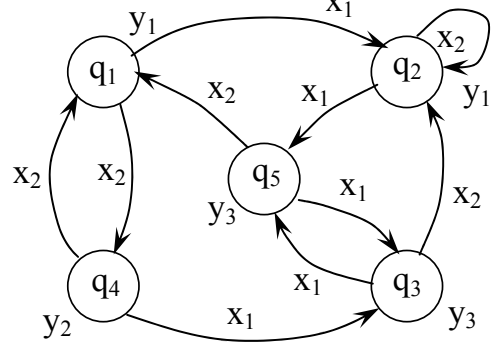


Рис. 1.9. Граф автомата A2

Любой автомат может быть задан с помощью графа, но не всякий граф в алфавитах Q, X, Y задает автомат. В графе автомата не должно существовать двух дуг с одинаковыми входными сигналами, выходящих из одной и той же вершины (условие однозначности).

Иногда применяется способ задания автомата с помощью матрицы переходов и выходов, которая представляет собой таблицу с двумя входами. Строки и столбцы таблицы отмечены состояниями. Если существует переход из состояния q_m под действием входного сигнала x_f в состояние q_s , с выдачей выходного сигнала y_i , то на пересечении строки q_m и столбца q_s записывается пара x_f / y_i .

Для автомата Мура используется матрица, столбцы которой отмечены выходными сигналами y_i , а на пересечении ее строк и столбцов указываются только входные сигналы x_f .

Ниже приведены матрицы переходов и выходов для рассмотренных ранее автоматов S1 и S2 (рис. 1.10).

	q_1	q_2	q_3
q_1		x_1/y_1	x_2/y_2
q_2		x_2/y_1	x_1/y_3
q_3	x_2/y_1	x_1/y_3	

а

	y_1	y_1	y_3	y_2	y_3
	q_1	q_2	q_3	q_4	q_5
q_1		x_1		x_2	
q_2		x_2			x_1
q_3		x_2			x_1
q_4	x_2		x_1		
q_5	x_2		x_1		

б

Рис. 1.10. Матрицы переходов и выходов автоматов A1 (а) и A2 (б)

Рассмотрим пример составления графа автомата. Пусть задано некоторое устройство в виде таблицы (табл. 1) и граф переходов к ней (рис. 1.11).

Таблица 1

Входы			Выходы		
x_1	x_2	x_3	y_1	y_2	y_3
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	1	1	1	0
1	0	0	1	1	1*
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	-	-	-

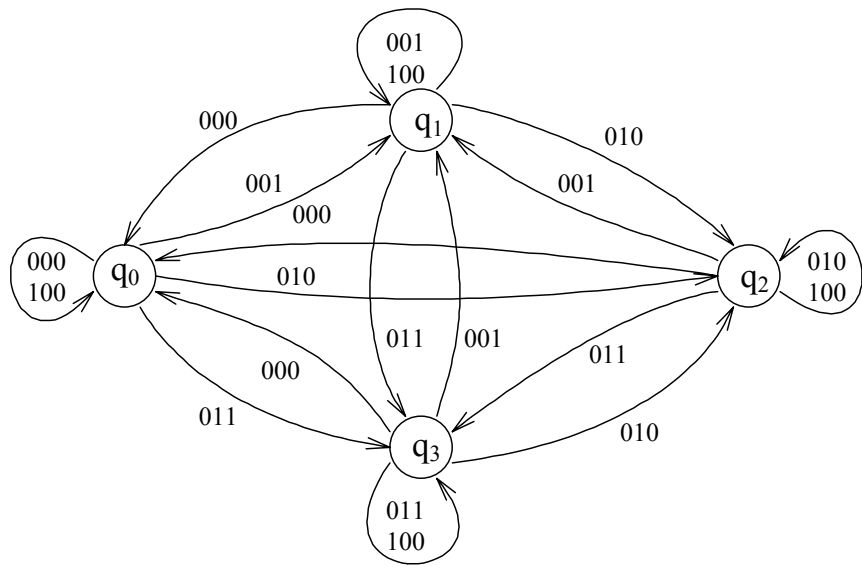


Рис. 1.11. Граф переходов к таблице 1

В таблице 1 три последние набора не имеют выходного значения, т. е. выходы не определены.

Рассмотрим последовательно входные наборы.

1) $x_1x_2x_3 = 000$. Этому набору соответствует начальное состояние q_0 , которое не должно изменяться, что отображается дугой, выходящей и входящей в q_0 .

2) $x_1x_2x_3 = 001$. Состояние устройства меняется на q_1 . Значит, проводится дуга от состояния q_0 к состоянию q_1 и помечается набором 001.

3) $x_1x_2x_3 = 010$. Новое состояние обозначим q_2 , для чего соединим q_0 и q_2 дугой и обозначим ее набором 010.

4) $x_1x_2x_3 = 011$. Переход в состояние q_3 , которое также отличается от всех предыдущих. Соединим q_0 и q_3 дугой, отмеченной набором 011.

5) $x_1x_2x_3 = 100$. В таблице этому набору соответствуют два набора: либо 111, либо 000, что помечено звездочкой (*). Для определенности можно уточнить, что выбран набор 000, т. е. возврат в начальное состояние. Значит, надо отметить дугу, входящую и выходящую из q_0 еще набором 100.

Аналогичным образом поступаем с остальными внутренними состояниями q_1 , q_2 , q_3 , в результате чего получаем окончательный граф переходов для рассматриваемого автомата.

На основании графа автомата можно составить таблицу переходов или таблицу выходов (табл. 2).

Состояние автомата, вершина графа для которого имеет только исходящие дуги, но не имеет входящих дуг, называется **переходящим**. В такое состояние попасть нельзя, из него можно только выйти. Состояние автомата называется **тупиковым**, если соответствующая вершина графа не содержит исходящих дуг, но имеет хотя бы одну входящую дугу.

Таблица 2

Состояния	Входы				
	000	001	010	011	100
q_0	q_0	q_1	q_2	q_3	q_0
q_1	q_0	q_1	q_2	q_3	q_1
q_2	q_0	q_1	q_2	q_3	q_2
q_3	q_0	q_1	q_2	q_3	q_3

Контрольные вопросы

1. Основные понятия теории автоматов.
2. Основные понятия теории формальных грамматик.
3. Классификация языков по Хомскому.
4. Концепция порождения и распознавания.
5. Распознающие и порождающие формальные грамматики.
6. Таблицы переходов и выходов.
7. Граф автомата.
8. Матрицы переходов и выходов.
9. Понятие об информации и ее преобразованиях.
10. Преобразование алфавитной информации.
11. Способы задания автоматов.

2. МАШИНЫ ТЬЮРИНГА

2.1. Основные понятия

Машины Тьюринга представляют собой абстрактные устройства самого общего типа и являются обобщением автоматов, рассмотренных ранее. Машины Тьюринга наиболее близки к реальным ЭВМ, т. к. они представляют собой хорошую математическую модель вычислительной машины. Как показали многочисленные теоретические исследования, классам языков, соответствующим четырем типам грамматики по классификации Хомского, можно поставить во взаимно-однозначное соответствие четыре типа распознающих устройств. Простейшим из них является класс так называемых конечных автоматов, которые допускают (распознают) все языки, порождаемые автоматными (регулярными) граммами, и только их.

Введем понятие детерминированного конечного автомата.

Детерминированным конечным автоматом называют следующую пятерку:

$$A = (X, Q, \delta, q_0, F),$$

где $X = \{x_1, \dots, x_m\}$ – входной алфавит (конечное множество входных сигналов);

$Q = \{q_0, q_1, \dots, q_{n-1}\}$ – алфавит состояний автомата (конечное множество символов);

δ – функция переходов;

$q_0 \in Q$ – начальное состояние автомата;

$F \subseteq Q$ – множество состояний, называемых заключительными.

На содержательном уровне функционирование конечного автомата можно представить следующим образом. Имеется бесконечная лента с ячейками, в каждой

из которых может находиться один символ из X . На ленте находится цепочка символов $\alpha \in X^*$. Ячейки слева и справа от цепочки не заполнены. Имеется некоторое конечное управляющее устройство с читающей головкой, которое может последовательно считывать символы с ленты, передвигаясь слева направо. При этом устройство может находиться в каком-либо одном состоянии из Q . Каждый раз, переходя к новой ячейке, устройство переходит к новому состоянию в соответствии с функцией δ .

На рис. 2.1 изображен конечный автомат в начальном состоянии q_0 , считывающий первый символ x_{i_1} , входной цепочки α_i . Стрелкой показано направление движения читающей головки. Отображение δ можно представить в виде совокупности так называемых команд, которые обозначаются следующим образом:

$$(q, x) \rightarrow q',$$

где $q, q' \in Q$; $x \in X$.

Число команд конечно, левая часть команды (q, x) называется ситуацией автомата, а правая q' – есть состояние, в котором автомат будет находиться на следующем шаге своей работы.

Графически команду удобно представлять в виде дуги графа, идущей из вершины q в вершину q' и помеченную символом x входного алфавита (рис. 2.2).

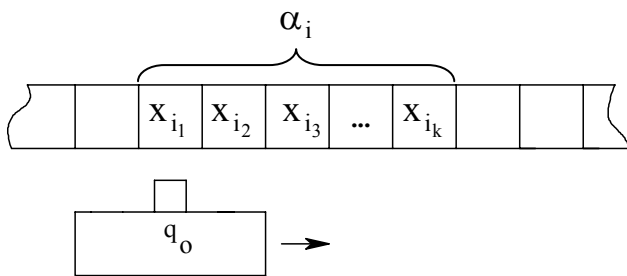


Рис. 2.1. Интерпретация конечного автомата

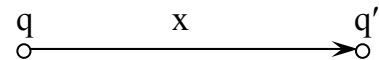


Рис. 2.2. Графическое представление команды автомата

Полностью отображение δ изображают с помощью диаграммы состояний, т. е. ориентированного графа, вершинам которого поставлены в соответствие символы Q , а дугам – команды отображения δ .

Если автомат оказывается в ситуации (q_j, x_i) , не являющейся левой частью какой-либо его команды, то он останавливается. Если управляющее устройство считывает все символы цепочки α , записанной на ленту, и при этом перейдет в состояние $q_f \in F$ (заключительное состояние), то говорят, что цепочка α **допускается автоматом A** (автомат допускает цепочку α). Множество цепочек, допускаемых данным автоматом, называют **языком** этого автомата.

Отображение δ можно представить и в виде функции:

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

где $q, q' \in Q$; $x \in X$.

Эта функция интерпретируется так же, как и команда $(q, x) \rightarrow q'$. Ее можно распространить с одного входного символа на цепочку следующим образом:

$$\delta(q, \varepsilon) = q, \text{ где } \varepsilon - \text{пустая цепочка};$$

$$\delta(q, \alpha x) = \delta(\delta(q, \alpha), x), \text{ где } x \in X, \alpha \in X^*.$$

Таким образом, можно сказать, что α допускается автоматом A , если

$$\delta(q_0, \alpha) = q_f, \text{ где } q_f \in F,$$

а язык, допускаемый автоматом A , это

$$L(A) = \{\alpha \mid \delta(q_0, \alpha) \in F\}.$$

Рассмотрим пример детерминированного конечного автомата

$$A = (X, Q, \delta, q_0, F),$$

где $X = \{a, b\}$; $Q = \{S, Y, Z, T\}$; $q_0 = S$; $F = \{T\}$, а δ задается диаграммой состояний, представленной на рис. 2.3.

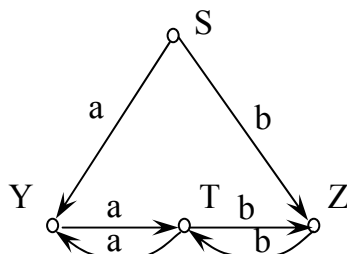


Рис. 2.3. Диаграмма состояний детерминированного конечного автомата

Очевидно, что язык, допускаемый этим автоматом,

$$L(A) = \{M^n \mid n \geq 1\},$$

где $M = \{aa, bb\}$.

Цепочка $\alpha_1 = aabbaa$, допускается данным автоматом, так как после ее просмотра автомат окажется в состоянии $T \in F$.

Цепочка $aabba$ не допускается, так как после ее просмотра автомат окажется в состоянии Y , не являющемся заключительным.

Цепочка abb не допускается потому, что после считывания символа a автомат окажется в ситуации (Y, b) , для которой нет команды.

Недетерминированный конечный автомат — есть пятерка того же вида. Единственное отличие заключается в том, что значениями функции переходов являются не состояния, а множество состояний (или, в терминах команд, возможны различные команды с одинаковыми левыми частями. Это соответствует тому факту, что в диаграмме состояний из одной вершины может исходить несколько дуг с одинаковой пометкой.

2.2. Машины Тьюринга с двумя выходами

С точки зрения лингвистики машины Тьюринга можно рассматривать как распознающие устройства, допускающие языки самого широкого из рассмотренных классов: языки типа 0 или рекурсивно-перечислимые множества.

Машина Тьюринга состоит из конечного управляющего устройства, входной ленты и головки, которая в отличие от головки конечного автомата может не только считывать символы с ленты, но и записывать на нее новые символы. Лента считается бесконечной. Перед началом работы n ячеек ленты содержат символы входной цепочки $\alpha_i = x_{i_1}, x_{i_2}, \dots, x_{i_n}$, все остальные ячейки считаются заполненными специальным символом B («пустое место»), который не является входным (рис. 2.4).

Формально машина Тьюринга определяется как следующая шестёрка:

$$T = (V_1, V_2, Q, \delta, q_0, F),$$

где $V_1 = \{a_1, \dots, a_n\}$ — входной алфавит (конечное множество символов);

$V_2 = \{A_1, \dots, A_k, B\}$ – конечное множество ленточных символов, которое в качестве своего подмножества содержит входной алфавит;

$Q = \{q_0, q_1, \dots, q_{n-1}\}$ – конечное множество состояний;

$q_0 \in Q$ – начальное состояние;

$F \subseteq Q$ – множество заключительных состояний;

δ – функция, отображающая $Q \times V_2$ в $Q \times (V_2 - \{B\}) \times \{Л, П\}$.

(Л и П – специальные символы, указывающие на направление движения головки).

Отображение (функцию) δ удобно задавать совокупностью команд вида:

$(q, A) \rightarrow (q', A', Л)$ либо $(q, A) \rightarrow (q', A', П)$.

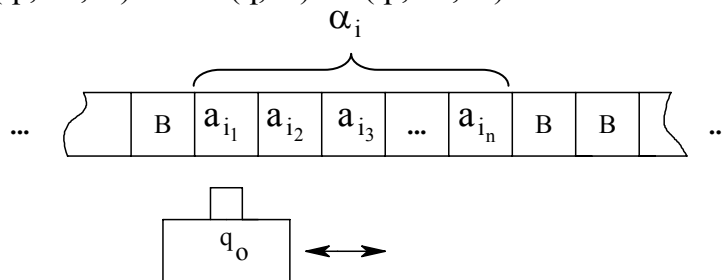


Рис. 2.4. Интерпретация машины Тьюринга

Ситуация машины Тьюринга T – это тройка вида (q, β, i) , где $q \in Q$;

$\beta = A_1, \dots, A_n$ – часть ленты, не содержащая символов B (непустая часть ленты);

$i = (0 \leq i \leq n+1)$ – расстояние ленточной (пишущей - читающей) головки от левого конца β ; при $i = 0$ головка находится левее самого левого символа β , при $i = n+1$ – правее самого правого.

Рассмотрим произвольную ситуацию машины T :

$$(q, A_1 \dots A_i \dots A_n, i), 1 \leq i \leq n.$$

Пусть среди команд отображения δ имеется следующая:

$$(q, A_i) \rightarrow (q', X, Л).$$

При этом возможно следующее движение (или элементарное действие) машины Тьюринга: головка стирает символ A_i , записывает вместо него символ X и перемещается на одну ячейку влево.

Между старой и вновь возникшей ситуациями в этом случае существует отношение, которое записывается следующим образом:

$$(q, A_1 \dots A_i \dots A_n, i) \vdash (q', A_1 \dots A_{i-1} X A_{i+1} \dots A_n, i-1).$$

Символ \vdash означает – «утверждается».

Аналогично для команды $(q, A_i) \rightarrow (q', X, П)$ движение машины Тьюринга записывается как

$$(q, A_1 \dots A_i \dots A_n, i) \vdash (q', A_1 \dots A_{i-1} X A_{i+1} \dots A_n, i+1).$$

Кроме рассмотренной ситуации возможны и такие:

$$(q, A_1 \dots A_n, 0);$$

$$(q, A_1 \dots A_n, n+1).$$

К ним применимы команды вида $(q, B) \rightarrow (q', X, Л)$ либо $(q, B) \rightarrow (q', X, П)$.

Первая из этих команд меняет указанные ситуации соответственно следующим образом:

$$(q, A_1 \dots A_n, 0) \vdash (q', X A_1 \dots A_n, 0);$$

$$(q, A_1 \dots A_n, n+1) \vdash (q', A_1 \dots A_n X, n).$$

Вторая из этих команд меняет их так:

$$(q, A_1 \dots A_n, 0) \vdash (q', X A_1 \dots A_n, 2);$$

$$(q, A_1 \dots A_n, n+1) \vdash (q', A_1 \dots A_n X, n+2).$$

Если ситуации (q_1, β_1, i_1) и (q_2, β_2, i_2) связаны между собой некоторым числом элементарных действий, то между ними имеет место отношение:

$$(q_1, \beta_1, i_1) \vdash^* (q_2, \beta_2, i_2).$$

Язык, допускаемый машиной Тьюринга T , это:

$$L(T) = \{\alpha \mid \alpha \in V_1^* \wedge (q_0, \alpha, 1) \vdash^* (q_f, \beta, i)\},$$

где $q_f = F$, $\beta \in V_2^*$, $i \geq 0$.

Другими словами, если, преобразуя входную цепочку α , машина T окажется в одном из своих заключительных состояний, эта цепочка допускается данной машиной.

Так же как для автоматов введем понятие недетерминированной машины Тьюринга. Ее отличие от детерминированной заключается в том, что функция δ отображает множество $Q \times V_2$ в множество подмножеств $Q \times (V_2 - \{B\}) \times \{L, P\}$.

Если язык L порождается грамматикой типа 0, то L допускается некоторой машиной Тьюринга. Верно и обратное, если язык L допускается некоторой машиной Тьюринга, то L порождается грамматикой типа 0.

2.3. Машины Тьюринга и линейно-ограниченные автоматы

Рассмотренные типы автоматов и машин Тьюринга часто используются для построения автоматного-лингвистических моделей, предназначенных для распознавания языков. Необходимо знать, разрешима ли для них так называемая проблема распознавания или нет.

Эта проблема заключается в следующем. Пусть есть некоторая цепочка α на входе машины Тьюринга, которая допускает язык L . Всегда ли можно установить принадлежность цепочки α к языку L за конечное число элементарных действий этой машины?

Однако не для всех языков типа 0 эта проблема разрешима. Другими словами, можно подобрать такой язык типа 0, что соответствующая ему машина Тьюринга для некоторой цепочки α за конечное число элементарных действий не сможет установить принадлежность ее к данному языку. Поэтому машина Тьюринга в общем виде не нашла применения в реальных кибернетических моделях; языки типа 0 также не используются. Наибольший интерес представляют различные специальные классы машин Тьюринга, к которым можно отнести автоматы, рассмотренные выше, а также так называемые линейно-ограниченные автоматы, допускающие языки типа 1 (НС-языки).

Линейно-ограниченным автоматом называется шестерка:

$$M = (V_1, V_2, Q, \delta, q_0, F),$$

где $V_1 = \{a_1, \dots, a_m, Z_\ell, Z_p\}$ – конечный входной алфавит;

$V_2 = \{A_1, \dots, A_k\}$ – конечное множество ленточных символов, причем $V_1 \leq V_2$;

$Q = \{q_0, q_1, \dots, q_{n-1}\}$ – конечное множество состояний;

$q_0 \in Q$ – начальное состояние;

$F \subseteq Q$ – множество заключительных состояний;

δ – функция, отображающая $Q \times V_2$ в множество подмножеств $Q \times V_2 \times \{L, P\}$.

Множество V_1 содержит два специальных символа Z_ℓ и Z_p , называемых граничными маркерами, которые не позволяют головке управляющего устройства уйти с той части ленты, на которой задана входная цепочка.

Таким образом, линейно-ограниченный автомат является недетерминированной машиной Тьюринга с ограничением на длину цепочки рассматриваемых символов. Ситуация линейно-ограниченного автомата и элементарное действие определяются так же, как и для машины Тьюринга. Язык, допускаемый линейно-ограниченным автоматом, определяется как множество:

$$L(M) = \{\alpha \mid \alpha \in (V_1 - \{Z_\ell, Z_p\})^* \wedge (q_0, Z_\ell \alpha Z_p, 1) \vdash^* (q_f, \beta, i)\},$$

где $q_f = F$, $\beta \in V_2^*$, $1 \leq i \leq n$ (n – длина исходной цепочки).

В настоящее время неизвестно, является ли класс языков, допускаемых детерминированными линейно-ограниченными автоматами, собственным подклассом класса языков, допускаемых недетерминированными линейно-ограниченными автоматами. Однако, для недетерминированных линейно-ограниченных автоматов установлено, что допускаемые ими языки являются НС-языками. Более того, доказано, что для любого НС-языка можно построить недетерминированный линейно-ограниченный автомат.

2.4. Автоматы с магазинной памятью и бесконтекстные языки

Автоматы и преобразователи с магазинной памятью играют важную роль при построении автоматного-лингвистических моделей различного назначения, связанных с использованием бесконтекстных (контекстно-свободных) языков. В частности, такие устройства используются в большинстве работающих программ для синтаксического анализа программ, написанных на различных языках программирования, которые во многих случаях можно рассматривать как бесконтекстные.

2.4.1. Автоматы с магазинной памятью

В отличие от конечных автоматов, рассмотренных ранее, автоматы и преобразователи с магазинной памятью снабжены дополнительной магазинной памятью (рабочей лентой) (рис. 2.5).

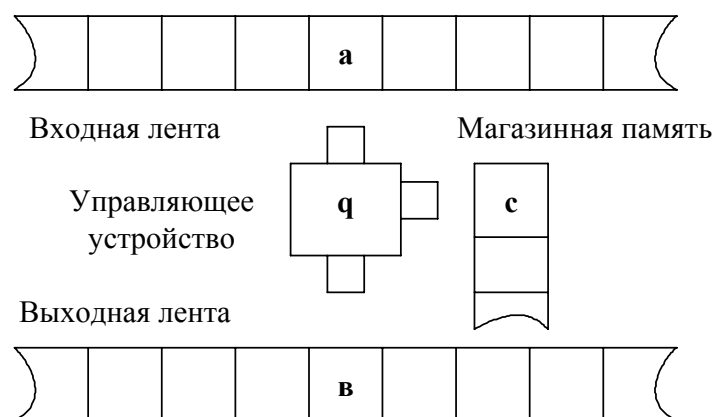


Рис. 2.5. Автоматы с магазинной памятью

Конечное управляющее устройство (УУ) снабжается дополнительной управляющей головкой, всегда указывающей на верхнюю ячейку магазинной памяти: за один такт работы автомата управляющая головка может произвести следующие движения:

- 1) стереть символ из верхней ячейки (при этом все символы, находящиеся на рабочей ленте, перемещаются на одну ячейку вверх);
- 2) стереть символ из верхней ячейки и записать на рабочую ленту непустую цепочку символов (при этом содержимое рабочей ленты сдвигается вниз ровно настолько, какова длина записываемой цепочки).

Таким образом, устройство магазинной памяти можно сравнить с устройством магазина боевого автомата; когда в него вкладывается патрон, те, которые уже были внутри, проталкиваются вниз; достать можно только патрон, вложенный последним.

Формально **детерминированный магазинный автомат** определяется как следующая совокупность объектов:

$$M = (V, Q, V_m, \delta, q_0, z_0, F),$$

где $V, Q, q_0 \in Q, F$ – определяются также, как и для конечного автомата;

$V_m = \{z_0, z_1, \dots, z_{p-1}\}$ – алфавит магазинных символов автомата;

δ – функция, отображающая множество $Q \times (V \cup \{\varepsilon\}) \times V_m$ в множество $Q \times V_m$, где ε – пустая цепочка;

$z_0 \in V_m$ – так называемый граничный маркер, т. е. символ, первым появляющийся в магазинной памяти.

Недетерминированный магазинный автомат отличается от детерминированного только тем, что значениями функции переходов являются не состояния, а множества состояний, т. е. функция δ отображает множество $Q \times (V \cup \{\varepsilon\}) \times V_m$ в множество конечных подмножеств $Q \times V_m$.

Как и в случае конечных автоматов, преобразователи с магазинной памятью отличаются от автоматов с магазинной памятью наличием выходной ленты. Далее будем рассматривать только недетерминированные магазинные автоматы.

Рассмотрим интерпретацию функции δ для такого автомата. Эту функцию можно представить совокупностью команд вида:

$$(q, a, z) \rightarrow (q_1, \gamma_1), \dots, (q_m, \gamma_m),$$

где $q, q_1, \dots, q_m \in Q, a \in V, z \in V_m, \gamma_1, \dots, \gamma_m \in V_m^*$.

При этом считается, что если на входе читающей головки автомата находится символ a , автомат находится в состоянии q , а верхний символ рабочей ленты z , то автомат может перейти к состоянию q_i , если записать на рабочую ленту цепочку γ_i ($1 \leq i \leq m$) вместо символа z и передвинуть входную головку на один символ вправо.

Крайний левый символ γ_i должен при этом оказаться в верхней ячейке магазина. Команда $(q, \varepsilon, z) \rightarrow (q_1, \gamma_1), \dots, (q_m, \gamma_m)$ означает, что независимо от входного символа i , не передвигая входной головки, автомат перейдет в состояние q_i , заменив символ z магазина на цепочку γ_i ($1 \leq i \leq m$).

Ситуацией магазинного автомата называется пара (q, γ) , где $q \in Q, \gamma \in V_m^*$.

Между ситуациями магазинного автомата (q, γ) и (q', γ') устанавливается отношение, обозначаемое символом \vdash , если среди команд найдется такая, что $(q, \varepsilon, z) \rightarrow$

$\rightarrow (q_1, \gamma_1), \dots, (q_m, \gamma_m)$, причем $\gamma = z \beta$, $\gamma' = \gamma_i \beta$, $q' = q_i$ для некоторого $1 \leq i \leq m$ ($z \in V_M$, $\beta \in V_M^*$).

Говорят, что магазинный автомат переходит из состояния (q, γ) в состояние (q', γ') и обозначают это следующим образом:

$$a : (q, \gamma) \vdash (q', \gamma').$$

Вводится и такое обозначение:

$a_1 \dots a_n : (q, \gamma) \vdash^* (q', \gamma')$, если справедливо, что:

$a_i : (q_i, \gamma_i) \vdash (q_{i+1}, \gamma_{i+1})$, $1 \leq i \leq n$,

где $a_i \in V$, $\gamma_i = \gamma_1, \gamma_2, \dots, \gamma_{n+1} = \gamma' \in V_M^*$; $q_i = q_1, \dots, q_{n+1} = q' \in Q$.

2.4.2. Бесконтекстные (контекстно-свободные) языки

Существует два способа определения языка, допускаемого магазинным автоматом. Согласно первому способу считается, что входная цепочка $\alpha \in V^*$ принадлежит языку $L_1(M)$ тогда, когда после просмотра последнего символа, входящего в эту цепочку, в магазине автомата M будет находиться пустая цепочка ε . То есть:

$$L_1(M) = \{ \alpha \mid \alpha : (q_0, z_0) \vdash^* (q, \varepsilon) \}, \text{ где } q \in Q.$$

Согласно второму способу, считается, что входная цепочка принадлежит языку $L_2(M)$ тогда, когда после просмотра последнего символа, входящего в эту цепочку, автомат M окажется в одном из своих заключительных состояний. Другими словами,

$$L_2(M) = \{ \alpha \mid \alpha : (q_0, z_0) \vdash^* (q_f, \gamma) \}, \text{ где } \gamma \in V_M^*, q_f \in F.$$

Доказано, что множество языков, допускаемых произвольными магазинными автоматами согласно первому способу, совпадает с множеством языков, допускаемых согласно второму способу.

Доказано также, что если $L(G_2)$ – бесконтекстный язык, порождаемый грамматикой $G_2 = (V_N, V_T, P, S)$, являющейся формой произвольной бесконтекстной грамматики G , то существует недетерминированный магазинный автомат M , такой, что $L_1(M) = L(G_2)$. При этом

$$M = (V, Q, V_M, \delta, q_0, z_0, 0),$$

где $V = V_T$; $Q = \{q_0\}$; $V_M = V_N$; $z_0 = S$, а для каждого правила G_2 вида $A \rightarrow a \alpha$, $a \in V_T$, $\alpha \in V_N^*$ строится команда отображения δ :

$$(q_0, a, A) \rightarrow (q_0, \alpha).$$

Аналогично, для любого недетерминированного магазинного автомата M , допускающего язык $L_1(M)$, можно построить бесконтекстную грамматику G такую, что $L(G) = L_1(M)$.

Если для конечных автоматов детерминированные и недетерминированные модели эквивалентны соответствующему классу допускаемых языков, то этого нельзя сказать в отношении магазинных автоматов.

Детерминированные автоматы с магазинной памятью допускают лишь некоторое подмножество бесконтекстных языков, которые называют детерминированными бесконтекстными языками.

Контрольные вопросы

1. Машины Тьюринга. Основные понятия.
2. Детерминированный конечный автомат.
3. Машина Тьюринга с двумя выходами.
4. Машины Тьюринга и линейно-ограниченные автоматы.
5. Автоматы с магазинной памятью.
6. Недетерминированный магазинный автомат.
7. Бесконтекстные языки.

3. АБСТРАКТНЫЙ КОНЕЧНЫЙ АВТОМАТ

3.1. Абстрактная теория автоматов

Объектом изучения в абстрактной теории автоматов являются абстрактные автоматы вместе с реализуемыми ими отображениями и событиями. Определим понятия изоморфности, эквивалентности, однозначности функций переходов и выходов. При синтезе цифровых автоматов будем употреблять такие обобщения понятия абстрактного автомата, для которых не выполняются условия полной определенности (частичные автоматы).

3.1.1. Модель дискретного преобразователя Глушкова В. М.

Дискретный преобразователь представляет собой абстрактный автомат A , функционирующий по соответствующим законам в дискретном времени.

Абстрактный автомат A задается совокупностью шести объектов:

$$A = (X, Y, Q, q_0, \delta, \lambda),$$

где X – конечное множество входных сигналов, называемое входным алфавитом автомата;

Y – конечное множество выходных сигналов, называемое выходным алфавитом автомата;

Q – произвольное множество, называемое множеством состояний автомата;

q_0 – элемент из множества Q , называемый начальным состоянием автомата;

$\delta(q, x)$ и $\lambda(q, x)$ – две функции, задающие однозначные отображения множества пар (q, x) , где $q \in Q$ и $x \in X$, в множества Q и Y . Функция $\delta(q, x)$ называется функцией переходов автомата, а функция $\lambda(q, x)$ – функцией выходов, либо сдвинутой функцией выходов.

Автомат, заданный функцией выходов, называется *автоматом первого рода*; автомат, заданный сдвинутой функцией выходов, – *автоматом второго рода*.

Абстрактный автомат функционирует в дискретном времени, принимающем целые неотрицательные значения $t = 0, 1, 2, \dots$. В каждый момент t этого времени он находится в определенном состоянии $q(t)$ из множества Q состояний автомата, причем в начальный момент времени $t = 0$ автомат всегда находится в своем начальном состоянии q_0 , т. е. $q(0) = q_0$.

В момент времени t , отличный от начального, автомат способен воспринимать входной сигнал $x(t)$ – произвольную букву входного алфавита X и выдавать соответствующий выходной сигнал $y(t)$ – некоторую букву выходного алфавита Y .

Закон функционирования абстрактного *автомата первого рода* задается уравнением:

$$\begin{cases} q(t) = d(q(t-1), x(t)), \\ y(t) = l(q(t-1), x(t)), \text{ где } t = 1, 2, \dots, \end{cases}$$

а абстрактного *автомата второго рода* – уравнением:

$$\begin{cases} q(t) = \delta(q(t-1), x(t)), \\ y(t) = \lambda(q(t), x(t)), \text{ где } t = 1, 2, \dots \end{cases}$$

Вывод: Таким образом, в абстрактной теории автоматов входные и выходные сигналы рассматриваются как буквы (символы) двух фиксированных для данного автомата алфавитов. Абстрактная теория изучает те переходы, которые претерпевает автомат под воздействием входных сигналов в дискретные моменты времени, и те выходные сигналы, которые он при этом выдает.

3.1.2. Понятие об абстрактном автомате и индуцируемом им отображении

Сущность индуцируемых абстрактным автоматом отображений заключается в реализации некоторого отображения φ из множества слов входного алфавита в множество слов выходного алфавита.

Отображение φ реализуется следующим образом: каждое слово $p = x_{i1}, x_{i2}, \dots, x_{ik}$ входного алфавита $X = (x_1, x_2, \dots, x_n)$ или, более кратко, каждое *входное слово*, последовательно, буква за буквой, подается на вход данного абстрактного автомата A , установленного предварительно в начальное состояние. Возникающая таким образом конечная последовательность входных сигналов $x(1) = x_{i1}, x(2) = x_{i2}, \dots, x(k) = x_{ik}$ на основании *закона функционирования автомата* вызывает появление однозначно определенной конечной последовательности $s = y(1), y(2), \dots, y(k)$ выходных сигналов. Эту последовательность будем называть *выходным словом*, соответствующим входному слову p .

Относя, таким образом, каждому входному слову p соответствующее ему выходное слово s , мы получаем искомое отображение φ , а именно, $s = \varphi(p)$. Построенное указанным способом отображение φ будем называть *отображением, индуцированным абстрактным автоматом A* .

Два абстрактных автомата с общим входным и выходным алфавитами называются *эквивалентными между собой*, если они индуцируют одно и то же отображение множества слов входного алфавита в множество слов выходного алфавита.

Наряду с эквивалентностью важнейшее значение имеет понятие *изоморфизма абстрактных автоматов*.

Предположим, что заданы два абстрактных автомата:

$$A_1(X_1, Y_1, Q_1, q_1, \delta_1(q, x), \lambda_1(q, x)) \quad \text{и} \\ A_2(X_2, Y_2, Q_2, q_2, \delta_2(q, x), \lambda_2(q, x))$$

одного и того же рода. Если для этих автоматов существуют взаимно однозначные отображения: α – отображающее множество X_1 на множество X_2 ; β – отображающее

множество Y_1 на множество Y_2 ; γ – отображающее множество Q_1 на множество Q_2 ; и, если удовлетворяются условия:

$$\begin{aligned}\gamma(q_1) &= q_2; \\ \gamma(\delta_1(q_1, x)) &= \delta_2(\gamma(q), \alpha(x)); \\ \beta(\lambda_1(q, x)) &= \lambda_2(\gamma(q), \alpha(x)) \text{ (для любых } q \in Q_1 \text{ и } x \in X_1),\end{aligned}$$

то абстрактные автоматы A_1 и A_2 называются **изоморфными**. В этом случае говорят, что отображения α , β и γ осуществляют **изоморфное отображение** одного автомата на другой.

Изоморфные между собой абстрактные автоматы отличаются друг от друга лишь обозначениями входных и выходных сигналов и состояний. Поэтому, в абстрактной теории автоматов, не занимаются проблемами кодирования состояний, а также входных и выходных сигналов, изоморфные автоматы считаются одинаковыми и будут заменяться один другим без каких-либо дальнейших пояснений. Операция перехода от данного абстрактного автомата к изоморфному ему автомату состоит просто в **переобозначении** элементов входного алфавита, выходного алфавита и множества состояний автомата.

Способ сведения автоматов второго рода к автоматам первого рода, рассмотренный ранее, мы условимся называть **интерпретацией** автомата второго рода как автомата первого рода. Обратная интерпретация автомата первого рода как автомата второго рода при сохранении того же самого множества состояний оказывается, вообще говоря, невозможной.

Произвольные автоматы первого рода, как отмечено ранее, называются **автоматами Мили**, а частный случай автоматов второго рода, у которых сдвинутая функция выходов $\lambda(q, x)$ не зависит от второй переменной x – **автоматами Мура**.

Способы задания автоматов были рассмотрены ранее в п. 1.5.3. Здесь отметим одну из особенностей задания автоматов. Так, если произвольным образом заполненная таблица переходов или таблица выходов представляют собой таблицу переходов или соответствующую таблицу выходов некоторого абстрактного автомата, то не всякий граф и не всякая автоматная матрица могут служить графом или матрицей переходов некоторого автомата. Для того, чтобы это имело место, необходимо соблюдение некоторых *условий*, связанных со свойствами функций переходов автомата.

Первое условие связано с однозначностью функций переходов и выходов и называется – **условием однозначности**.

Соблюдение этого условия требует, чтобы на графе автомата из любой вершины выходило бы не более одной стрелки, обозначенной конкретным входным сигналом.

Применительно к матрице это условие означает, что в любой ее строке конкретный входной сигнал должен встречаться не более одного раза.

Второе условие, называется **условием определенности**, требующим, чтобы для любого входного сигнала x из каждой вершины обязательно выходила бы стрелка, обозначенная этим входным сигналом, и чтобы этот входной сигнал обязательно присутствовал в каждой строке автоматной матрицы.

Частичным автоматом называется абстрактный автомат, у которого функция переходов или функция выходов (обычная или сдвинутая), или обе эти функции определены не для всех пар значений своих аргументов q и x . В отличие от частичных, ранее рассмотренные автоматы назывались *вполне определенными*.

3.2. Представление событий в автоматах

Рассмотрим задачи, возникающие на этапе абстрактного анализа и синтеза автоматов. Рассмотрим условия, при которых можно индуцировать любое алфавитное отображение в автоматное.

Условия автоматности накладывают жесткие ограничения на класс отображений, которые могут быть индуцированы абстрактными автоматами. Большинство алфавитных отображений, с которыми приходится иметь дело на практике, не удовлетворяют этим условиям. Существует, однако, стандартный прием, позволяющий индуцировать любое алфавитное отображение в автоматное. К описанию этого приема мы и переходим.

3.2.1. Автоматные отображения и события

Отображения, индуцируемые абстрактными автоматами, будем называть **автоматными отображениями**.

Всякое автоматное отображение удовлетворяет следующим четырем условиям:

1. Автоматное отображение осуществляет однозначное отображение (как правило, частичное) множества слов в некотором конечном алфавите X (входное алфавитное отображение) в множества слов в некотором конечном алфавите Y (выходное алфавитное отображение).

2. Область определения автоматного отображения удовлетворяет условию полноты, т. е. вместе с любым содержащимся в ней словом также содержатся все начальные отрезки этого слова. Пустое слово всегда входит в область определения отображения.

3. Автоматное отображение φ сохраняет длину слова. Любое слово p входного алфавита на котором отображение φ определено, имеет ту же длину, что и его образ $\varphi(p)$. В частности, пустое слово переводится отображением φ в пустое слово.

4. Автоматное отображение φ переводит любой начальный отрезок слова p , на котором оно определено, в соответствующий (имеющий ту же длину) начальный отрезок слова $\varphi(p)$.

Перечисленные условия называются **условиями автоматности отображения**.

Все слова входного алфавита разбиваются автоматным отображением на два класса: на класс допустимых и класс запрещенных слов. Совокупность всех запрещенных слов для данного автоматного отображения будет называться его **областью запрета**.

Рассмотрим произвольное (частичное) отображение φ , для которого выполняются сформулированные выше условия. Построим абстрактный (частичный) автомат Мура U , состояниями которого будут служить всевозможные допустимые для отображения φ слова входного алфавита $X = (x_1, \dots, x_n)$. Обозначим множество всех таких слов через A и определим функцию переходов $\delta(a, x)$ этого автомата следующим образом: если a_j – любое слово из A , а x_i – произвольная буква входного алфавита, то будем считать, что $\delta(a_j, x_i)$ равняется слову $a_j x_i$ (получающемуся в результате приписывания буквы x_i к слову a_j), если слово $a_j x_i$ содержится в A , и что $\delta(a_j, x_i)$ не определена в противном случае.

Выбирая в качестве выходного алфавита автомата U выходной алфавит отображения φ , построим сдвинутую функцию выходов $\lambda(a)$ автомата Мура U следующим образом: для любого непустого слова a_i из A полагаем $\lambda(a_i)$ равным последней букве слова $\varphi(a_i)$; на пустом слове функция $\lambda(a)$ остается неопределенной.

В качестве начального состояния автомата выбираем пустое слово ε и получаем автомат Мура, индуцирующий исходное отображение φ . В самом деле, пусть, $q = x_{i1}, x_{i2}, \dots, x_{ik}$ – любое допустимое слово отображения φ . Тогда все его начальные отрезки будут также допустимыми словами (в силу условия 2). После подачи на вход построенного автомата слова q , будет осуществляться последовательный его перевод в состояние ε $x_{i1} = x_{i1}, x_{i2}, \dots, x_{ik}$.

При этом автомат выдает некоторую последовательность выходных букв $y_{j1}, y_{j2}, \dots, y_{jk} = p$. Из способа определения данного автомата следует, что последняя буква слова p совпадает с последней буквой слова $\varphi(q)$, предпоследняя буква слова p – с последней буквой слова $\varphi(x_{i1}, x_{i2}, \dots, x_{ik-1})$, которая в силу условия 4, совпадает с предпоследней буквой слова $\varphi(q)$ и т. д. Продолжая сравнение и используя условия 3, можно установить совпадение всех букв слова p с соответствующими им буквами слова $\varphi(q)$. Следовательно, построенный автомат Мура U индуцирует исходное (частичное) отображение φ .

Поскольку можно интерпретировать автомат Мура U как автомат Мили, то очевидно следующее предложение.

Если алфавитное отображение φ удовлетворяет сформулированным выше четырем условиям автоматности, то можно построить автоматы Мили и Мура (как правило, бесконечные), индуцирующие это отображение. В случае, когда область определения отображения φ конечна, эти автоматы также можно считать конечными.

Данное предложение позволяет применять термины «автоматное отображение» по всему алфавитному отображению, удовлетворяющему условиям автоматности.

Из этого предложения вытекает конструктивный прием, позволяющий по любому автоматному отображению **с конечной областью определения** (заданному на конечном множестве слов) строить индуцирующий это отображение конечный автомат Мили или Мура.

Ранее рассматривалась возможность интерпретации автомата Мура как автомата Мили с тем же самым числом состояний. Рассмотрим теорему:

Для всякого конечного автомата Мили существует эквивалентный ему (индуцирующий то же самое отображение) конечный автомат Мура. Существует единственный конструктивный прием (универсальный алгоритм преобразования автоматов Мили в автоматы Мура), позволяющий по произвольному конечному автомату Мили, имеющему m входных сигналов и n состояний, построить эквивалентный ему автомат Мура, имеющий не более $m \cdot n + 1$ состояний.

Условия автоматности накладывают весьма жесткие ограничения на класс отображений, которые могут быть индуцированы абстрактными автоматами. Большинство алфавитных отображений, с которыми приходится иметь дело на практике, в частности большинство алгоритмов, не удовлетворяют этим условиям. Существует, однако, стандартный прием, позволяющий индуцировать любое алфавитное отображение в автоматное.

Рассмотрим этот прием.

Пусть φ – произвольное (как правило, частичное) отображение множества слов в (конечном) алфавите X в множество слов в (конечном) алфавите Y . Обозначим через P область определения этого отображения. Будем применять к отображению φ две операции.

Первая операция – выравнивания длин слов (операция выравнивания).

Ее суть: во входной и выходной алфавиты отображения φ (т. е. X и Y) добавляется по одной новой букве, которые будем называть пустыми и обозначать через α и β .

Каждому слову p из P приписывается справа m_p экземпляров букв α , а к его образу $q = \varphi(p)$ приписывается слева n_q экземпляров букв β так, чтобы длины полученных в результате приписывания новых букв словам p_1 и q_1 совпали.

Далее строится новое отображение φ_1 некоторого множества P_1 , слов в алфавите $X \cup \{\alpha\}$ в множества слов в алфавите $Y \cup \{\beta\}$, которое переводит друг в друга слова p_1 и q_1 , полученные в результате выравнивания длин слов p и q соответственно: $\varphi_1(p_1) = q_1$ (p – пробегает при этом все множество P). Условимся считать, что отображение φ_1 , получается из отображения φ с помощью операции выравнивания. Существует некоторая стандартная операция выравнивания, при которой отображение φ_1 , однозначно определяется отображением φ и присоединенными пустыми буквами α и β . Суть ее в том, что число m_p пустых букв α , приписываемых справа к произвольному слову p из области определения отображения φ , принимается равным длине слова $q = \varphi(p)$, а число n_q пустых букв β , приписываемых слева к слову q , принимается равным длине слова p .

Вторая операция применяется только к выравниванию алфавитных отображений φ , т. е. к таким отображениям у которых длины входных и соответствующих им выходных слов равны между собой. Сущность этой операции – *операции пополнения* отображения φ , заключается в распространении отображения φ на начальные отрезки слов.

Если s – произвольный начальный отрезок любого слова p из области определения отображения φ , то полагаем $\varphi(s)$ равным начальному отрезку слова $\varphi(p)$, т. е. имеющему длину s .

В результате применения операции пополнения к произвольному выровненному алфавитному отображению φ возникает новое отображение φ_1 область определения которого удовлетворяет условию полноты. Условимся называть это отображение *пополнением отображения φ* .

Если пополнение φ_1 отображения φ является однозначным, то очевидно, что оно удовлетворяет всем четырем условиям автоматности.

3.2.2. Представление событий в автоматах

Основной задачей абстрактной теории автоматов является установление связей, существующих между автоматами и индуцируемыми ими отображениями. Произвольное автоматное отображение можно задавать событиями, которые соответствуют этим отображениям.

Рассмотрим как устанавливаются связи между событиями и автоматами. Для этого используем следующие определения:

1. Для произвольного автомата A множество S_y всех входных слов, вызывающих появление выходных слов, которые оканчиваются одной и той же буквой (вы-

ходным сигналом) y , называется **событием**, представленным в автомате A выходным сигналом y . Множество, состоящее из событий S_y , для всех букв выходного алфавита автомата A , называется **каноническим множеством** событий данного автомата A .

2. Каноническое множество событий любого автомата является автоматным множеством событий. Очевидно и обратное, что для любого автоматного множества событий M существует такой автомат (в качестве которого можно выбрать как автомат Мили, так и автомат Мура), каноническое множество событий которого совпадает с множеством M .

В отличие от автоматного отображения абстрактный **автомат не определяется однозначно** соответствующим ему каноническим множеством событий, поскольку одно и то же автоматное отображение может индуцироваться различными автоматами.

Из рассмотренных определений возникает необходимость решения следующих важнейших задач абстрактной теории автоматов:

- задачу нахождения по заданному абстрактному автомату A соответствующего ему канонического множества событий – каноническая задача анализа абстрактного автомата;
- обратную задачу, по заданному автоматному множеству событий M найти абстрактный автомат, каноническое множество событий которого совпадает с M – каноническая задача синтеза абстрактного автомата.

3.2.3. Регулярные языки и конечные автоматы

Как отмечалось в п. 3.2.2, любое автоматное отображение φ может быть задано конечным множеством M событий во входном алфавите этого отображения. Если область определения отображения φ конечна, то все события множества M конечны. Конечное событие можно задать, перечислив все его элементы. Ясно, что в случае, когда область определения отображения φ бесконечна, хотя бы одно из событий множества M также будет бесконечным. Поэтому необходимо разработать специальный язык, который позволял бы представлять (описывать) бесконечные события, соответствующими конечными выражениями.

Рассмотрим язык регулярных выражений алгебры событий. Для этого используем три операции над событиями:

- 1) $A \cup B$ – объединение (дизъюнкция);
- 2) $A \cdot B$ – умножение (конкатенация);
- 3) $\{A\}$ – итерация (обозначается также A^*).

Определим конечные множества входных букв $X = \{x_1, \dots, x_n\}$ и зададим для этого множества регулярное выражение.

Регулярным выражением в алфавите X называется выражение, построенное из букв алфавита X и из символов операций объединения, умножения и итерации с использованием соответствующим образом круглых скобок. Всякое регулярное выражение определяет некоторое событие S (S получается в результате выполнения всех операций, входящих в регулярное выражение). События, определяемые таким образом, называются регулярными событиями над алфавитом X . Другими словами **регулярным событием (или языком)** называется событие, полученное из элементарных событий (однобуквенных слов x_i) применением конечного числа раз операций дизъюнкции, конкатенации и итерации. Например, в алфавите из трех букв x, y, z регуляр-

ное выражение $x \{x \vee y \vee z\} (y \vee z)$ задает регулярное событие, состоящее из всех слов, которые начинаются буквой x и заканчиваются буквой y или z . Для конечных автоматов характерны только регулярные события.

Таким образом, общая задача анализа абстрактного автомата ставится как задача нахождения по заданному абстрактному автомату такого события, которое представлено любым множеством его состояний. Аналогично, общая задача синтеза абстрактного автомата ставится как задача построения по любому конечному множеству M событий такого автомата, который представляет каждое событие этого множества некоторым множеством своих выходных сигналов.

3.3. Алгоритм синтеза конечных автоматов

Рассмотренный выше стандартный прием позволяет свести общую задачу синтеза автоматов к канонической задаче. При решении этой задачи необходимо, с одной стороны, фиксировать некоторый язык, приспособленный для конструктивного описания событий, а с другой стороны – ограничиться вполне определенным классом автоматов, допускающих какой-либо конструктивный способ задания. Поэтому ограничимся лишь конечными автоматами и регулярными событиями. Оба эти класса объектов допускают конструктивное описание: первый – на языке таблиц переходов и выходов, а второй – на языке регулярных выражений алгебры событий. Такое ограничение вполне достаточно для практических целей, потому что цифровые автоматы, с которыми приходится иметь дело на практике, всегда конечны.

3.3.1. Основной алгоритм синтеза конечных автоматов

Пусть требуется построить алгоритм, который позволял бы по любому конечному множеству M регулярных событий, заданных своими регулярными выражениями, находить таблицу переходов и выходов конечного вполне определенного автомата Мили и отмеченную таблицу переходов конечного вполне определенного автомата Мура A таких, что все события множества M представляются как в автомате Мили, так и в автомате Мура некоторыми множествами их выходных сигналов.

Перейдем от представления событий R_1, \dots, R_p в автомате Мура множествами состояний к представлению их множествами выходных сигналов. Для этого достаточно в качестве выходных сигналов взять различные подмножества заданного множества событий (R_1, \dots, R_p) и отмечать каждое состояние q автомата множеством тех событий, которые содержат слова, переводящие автомат из начального состояния в состояние q .

Если состояния, не представляют ни одного из заданных событий, то они отмечаются пустым множеством событий. Такой способ отметок состояний автомата Мура называется *каноническим способом отметок*.

Если исходные для алгоритма регулярные выражения R_1, \dots, R_p заданные регулярные события являются многочленами, то они заключаются в обычные (неитерационные) скобки. Это условие будет называться *условием правильной записи регулярных выражений*.

Местами в правильно записанном регулярном выражении R над алфавитом $X = (x_1, \dots, x_n)$ условимся называть специально вводимые знаки раздела (вертикальные линии), ставящиеся между любыми двумя знаками этого выражения (знаками в

выражении R являются буквы алфавита X , символ пустого слова ε , знак дизъюнкции, итерационные и обычные скобки).

Кроме этих так называемых *разделяющих мест* (знаков раздела) вводится еще два места – *начальное* и *конечное*. Первое из них располагается слева от самого левого знака выражения R , а второе – справа от самого правого знака.

Например, $R = z \vee x \{y \vee z\}$, где R – регулярное выражение.

Приведем это выражение к правильной форме

$$R = (z \vee x \{y \vee z\}).$$

Проведем разметку

$$\begin{array}{cccccccccccc} | & (& | & z & | & \vee & | & x & | & \{ & | & y & | & \vee & | & z & | & \} & | &) & | \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \end{array} \quad (1)$$

Из выражения (1) видно, что регулярное выражение R имеет 11 мест.

Развернем данное регулярное выражение в слово, т. е. последовательно, буква за буквой, выпишем какое-либо слово из представляемого им события. При этом развертывание, будем осуществлять переходя от одного места данного регулярного выражения к другому и от начального места к конечному месту. Такие переходы бывают двух видов – *непосредственный переход* и *переход через букву основного алфавита X* , в котором задается представляемое выражением событие.

В качестве примера рассмотрим процессы развертывания размеченного выше выражения R в принадлежащие представляемому им событию слова z и xuz .

При развертывании выражения в первое слово необходимо осуществить непосредственный переход от начального места 1 к месту 2, переход через букву z от места 2 к месту 3 и непосредственный переход от 3 к конечному месту 11.

При развертывании выражения в слово xuz порядок переходов будет следующий: непосредственный переход от места 1 к месту 4 переход через букву x – от 4 к 5, непосредственный переход – от 5 к 6, переход через букву y – от 6 к 7, непосредственный переход от 7 к 10, непосредственный переход от 10 к 8, переход через букву z – от 8 к 9, непосредственный переход от 9 к 10 и непосредственный переход от 10 к 11.

Пусть R – регулярное выражение, $q = x_{i1} x_{i2} \dots x_{ik}$ произвольное слово во входном алфавите события, представляемого выражением R .

Условимся, что место α в выражении R *связано* словом q с местом β в том же выражении, если от места α к месту β можно перейти с помощью чередования любого числа непосредственных переходов и переходов через буквы $x_{i1}, x_{i2}, \dots, x_{ik}$ слова q , взятых по одному разу в том порядке, в каком они входят в слово. Это условие означает, что место β *q-следует* за местом α всякий раз, когда α связано с местом β словом q .

Также условимся говорить, что место β *подчинено месту α* , если от места α к месту β можно перейти с помощью одних лишь непосредственных переходов, т. е. если место α связано с местом β пустым словом ε .

Правила подчинения мест в регулярных выражениях

1. Начальные места всех термов многочлена, помещенного в обычные или итерационные скобки, подчинены месту, находящемуся непосредственно слева от открывающей скобки из пары скобок, в которые заключен данный многочлен (многочлен по этому правилу может вырождаться в одночлен с обязательным заключением его в скобки).

2. Место, расположенное непосредственно справа от закрывающей (итерационной или обыкновенной) скобки подчинено конечным местам всех термов многочлена (в частном случае, одночлена), заключенного в соответствующие скобки, а в случае итерационных скобок – также месту, расположенному непосредственно слева от соответствующей открывающей скобки.

3. Начальные места всех термов многочлена (в частном случае, одночлена), заключенного в итерационные скобки, подчинены месту, расположенному непосредственно справа от соответствующей закрывающей скобки.

4. Место, расположенное непосредственно справа от символа пустого слова ϵ , подчинено месту, расположенному непосредственно слева от этого символа.

5. Если место γ подчинено месту β , а место β подчинено месту α , то место γ подчинено месту α (свойства транзитивности для отношения подчиненности мест).

6. Каждое место подчинено самому себе.

В качестве примера использования приведенной системы правил установим отношение подчиненности мест для рассмотренного выше регулярного выражения (1). Место 1 кроме самого себя не подчинено никаким другим местам. То же самое относится к местам 3, 5, 7, 9.

Места 2, 4 кроме самих себя, подчинены месту 1 (по правилу 1).

Место 6 и 8 – месту 5 (по правилу 1) и месту 10 (по правилу 3).

Место 10 подчинено (кроме самого себя) местам 5, 7 и 9 (по правилу 2).

Место 11 подчинено (кроме самого себя) местам 3 и 10 (по правилу 2) и местам 5, 7 и 9 (по правилу 5).

Основными местами в таких выражениях считаются по определению все места, непосредственно слева от которых стоит буква основного алфавита, а также начальное место.

Все места, непосредственно *справа* от которых стоит буква основного алфавита, называются **предосновными**.

Правила построения основного алгоритма синтеза конечных автоматов

1. Заданные регулярные события (число которых предполагается конечным) представляются правильно записанными регулярными выражениями R_1, \dots, R_p . Все места (как основные, так и не основные) в этих выражениях обозначаются вертикальными черточками (линиями).

2. Каждому основному месту в выражениях R_1, \dots, R_p приписывается в качестве индекса неотрицательное целое число. При этом всем начальным местам приписывается один и тот же индекс (нуль). Что же касается остальных основных мест, то они нумеруются в произвольном порядке натуральными числами 1, 2, ...

Все введенные здесь индексы называются основными. Основные индексы подписываются под вертикальными чертами соответствующих им (основных) мест и подчеркиваются снизу общей для каждого из выражений R_1, \dots, R_p горизонтальной разделительной чертой.

3. Индекс (основной) каждого основного места α распространяется в качестве неосновного индекса на все места (как основные, так и неосновные), подчиненные месту α , но отличные от него самого.

При этом каждое место β получает некоторое множество неосновных индексов. Все индексы этого множества подписываются в произвольном порядке под вертикальной чертой, соответствующей месту β , ниже разделительной горизонтальной черты.

Все индексы (как основные, так и неосновные), относящиеся к любому предосновному месту, заключаются в общую рамку.

4. Строится таблица переходов некоторого автомата Мура A . В качестве состояний этого автомата берутся подмножества множества всех основных индексов. При этом подмножество, состоящее из основных индексов i_1, \dots, i_k ($k \geq 1$), будем обозначать через $i_1 \vee i_2 \vee \dots \vee i_k$, а пустое множество основных индексов – звездочкой (соответствующее ему состояние автомата A называется пустым).

Построение таблицы переходов с состояниями автомата A начинается с вписывания обозначений строк и столбцов таблицы.

Строки таблицы обозначаются (в произвольном порядке) различными буквами входного алфавита заданного множества событий.

Столбцы обозначаются состояниями автомата A , начиная с нулевого. Последующий столбец обозначается в произвольном порядке состоянием из предыдущего столбца после вписывания всех состояний в этот столбец.

На пересечении произвольной (x_i -й) строки и произвольного (q_j -го) столбца таблицы вписываются состояния (множества основных индексов), состоящие из основных индексов всех тех и только тех основных мест, которые x_i -следуют за предосновными местами, в числе индексов которых (как основных, так и не основных) находится хотя бы один индекс, принадлежащий состоянию q_j . В случае отсутствия основных мест с требуемыми свойствами на соответствующем месте таблицы вписывается пустое состояние.

5. Каждое из состояний $i_1 \vee i_2 \vee \dots \vee i_k$ ($k \geq 1$), обозначающее столбцы таблицы переходов, отмечается множеством (R_{j1}, \dots, R_{jm}) всех символов тех и только тех регулярных выражений R_1, \dots, R_p , конечные места которых содержат в числе своих индексов (как основных, так не основных) хотя бы один из индексов i_1, \dots, i_k .

Пустое состояние отмечается пустым множеством регулярных выражений R_1, \dots, R_p и обозначается через $()$. С помощью введенных отметок, принимаемых за выходной алфавит, строится отмеченная таблица переходов искомого конечного автомата Мура A .

6. В случае необходимости найденный автомат Мура A интерпретируется как автомат Мили B . Таблица переходов автомата A при этом принимается за таблицу переходов B . Таблица выходов автомата B получается в результате подстановки в его таблицу переходов вместо символов состояний символов, соответствующих состояниям выходных сигналов (отметок) автомата A .

Построенные автоматы A и B представляют заданные события множествами своих состояний и (с точностью до пустого слова) множествами своих выходных сигналов.

Событие, заданное регулярным выражением R_i , представляется множеством всех тех и только тех состояний, которые отмечены множествами, содержащими в качестве своих элементов выражение R_i . Это же событие (за вычетом лишь пустого слова, если оно содержится в событии) представляется в автоматах A и B множеством всех тех и только тех выходных сигналов (множеств), состоящих из выражений R_1, \dots, R_p , которые содержат в своем составе выражение R_i ($i = 1, \dots, p$). Множество со-

стояний, отмеченных пустым множеством (), или выходной сигнал () представляет событие, состоящее из всех слов входного алфавита, не вошедших в заданные события.

В процессе синтеза автомата или по окончании этого процесса производят переобозначения состояний и выходных сигналов с целью упрощения записи таблиц переходов и выходов. Обычно при переобозначении состояний их просто нумеруют натуральными числами 1, 2, ..., используя для обозначения начального состояния единицу. В некоторых случаях оказывается целесообразным обозначать состояния числами 0, 1, 2, ..., тогда начальное состояние обозначается всегда нулем.

Рассмотрим пример синтеза конечного автомата в соответствии с описанным алгоритмом.

Пусть требуется построить конечный автомат, в котором для входного алфавита X , состоящего из двух букв x и y , были бы представлены два события: событие R_1 , состоящее из всех слов в алфавите X , в которых все буквы x предшествуют всем буквам y , и событие R_2 , состоящее из всех слов в алфавите X , которые кончаются буквой x .

Применяя правило 1, записываем заданные события в виде следующих регулярных выражений:

$$R_1 = \{x\}^* \{y\}^*, \quad R_2 = \{x \vee y\}^* x.$$

После разметки мест эти выражения приобретут вид:

$$R_1' = |\{x\}^* \{y\}^*|, \quad R_2' = |\{x \vee y\}^* x|.$$

Применяем правило 2, в результате получим выражения:

$$R_1'' = |\{x\}^* \{y\}^*|, \quad R_2'' = |\{x \vee y\}^* x|.$$

0	1	2
---	---	---

0	3	4	5
---	---	---	---

В результате применения правила 3 получаем:

$$R_1''' = |\{x\}^* \{y\}^*|, \quad R_2''' = |\{x \vee y\}^* x|.$$

0	1	2
0	0	0
1	1	1
	2	2

0	3	4	5
0	0	0	
3	3	3	
4	4	4	

Применение правила 4 приводит к построению таблицы переходов 3.1.

Применение правила 5 дает отмеченную таблицу переходов 3.2.

Обозначая выходные сигналы (), (R_1), (R_2), (R_1, R_2) соответственно через z , u , v и w и, нумеруя состояния, переходим к отмеченной таблице переходов 3.3.

При интерпретации построенного автомата как автомата Мили в соответствии с правилом 6 мы получим таблицу его выходов 3. 4.

Таблица 3.1

	0	$1 \vee 3 \vee 5$	$2 \vee 4$	$3 \vee 5$	4
x	$1 \vee 3 \vee 5$	$1 \vee 3 \vee 5$	$3 \vee 5$	$3 \vee 5$	$3 \vee 5$
y	$2 \vee 4$	$2 \vee 4$	$2 \vee 4$	4	4

Таблица 3.2

	(R_1)	(R_1, R_2)	(R_1)	(R_2)	$()$
	0	$1 \vee 3 \vee 5$	$2 \vee 4$	$3 \vee 5$	
x	$1 \vee 3 \vee 5$	$1 \vee 3 \vee 5$	$3 \vee 5$	$3 \vee 5$	$3 \vee 5$
y	$2 \vee 4$	$2 \vee 4$	$2 \vee 4$	4	4

Таблица 3.3

	u	w	u	v	z
	1	2	3	4	5
x	2	2	4	4	4
y	3	3	3	5	5

Таблица 3.4

	1	2	3	4	5
x	w	w	v	v	v
y	u	u	u	z	z

Построенные автоматы представляют событие R_1 состояниями 1, 2, 3, а событие R_2 – состояниями 2, 4. Событие R_1 за вычетом содержащегося в нем пустого слова представляется также выходными сигналами u и w, а событие R_2 – выходными сигналами v, w. Состоянием 2 или, что то же самое, выходным сигналом w представлено пересечение событий R_1 и R_2 .

3.3.2. Усовершенствованный основной алгоритм синтеза конечных автоматов

Рассмотренный выше алгоритм синтеза конечного автомата допускает ряд уточнений и изменений, позволяющих упрощать синтезируемый автомат.

Первое уточнение преследует цель сократить количество используемых основных индексов и уменьшить, по возможности, число состояний в синтезируемом автомате.

Рассмотрим понятие **комплекс регулярных выражений K** , под которым понимается любое конечное множество регулярных выражений R_1, \dots, R_p в одном и том же конечном алфавите (называемом входным алфавитом комплекса), у которых размечены и снабжены индексами все основные места. При этом различные основные места в выражениях R_1, \dots, R_p могут иметь один и тот же основной индекс. Всякое множество, состоящее только из основных мест всех выражений R_1, \dots, R_p , имеющих один и тот же основной индекс k , называется **основным местом комплекса K** , при этом индекс k называется **основным индексом этого места**. Начальные места всех выражений R_1, \dots, R_p всегда снабжаются одинаковым индексом 0 и составляют **начальное место комплекса K** .

Если некоторое основное место α комплекса K состоит из основных мест $\alpha_1, \alpha_2, \dots, \alpha_k$ в выражениях R_1, \dots, R_p , составляющих комплекс, то местами, **подчиненными месту α в комплексе K** , считаются все места в выражениях R_1, \dots, R_p , подчиненные хотя бы одному из мест $\alpha_1, \dots, \alpha_k$ в этих выражениях.

Будем считать, что основное место в комплексе K x_i -*следует* за предосновным местом β , если хотя бы одно из составляющих место α основных мест в выражениях R_1, \dots, R_p x_i -*следует* за местом β .

Слово q входного алфавита *связывает* основное место комплекса K с основным местом β того же комплекса, если оно связывает некоторое основное место в

одном из выражений R_1, \dots, R_p , входящее в место α , с каким-либо основным местом того же самого выражения, входящим в место β . При этом место β *q-следует* за местом α .

Основные места в данном комплексе K регулярных выражений называются **соответственными**, если множества слов входного алфавита, связывающих начальное место комплекса с каждым из этих мест, одинаковы.

Основные места $\alpha_1, \dots, \alpha_k$ в комплексе K называются **подобными**, если в комплексе K им подчинены одинаковые множества конечных мест и если для любой буквы x входного алфавита комплекса множества M_1, \dots, M_k основных мест, *x-следующих* за местами $\alpha_1, \dots, \alpha_k$, одинаковы, либо становятся одинаковыми при замене входящих в них мест $\alpha_2, \dots, \alpha_k$ местом α_1 .

Операция отождествления мест

Ее сущность заключается в том, что некоторому множеству основных мест данного комплекса K , имеющих различные основные индексы, приписывается один и тот же индекс k и, следовательно, все основные места в регулярных выражениях комплекса, входившие в отождествляемые места комплекса K , составят новое основное место комплекса, которое будет иметь основной индекс k .

Правила 1 и 2 основного алгоритма синтеза конечных автоматов применимы и к заданному множеству регулярных событий некоторого комплекса K регулярных выражений. В этом комплексе каждое основное место, за исключением начального места, состоит из единственного основного места в выражениях R_1, \dots, R_p , составляющих комплекс. Таким образом, в комплексе K_0 не проводится никакого отождествления мест, за исключением отождествления начальных мест, входящих в комплекс выражений.

Все последующие правила (т. е. правила 3 – 6) основного алгоритма синтеза автоматов оперируют с местами и с их индексами в этом комплексе K_0 , который будем называть **нулевым**. На практике оказывается возможным до перехода к правилам 3 – 6 применять к нулевому комплексу **операцию отождествления мест** так, чтобы последующее применение правил 3 – 6 к новому комплексу привело к построению автоматов, представляющих исходные события R_1, \dots, R_p . Для этого правила 1 – 6 основного алгоритма синтеза конечных автоматов могут быть дополнены следующими правилами.

Правило 2а. К комплексу K_0 заданных регулярных выражений R_1, \dots, R_p , полученному по правилам 1 – 2, можно применять последовательно, шаг за шагом, операцию отождествления соответственных мест и операцию отождествления подобных мест. Последующие правила 3 – 6 применяются не к нулевому комплексу K_0 , а к комплексу, полученному после выполнения любого числа таких отождествлений.

При практическом применении этого правила необязательно проводить все возможные отождествления, достаточно ограничиться лишь некоторыми из них. Возникающие таким образом алгоритмы (с тем или иным числом предварительных отождествлений мест) будем называть **усовершенствованными алгоритмами синтеза конечных автоматов**.

Если стоит задача синтеза автомата Мили, то в общий алгоритм синтеза могут быть внесены изменения, приводящие к уменьшению числа состояний синтезируемого автомата. В данном случае нет необходимости отмечать состояния автомата в соответствии с правилом 5 и затем применять правило 6: их последовательное приме-

нение можно заменить применением одного правила 5а, относящегося к построению таблицы выходов синтезируемого автомата Мили.

Правило 5а. Таблица выходов автомата Мили имеет те же обозначения строк и столбцов, что и построенная по правилу 4 таблица его переходов. На пересечении произвольной строки, обозначенной буквой входного алфавита x_i , и произвольного столбца, обозначенного состоянием $i_1 \vee i_2 \vee \dots \vee i_k$ ($k \geq 1$), в таблице выходов ставится выходной сигнал (R_{j1}, \dots, R_{jm}) , состоящий из символов всех тех регулярных выражений R_1, \dots, R_p , конечные места которых x_i -следуют хотя бы за одним из мест i_1, i_2, \dots, i_k . В столбце, обозначенном пустым состоянием, во всех строках ставится пустой выходной сигнал $()$.

Алгоритм, состоящий из правил 1, 2, 3, 4, 5а, будем называть **особым алгоритмом синтеза конечных автоматов Мили**. Этот алгоритм допускает отождествление соответственных мест в нулевом комплексе исходных регулярных выражений, полученном по правилам 1 и 2 основного алгоритма. Что же касается подобных мест, то, ввиду отсутствия в рассматриваемом случае отметок состояний, их необходимо заменить так называемыми **квазиподобными местами**, определяемыми следующим образом.

Основные места $\alpha_1, \dots, \alpha_k$ любого заданного комплекса регулярных выражений называются квазиподобными, если для любой буквы x входного алфавита комплекса K множество конечных мест комплекса, x -следующих за основными местами $\alpha_1, \dots, \alpha_k$, совпадают между собой, а множества M_1, \dots, M_k основных мест комплекса, x -следующих за местами $\alpha_1, \dots, \alpha_k$ либо одинаковы, либо становятся одинаковыми после замены входящих в множества M_1, \dots, M_k мест $\alpha_2, \dots, \alpha_k$ местом α_1 .

Сформулируем правило отождествления мест в случае синтеза автоматов Мили.

Правило 2б. При синтезе автоматов Мили к нулевому комплексу K_0 , полученному по правилам 1–2 основного алгоритма синтеза, можно применять последовательно, шаг за шагом, но в любом порядке операции отождествления соответственных и квазиподобных мест. К полученному в результате таких отождествлений комплексу применяются правила 3, 4, 5а особого алгоритма синтеза конечных автоматов Мили.

Согласно приведенному правилу, квазиподобными являются все **тупиковые** места комплекса, т. е. такие его основные места, которые не связаны ни одной буквой входного алфавита ни с одним местом комплекса. Если в синтезированном автомате Мили имеются состояния, образуемые основными индексами лишь одних тупиковых мест, то в таблице переходов автомата столбцы, обозначенные всеми такими состояниями, будут состоять из одних символов пустого состояния $*$, а соответствующие столбцы в таблице выходов – из одних лишь символов пустого выходного сигнала. При синтезе автоматов Мили наряду с правилом 2б может применяться следующее правило (2в).

Правило 2в. При синтезе автоматов Мили тупиковым основным местам в комплексе регулярных выражений можно не присваивать никаких основных индексов.

Алгоритмы синтеза, использующие правила 1, 2, 2б, 2в, 3, 4, 5а, будем называть **усовершенствованными особыми алгоритмами синтеза конечных автоматов Мили**.

Усовершенствованные алгоритмы синтеза автоматов Мили, в отличие от основного алгоритма, не приводят к автоматам, в которых заданные события представлены множеством состояний, а лишь к таким автоматам, в которых эти события пред-

ставлены множеством выходных сигналов. Продемонстрируем работу усовершенствованных алгоритмов синтеза на примере.

Пример. Найти конечные автоматы Мура и Мили, представляющие регулярные события в алфавите (x, y) , заданные следующими регулярными выражениями:

$$R = x \{y\}, \quad P = x x.$$

Выпишем нулевой комплекс K_0 заданных регулярных выражений:

$$R = \begin{array}{|c|c|c|} \hline x & \{ & y & \} \\ \hline 0 & 1 & 2 & \\ \hline \end{array}, \quad P = \begin{array}{|c|c|c|} \hline x & x & \\ \hline 0 & 3 & 4 & \\ \hline \end{array}$$

В этом комплексе места 1 и 3 являются соответственными между собой, места 1 и 2 – подобными, а место 4 – тупиковым.

Проведем последовательные отождествления мест в соответствии с правилом 2а. Отождествляя сначала соответственные места, придем к комплексу:

$$R = \begin{array}{|c|c|c|} \hline x & \{ & y & \} \\ \hline 0 & 1 & 2 & \\ \hline \end{array}, \quad P = \begin{array}{|c|c|c|} \hline x & x & \\ \hline 0 & 1 & 4 & \\ \hline \end{array}$$

После такого отождествления места 1 и 2 перестают быть подобными (за местом 1 x – следует место 4, а за местом 2 – нет). Дальнейшие отождествления по правилу 2а невозможны.

Применяя теперь правило 3, получим размеченный комплекс:

$$R = \begin{array}{|c|c|c|} \hline x & \{ & y & \} \\ \hline \boxed{0} & 1 & 2 & \\ \hline \end{array}, \quad P = \begin{array}{|c|c|c|} \hline x & x & \\ \hline \boxed{0} & \boxed{1} & 4 & \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline \end{array} \quad \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline \end{array}$$

По правилам 4 и 5 получаем отмеченную таблицу 3.5 переходов автомата Мура.

После переобозначений:

$0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 4 \rightarrow 4, * \rightarrow 5, (R) \rightarrow u, (P) \rightarrow v, () \rightarrow w$, получаем отмеченную таблицу 3.6 переходов автомата Мура. (Переобозначение производят с целью упрощения записи таблиц переходов и выходов).

Таблица 3.5

	()	(R)	(R)	(P)	()
	0	1	2	4	*
x	1	4	*	*	*
y	*	2	2	*	*

Таблица 3.6

	w	u	u	v	w
	1	2	3	4	5
x	2	4	5	5	5
y	5	3	3	5	5

Событие R представлено в этом автомате выходным сигналом u , а событие P – выходным сигналом v . Применяя правило 6, получим таблицы переходов и выходов автомата Мили (табл. 3.7 и 3.8).

Таблица 3.7

	0	1	2	4	5
x	2	4	5	5	5
y	5	3	3	5	5

Таблица 3.8

	1	2	3	4	5
x	u	v	w	w	w
y	w	u	u	w	w

При использовании основного и усовершенствованного алгоритмов синтеза необходимо использовать правила 5б и 6а.

Правило 5б. По заданной области запрета или области допустимых слов автомата Мура А находятся неопределенные состояния автомата. Все вхождения неопределенных состояний в отмеченную таблицу переходов автомата А заменяются черточками, а обозначенные этими состояниями столбцы исключаются из таблицы. Если к числу неопределенных относится начальное состояние автомата, то обозначенный им (начальный) столбец сохраняется в таблице, но отметка начального состояния делается неопределенной. В случае возникновения недостижимых состояний обозначенные ими столбцы также вычеркиваются.

Правило 6а. По заданной области запрета или области допустимых слов автомата Мили В находятся неопределенные выходные сигналы и неопределенные состояния автомата. Все элементы таблицы выходов, являющиеся неопределенными выходными сигналами, и соответствующие им (стоящие на пересечении тех же строк и столбцов, что и неопределенные выходные сигналы) элементы таблицы переходов – заменяются черточками. Все столбцы (за исключением начального), обозначенные неопределенными состояниями, вычеркиваются из таблиц переходов и выходов автомата. В случае возникновения недостижимых состояний соответствующие им столбцы также вычеркиваются.

При этом запрещенными называют слова (например, пустое слово) под действием которых автомат переходит в неопределенное состояние. Остальные слова называются *допустимыми*.

3.4. Автоматы Мили и Мура

Детерминированные преобразователи дискретной информации, реализующие некоторые заданные алфавитные операторы, называют дискретными (конечными) автоматами. Конечные автоматы, рассматриваемые безотносительно к их внутренней структуре, обычно называют абстрактными конечными автоматами.

Конечные автоматы разделяют на автоматы без памяти и автоматы с памятью. Автоматом с памятью называют автомат, описываемый функциями переходов и выходов, оператор которого является оператором с памятью. Выходные слова автомата с памятью зависят не только от входных слов, но и от последовательности их поступления.

Автомат с памятью имеет множество внутренних состояний, в которое он переходит под воздействием слов входного алфавита. Наличие множества внутренних состояний придает автомату способность запоминания входной информации, поступившей на вход автомата в прошлом. В зависимости от вида функции выходов автоматы с памятью делятся на автоматы Мили и автоматы Мура.

Автоматом Мили называется автомат, выходные слова которого зависят как от внутренних состояний автомата, так и от значений входных слов.

Автоматом Мура называется автомат, выходные слова которого зависят только от внутренних состояний автомата и не зависят непосредственно от входных слов.

3.4.1. Автомат Мили

Широко распространенный на практике класс автоматов Мили получил свое название по имени американского ученого G. H. Mealy, впервые исследовавшего эту модель.

Оператор автомата Мили (автомата с памятью) полностью описывается функцией переходов $q(t+1)$ и функцией выходов $y(t)$.

Закон функционирования автомата Мили задается уравнениями:

$$\begin{aligned} q(t+1) &= \delta(q(t), x(t)), \\ y(t) &= \lambda(q(t), x(t)), \text{ где } t = 0, 1, 2, \dots \end{aligned}$$

Как отмечалось в п. 3.1.1, конечный автомат A описывается выражением:

$$A = (Q, X, Y, \delta, \lambda, q_0),$$

в котором заданы входной и выходной алфавиты, алфавит состояния, а также функции переходов и выходов.

Выделение в множестве состояний конечного автомата A начального состояния q_0 объясняется чисто практическими соображениями, связанными, в первую очередь, с необходимостью фиксировать условия начала работы дискретного устройства. Автомат с выделенным начальным состоянием q_0 называется **инициальным**.

Многие же задачи можно решать, описывая автомат без начального состояния q_0 :

$$A = (Q, X, Y, \delta, \lambda).$$

Функцию переходов δ и функцию выходов λ определим на их множестве пар <состояние – входное слово>.

Пусть

$\xi = x_{i1} x_{i2} \dots x_{ik}$ – входное слово длины k ; E – множество всех конечных входных слов ненулевой длины; ε – входное слово нулевой длины (пустое слово); $\tilde{\delta}(q_m, \varepsilon) = q_m$ для всех $q_m \in Q$.

Тогда функцию **заключительного состояния** $\tilde{\delta}(q_m, \xi)$ определим в множестве $Q \times (E \cup \{\varepsilon\})$ следующим образом:

$$\begin{aligned} \tilde{\delta}(q_m, \xi) &= \tilde{\delta}(q_m, x_{i1} \dots x_{ik}) = \\ &= \begin{cases} \underbrace{\delta(\tilde{\delta}(q_m, x_{i1} \dots x_{ik-1}), x_{ik})}_{q_{ik}} = \delta(q_{ik}, x_{ik}), & \text{если } \delta(q_{ij}, x_{ij}) \text{ определена для всех } j = 1, \dots, k; q_{i1} = q_m; \\ \text{не определена} & \text{в противном случае.} \end{cases} \end{aligned}$$

Пример 1

Пусть автомат A_1 задан совмещенной таблицей 3.9.

Таблица 3.9

	$\delta : Q \times X \rightarrow Q,$		$\lambda : Q \times X \rightarrow Y$	
	q_1	q_2	q_3	q_4
x_1	q_2/y_1	q_3/y_3	q_4/y_3	–
x_2	q_3/y_2	–	$q_2/-$	q_2/y_2

Требуется найти $\tilde{\delta}(q_2, \xi)$, при $\xi = x_1 x_1 x_2 x_1 x_2$.

$$\begin{aligned} \tilde{\delta}(q_2, x_1 x_1 x_2 x_1 x_2) &= \tilde{\delta}(\delta(q_2, x_1), x_1 x_2 x_1 x_2) = \tilde{\delta}(q_3, x_1 x_2 x_1 x_2) = \\ &= \tilde{\delta}(\delta(q_3, x_1), x_2 x_1 x_2) = \tilde{\delta}(q_4, x_2 x_1 x_2) = \tilde{\delta}(\delta(q_4, x_2), x_1 x_2) = \tilde{\delta}(q_2, x_1 x_2) = \\ &= \tilde{\delta}(\delta(q_2, x_1), x_2) = \tilde{\delta}(q_3, x_2) = \delta(q_3, x_2) = q_2. \end{aligned}$$

Если $\xi = x_1 x_1 x_2 x_2 x_2$, то получим:

$$\begin{aligned}\tilde{\delta}(q_2, x_1 x_1 x_2 x_2 x_2) &= \tilde{\delta}(\delta(q_2, x_1), x_1 x_2 x_2 x_2) = \tilde{\delta}(q_3, x_1 x_2 x_2 x_2) = \\ &= \tilde{\delta}(\delta(q_3, x_1), x_2 x_2 x_2) = \tilde{\delta}(q_4, x_2 x_2 x_2) = \tilde{\delta}(\delta(q_4, x_2), x_2 x_2) = \tilde{\delta}(q_2, x_2 x_2) = \\ &= \tilde{\delta}(\delta(q_2, x_2), x_2).\end{aligned}$$

Функция $\tilde{\delta}(q_2, \xi)$ при $\xi = x_1 x_1 x_2 x_2 x_2$ не определена, так как в таблице 3.9 не определена функция переходов $\delta(q_2, x_2)$.

Таким образом, автомат A_1 переходит в заключительное состояние $\tilde{\delta}(q_m, \xi)$ из состояния q_m только под действием определенного входного слова o .

Функция заключительного выхода $\tilde{\lambda}(q_m, o)$ модели автомата Мили определяется в множестве $Q \times (E \cup \{\varepsilon\})$ следующим образом:

$$\tilde{\lambda}(q_m, o) = \begin{cases} \lambda(\tilde{\delta}(q_m, \xi'), x_{ik}), \xi = \xi' x_{ik}, & \text{если } \tilde{\delta}(q_m, \xi') \text{ — определена;} \\ \text{не определена, если не определена } \tilde{\delta}(q_m, \xi'), \end{cases}$$

где $\xi = x_{i1} \dots x_{ik-1} x_{ik}$; $\xi' = x_{i1} \dots x_{ik-1}$.

Таким образом, в модели автомата Мили $\tilde{\lambda}(q_m, o)$ — выходной сигнал, выдаваемый на последнем переходе (переходе в заключительное состояние). Он появляется на выходе в тот же момент времени, когда приходит последняя буква входного слова o .

Пример 2

Пусть автомат A_2 , задан таблицей 3.10, найдем $\tilde{\lambda}(q_1, x_1 x_2 x_1 x_1)$.

Таблица 3.10

$\delta: Q \times X \rightarrow Q;$		$\lambda: X \rightarrow Y$	
	q_1	q_2	q_3
x_1	q_2/y_1	q_3/y_3	q_2/y_3
x_2	q_3/y_2	q_2/y_1	q_1/y_1

$$\begin{aligned}\tilde{\delta}(q_1, x_1 x_2 x_1 x_1) &= \tilde{\delta}(\delta(q_1, x_1), x_2 x_1 x_1) = \tilde{\delta}(q_2, x_2 x_1 x_1) = \\ &= \tilde{\delta}(\delta(q_2, x_2), x_1 x_1) = \tilde{\delta}(q_2, x_1 x_1) = \tilde{\delta}(\delta(q_2, x_1), x_1) = \tilde{\delta}(q_3, x_1) = \delta(q_3, x_1) = q_2.\end{aligned}$$

Последним в этой последовательности переходов является переход (q_3, x_1) , поэтому $\tilde{\lambda}(q_1, x_1 x_2 x_1 x_1) = \lambda(q_3, x_1) = y_3$.

Функцию $\omega(q_m, \xi)$ — **реакцию автомата** в состоянии q_m на входное слово $\xi = x_{i1} \dots x_{ik}$ определим в множестве $Q \times (E \cup \{\varepsilon\})$ следующим образом:

$\omega(q_m, \xi)$ — не определена, если не определена $\tilde{\delta}(q_m, \xi')$,

где $\xi' = x_{i1} \dots x_{ik-1}$, т.е. $\xi = \xi' x_{ik}$.

Иначе, если $\tilde{\delta}(q_m, \xi')$ определена, то

$$\omega(q_m, \xi) = \lambda(q_m, x_{i1}) \omega(\delta(q_m, x_{i1}), x_{i2} \dots x_{ik}) = \lambda(q_m, x_{i1}) \lambda(q_{i2}, x_{i2}) \dots \lambda(q_{ik}, x_{ik}) = y_{i1} \dots y_{ik},$$

где $q_{i2} = \delta(q_m, x_{i1})$; $y_{ij} = \lambda(q_{ij}, x_{ij})$; $j = 2, 3, \dots, k$; $y_{i1} = \lambda(q_m, x_{i1})$.

Пример 3

Для автомата A_2 $\omega = (q_1, x_1 x_2 x_1 x_1) = \lambda(q_1, x_1) \omega(\delta(q_1, x_1), x_2 x_1 x_1) =$
 $= y_1 \omega(q_2, x_2 x_1 x_1) = y_1 \lambda(q_2, x_2) \omega(\delta(q_2, x_2), x_1 x_1) = y_1 y_1 \omega(q_2, x_1 x_1) =$
 $= y_1 y_1 \lambda(q_2, x_1) \omega(\delta(q_2, x_1), x_1) = y_1 y_1 y_3 \lambda(q_3, x_1) = y_1 y_1 y_3 y_3.$

3.4.2. Автомат Мура

Автомат Мура получил название по имени впервые исследовавшего эту модель американского ученого E. F. Moore.

Закон функционирования автомата Мура задается уравнениями:

$$q(t+1) = \delta(q(t), x(t)),$$

$$y(t) = \lambda(q(t)), \quad t = 0, 1, 2, \dots$$

Так как в автомате Мура выходной сигнал зависит только от внутреннего состояния автомата и не зависит непосредственно от входного сигнала, то он задается одной **отмеченной** таблицей переходов, в которой каждому ее столбцу приписан кроме состояния q_m еще и выходной сигнал $y_g = \lambda(q_m)$, соответствующий этому состоянию. Пример табличного описания автомата Мура A_3 иллюстрируется таблицей 3.11.

Таблица 3.11

$\delta: Q \times X \rightarrow Q;$		$\lambda: Q \rightarrow Y$			
	y_1	y_1	y_3	y_2	y_3
	q_1	q_2	q_3	q_4	q_5
x_1	q_2	q_5	q_5	q_3	q_3
x_2	q_4	q_2	q_2	q_1	q_1

Функция заключительного состояния $\tilde{\delta}(q_m, \xi)$ в множестве $Q \times (E \cup \{\varepsilon\})$ определяется также, как и для автомата Мили.

Функция заключительного выхода $\tilde{\lambda}(q_m, o)$ модели автомата Мура определена в множестве $Q \times (E \cup \{\varepsilon\})$ следующим образом:

$$\tilde{\lambda}(q_m, o) = \begin{cases} \lambda(\tilde{\delta}(q_m, o)), & \text{для всех } o \in E, \text{ если функция } \tilde{\delta}(q_m, o) - \text{определена;} \\ \text{не определена, если не определена } \tilde{\delta}(q_m, o). \end{cases}$$

Очевидно, что в модели автомата Мура функция $\tilde{\lambda}(q_m, o)$ представляет собой выходной сигнал, который отмечает заключительное состояние.

Функция $\omega(q_m, \xi)$ – реакция автомата в состоянии q_m на входное слово $\xi = x_i, \dots, x_{ik}$ – не определена, если $\tilde{\delta}(q_m, \xi)$ не определена.

Если $\tilde{\delta}(q_m, \xi)$ определена, то:

$$\omega(q_m, \xi) = \lambda(\delta(q_m, x_{i1})) \omega(\delta(q_m, x_{i1}), x_{i2} \dots x_{ik}) = \lambda(q_{i2}) \lambda(q_{i3}) \dots \lambda(q_{ik+1}) = y_{i2} y_{i3} \dots y_{ik+1},$$

где $q_{i2} = \delta(q_m, x_{i1}); \quad y_{ij} = \lambda(q_{ij}), \quad j = 2, 3, \dots, k+1.$

Пример 4

Определим реакцию $\omega = (q_1, x_2 x_1 x_1 x_2)$ автомата Мура, заданного отмеченной таблицей переходов 3.11.

$$\begin{aligned}\omega(q_1, x_2 x_1 x_1 x_2) &= \lambda(\delta(q_1, x_2))\omega(\delta(q_1, x_2), x_1 x_1 x_2) = \lambda(q_4)\omega(q_4, x_1 x_1 x_2) = \\ &= y_2 \lambda(\delta(q_4, x_1))\omega(\delta(q_4, x_1), x_1 x_2) = y_2 \lambda(q_3)\omega(q_3, x_1 x_2) = y_2 y_3 \lambda(\delta(q_3, x_1))\omega(\delta(q_3, x_1), x_2) = \\ &= y_2 y_3 \lambda(q_5)\omega(q_5, x_2) = y_2 y_3 y_3 \lambda(\delta(q_5, x_2)) = y_2 y_3 y_3 \lambda(q_1) = y_2 y_3 y_3 y_1.\end{aligned}$$

Таким образом, в автомате Мура выходной сигнал $y_{i1} = \lambda(q_m)$ в момент времени i_1 не зависит от входного сигнала x_{i1} , а определяется только состоянием q_m . Этот сигнал y_{i1} никак не связан с входным словом, поступающим на вход автомата, начиная с момента i_1 . В связи с этим под реакцией автомата Мура в состоянии q_m на входное слово $\xi = x_{i1} x_{i2} \dots x_{ik}$ понимается выходное слово той же длины $\omega(q_m, \xi) = y_{i2} y_{i3} \dots y_{i_{k+1}}$, но сдвинутое на один такт автоматного времени.

3.4.3. Получение неполностью определенных (частичных) автоматов

Автомат называется **полностью определенным**, если $D_\delta = D_\lambda = Q \times X$, определена для всех пар $(q_m, x_j) \in Q \times X$,

где D_δ – отображение функции переходов δ ;

D_λ – отображение функции выходов λ .

У полностью определенного автомата области определения функций δ и λ совпадают с множеством $Q \times X$ – множеством всевозможных пар вида (q_m, x_f) .

У **неполностью определенного**, или **частичного**, автомата функции δ или λ определены не для всех пар $(q_m, x_f) \in Q \times X$.

При задании неполностью определенных автоматов табличным способом на месте неопределенных состояний и выходных сигналов ставится прочерк. Для частичного автомата Мили S_3 (табл. 3.9) реакция автомата $\omega(q_2, x_1 x_1 x_2 x_1 x_2) = y_3 y_3 y_2 y_3$. Черточка на последнем месте означает, что в реакции последний выходной сигнал не определен. Сама же реакция определена, так как $\tilde{\delta}(q_2, x_1 x_1 x_2 x_1 x_2)$ определена. Однако, для этого же автомата реакция $\omega(q_2, x_1 x_1 x_2 x_2 x_2)$ будет не определена, так как не определена функция $\tilde{\delta}(q_2, x_1 x_1 x_2 x_2 x_2)$.

Можно определить реакцию автомата Мили в состоянии q_m на входное слово ξ и через заключительный выход:

$$\omega(q_m, \xi) = \tilde{\lambda}(q_m, x_{i1}) \tilde{\lambda}(q_m, x_{i1} x_{i2}) \dots \tilde{\lambda}(q_m, x_{i1} \dots x_{ik}).$$

3.5. Синтез автоматов по индуцируемым ими отображениям

Одной из наиболее важных задач теории абстрактных автоматов является задача синтеза конечных автоматов по индуцируемым ими алфавитным отображениям. Такая задача может решаться так называемым общим методом решения задачи.

3.5.1. Общий метод решения задачи

Абстрактный автомат с входным и выходным алфавитами индуцирует однозначное отображение множества слов во входном алфавите (входном алфавитном

отображении) в множество слов в выходном алфавите (конечном алфавите). Отображения (как правило, частичные), индуцируемые абстрактными автоматами, называются автоматными отображениями.

Рассмотрим этапы решения задачи синтеза автоматов по индуцируемым ими отображениям.

Первый этап решения задачи состоит в том, что исходное (частичное) алфавитное отображение φ записывается в виде таблицы соответствия и приводится к автоматному виду с помощью применения операции выравнивания длин слов (п. 3.2.1). При этом прежде чем использовать стандартную операцию выравнивания длин слов, производится ряд последовательных попыток приведения отображения к автоматному виду приписыванием к словам по возможности меньшего числа пустых букв с проверкой после каждой попытки выполнимости **условий автоматности**. Результатом первого этапа является сокращенная таблица соответствия автоматного отображения ψ , полученная на основе исходного отображения φ после применения операции выравнивания длин слов.

Второй этап решения состоит в нахождении **канонического множества событий**, соответствующего отображению ψ . Число событий канонического множества равно числу букв выходного алфавита $Y = (y_1, \dots, y_m)$ отображения ψ . Событие $S(y_i)$ этого множества, соответствующее выходной букве y_i , строится следующим образом. Просматривая сокращенную таблицу соответствия отображения ψ , выделяют все начальные отрезки выходных слов, оканчивающиеся буквой y_i .

Начальные отрезки входных слов отображения, соответствующие выделенным отрезкам, и составят событие $S(y_i)$ ($i = 1, \dots, m$).

Результатом второго этапа является каноническое множество событий $S(y_1), \dots, S(y_m)$ отображения ψ . На этом этапе также контролируется правильность выполнения операций первого этапа. Если найденные события попарно не пересекаются, то только тогда отображение ψ считается автоматным.

Третий этап решения состоит в нахождении возможно более простых регулярных выражений для найденных на предыдущем этапе событий $S(y_1), \dots, S(y_m)$. При этом используют тождественные преобразования алгебры событий, а также возможность расширения событий **подыскиванием** для события $S(y_i)$ наиболее простого регулярного выражения и добавлением к нему произвольного множества запрещенных слов, т. е. таких слов, которые не содержатся ни в одном из исходных событий $S(y_1), \dots, S(y_m)$. Найденное таким образом регулярное выражение события $S(y_i)$ для простоты будем обозначать через y_i ($i = 1, \dots, m$). Отметим, что события, представленные простыми регулярными выражениями y_1, \dots, y_m , могут пересекаться между собой.

Результатом третьего этапа является множество M всех найденных регулярных выражений y_1, \dots, y_m .

Четвертый этап решения состоит в синтезе конечного автомата Мили или Мура, представляющего события R_1, \dots, R_m , задаваемые регулярными выражениями y_1, \dots, y_m , посредством множеств своих выходных сигналов. Процесс синтеза может производиться по двум основным вариантам.

По первому варианту синтеза с естественной областью запрета считаются запрещенными, *во-первых*, все те слова, у которых хотя бы один непустой начальный отрезок (включая само слово) не содержится ни в одном из событий R_1, \dots, R_m , и, *во-вторых*, все те слова, у которых хотя бы один непустой начальный отрезок (включая само слово) содержится одновременно в нескольких (более чем в одном) собы-

тиях R_1, \dots, R_m . Синтез конечного частичного автомата Мура или Мили, представляющего события R_1, \dots, R_m , производится с учетом введенной области запрета. При этом можно применять любой из описанных алгоритмов синтеза. Выходные сигналы в синтезированных автоматах будут совпадать с буквами y_1, \dots, y_m выходного алфавита отображения ψ , а автоматы будут индуцировать частичное отображение ψ_1 , продолжающее исходное частичное отображение ψ с сохранением входного и выходного алфавитов отображения ψ .

По второму варианту синтеза с исключением одного из событий исключается из заданного множества y_1, \dots, y_m регулярных выражений одно регулярное выражение (обычно наиболее сложное). Пусть этим выражением будет y_1 . К оставшимся выражениям y_2, \dots, y_m применяется любой из алгоритмов синтеза частичных автоматов Мили или Мура. При этом запрещенными считаются все те слова, у которых хотя бы один непустой начальный отрезок (включая само слово) содержится одновременно в нескольких (более чем в одном) событиях R_2, \dots, R_m . Выходные сигналы, содержащие более одного символа y_2, \dots, y_m будут при этом неопределенными. Из оставшихся выходных сигналов y_2, \dots, y_m , () пустой выходной сигнал () заменяется выходным сигналом y_1 . После этого синтезированные автоматы будут индуцировать частичное отображение ψ_2 , продолжающее исходное частичное отображение ψ с сохранением его входного и выходного алфавитов.

Второй вариант применяется с учетом следующих *особенностей*. Этот вариант позволяет относительно просто произвести в таблице выходов автомата (обычной или сдвинутой) дифференциацию пустых выходных сигналов, отождествляя только некоторую часть из них с выходным сигналом y_1 и производя замену остальных таких сигналов черточками. Так, например, можно отождествить все слова исключаемого события $S(y_1)$, совпадающие с начальными отрезками слов остающихся событий $S(y_2), \dots, S(y_m)$.

Очевидно, что в этом случае пустые выходные сигналы, относящиеся к пустому состоянию автомата, не будут представлять слов события $S(y_1)$, и их можно считать неопределенными. Неопределенным будет и пустое состояние.

Описанное уточнение второго варианта сводится к тому, что исключаемое событие $S(y_1)$ расширяется лишь до некоторой части дополнения объединения остальных событий.

3.5.2. Синтез автомата Мили

Применение общего метода решения задачи рассмотрим на примере синтеза автомата Мили по индуцируемым им отображениям.

Пример. Построить автомат Мили, который преобразовывал бы числа от 1 до 9 в четверичной системе счисления, подаваемые последовательно на его вход, начиная со старшего разряда, в приближенные (целочисленные) значения квадратного корня из этих чисел, выдаваемые соответственно на выход также в четверичной системе счисления, начиная со старшего разряда.

Решение. Приближенные значения квадратного корня из чисел от 1 до 9 будут равны соответственно 1, 1, 2, 2, 2, 2, 3, 3, 3. Полученные числа в четверичной системе счисления представим таблицей соответствия отображения φ , которое требуется реализовать автоматом:

$01 \rightarrow 01$ $12 \rightarrow 02$
 $02 \rightarrow 01$ $13 \rightarrow 03$
 $03 \rightarrow 02$ $20 \rightarrow 03$
 $10 \rightarrow 02$ $21 \rightarrow 03$
 $11 \rightarrow 02$

Алфавитное отображение φ , будучи естественным образом продолжено на начальные отрезки слов, является, очевидно, автоматным.

Находим соответствующее ему каноническое множество событий:

$$S(0) = 0 \vee 1 \vee 2,$$

$$S(1) = 01 \vee 02,$$

$$S(2) = 03 \vee 10 \vee 11 \vee 12,$$

$$S(3) = 13 \vee 20 \vee 21.$$

Применим второй вариант синтеза, исключив из рассмотрения событие $S(0)$. Оставшиеся события запишем в виде регулярных выражений:

$$1 = 0 (1 \vee 2),$$

$$2 = (03 \vee 10 \vee 11 \vee 12),$$

$$3 = (13 \vee 20 \vee 21).$$

Разметка этих выражений по правилам 2, 2б, 2в, 3 (пп. 3. 3. 1, 3. 3. 2) приводит к следующему разметочному комплексу:

$$1 = \begin{array}{|c|c|c|c|} \hline 0 & (& 1 & \vee & 2) \\ \hline \boxed{0} & 1 & & \\ \hline & \boxed{1} & \boxed{1} & \\ \hline \end{array}, \quad 2 = \begin{array}{|c|c|c|c|c|c|} \hline (& 0 & 3 & \vee & 1 & 0 & \vee & 1 & 1 & \vee & 1 & 2 \\ \hline 0 & 1 & 2 & 2 & 2 & & & & & & \\ \hline \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & & & & & & \\ \hline \end{array}$$

$$2 = \begin{array}{|c|c|c|c|c|} \hline (& 1 & 3 & \vee & 2 & 0 & \vee & 2 & 1 \\ \hline 0 & 2 & 3 & 3 & & & & & \\ \hline \boxed{0} & \boxed{0} & \boxed{0} & & & & & & \\ \hline \end{array}$$

По правилам 4 и 5а находим таблицы переходов (3.12) и выходов (3.13) автомата Мили.

Таблица 3.12

	0	1	2	3	*
0	1	*	*	*	*
1	2	*	*	*	*
2	3	*	*	*	*
3	*	*	*	*	*

Таблица 3.13

	0	1	2	3	*
0	()	()	2	3	()
1	()	1	2	3	()
2	()	1	2	()	()
3	()	2	3	()	()

В соответствии со вторым вариантом синтеза автомата пустой выходной сигнал () должен быть заменен выходным сигналом 0. Однако, очевидно, что применяя этот вариант синтеза необходимо учесть особенности его применения, указанные в четвертом этапе общего метода решения задачи.

По правилу ба получим таблицы переходов (3.14) и выходов (3.15) искомого синтезируемого автомата Мили.

Таблица 3.14

	0	1	2	3
0	1	-	-	-
1	2	-	-	-
2	3	-	-	-
3	-	-	-	-

Таблица 3.15

	0	1	2	3
0	0	0	2	3
1	0	1	2	3
2	0	2	2	0
3	0	3	3	0

3.5.3. Синтез автомата Мура

Синтез автомата Мура рассмотрим на следующем примере.

Пример. Построить конечный автомат Мура, возводящий в квадрат двузначные двоичные числа. Числа подаются на вход автомата последовательно, разряд за разрядом (младшими разрядами) и, соответственно, после преобразования выдаются на выход.

Решение. Поскольку после возведения в квадрат двузначного числа полученное число может быть четырехзначным, то в первоначально заданной таблице соответствия входные и выходные слова имеют различную длину. Используя прием выравнивания длин слов, добавим к входным словам справа по две пустые буквы, так как числа подаются младшими разрядами. В качестве пустой буквы выберем цифру 0, поскольку добавление нуля в старшие разряды входных слов позволяет однозначно восстановить их первоначальный вид. После этого добавления получаем автоматное отображение φ , заданное следующей сокращенной таблицей соответствия:

0000 \rightarrow 0000

1000 \rightarrow 1000

0100 \rightarrow 0010

1100 \rightarrow 1001

Каноническое множество событий для отображения φ имеет вид:

$S(0) = 0 \vee 00 \vee 000 \vee 0000 \vee 10 \vee 100 \vee 1000 \vee 01 \vee 0100 \vee 11 \vee 1100$,

$S(1) = 1 \vee 010 \vee 1100$.

Первое из найденных событий можно представить более простым регулярным выражением, записывая вместо первых значений $\{0\}$, а вместо трех следующих — $10 \{0\}$. Однако можно поступить проще, применив второй вариант синтеза с исключением события $S(0)$. Что касается второго события, то для него получим следующее регулярное выражение:

$1 = (1 \vee 010 \vee 1100)$.

Произведем разметку мест в этом выражении, чтобы найти в нем два соответственных и два подобных места:

$1 = ((1 \vee 0 | 1 | 0 | \vee | 1 | 1 | 0 | 0 |)) |$.

0	1	2	3	6	1	4	5	6
---	---	---	---	---	---	---	---	---

0	0	0	1
			6

По полученному выражению находим отмеченную таблицу переходов автомата Мура (табл. 3.16).

Таблица 3.16

	-	1	()	()	()	()	1	()
	0	1	2	3	4	5	6	*
0	2	*	*	6	5	6	*	*
1	1	4	3	*	*	*	*	*

В соответствии с правилами второго варианта синтеза в этой таблице заменим пустой выходной сигнал () – выходным сигналом 0. Произведя переобозначения пустого состояния (* → 7), получаем окончательный вид отмеченной таблицы переходов искомого автомата Мура (табл. 3.17).

Таблица 3.17

	-	1	0	0	0	0	1	0
	0	1	2	3	4	5	6	7
0	2	7	7	6	5	6	7	7
1	1	4	3	7	7	7	7	7

Таким образом, для построения абстрактных автоматов Мура и Мили функционирование управляющего автомата представляют в виде таблиц переходов и выходов. В дальнейшем состояния автомата кодируются двоичными кодами, определяется тип и количество триггеров. По таблице переходов устанавливаются значения сигналов на входах триггеров, по которым осуществляются переходы; определяются функции возбуждения триггеров и производится их минимизация. По найденным выражениям строится схема управляющего автомата на выбранных элементах. Такие задачи решаются при синтезе *структурных конечных автоматов*.

Контрольные вопросы

1. Понятие об абстрактном автомате и индуцируемом им отображении.
2. Автоматные отображения и события.
3. Представление событий в автоматах.
4. Регулярные языки и конечные автоматы.
5. Основной алгоритм синтеза конечных автоматов.
6. Получение не полностью определенных автоматов.
7. Модель Мили.
8. Модель Мура.
9. Связь между моделями Мили и Мура.

4. СТРУКТУРНЫЙ КОНЕЧНЫЙ АВТОМАТ

4.1. Основные понятия структурной теории автоматов

В теории автоматов выделяют абстрактную теорию автоматов и структурную теорию автоматов. По сравнению с абстрактной теорией в структурной теории дела-

ются дальнейшие шаги в направлении учета большого числа свойств реально существующих дискретных автоматов. Главная отличительная особенность структурной теории автоматов состоит в том, что, в отличие от абстрактной теории, она учитывает структуру входных и выходных сигналов автомата, а также его внутреннюю структуру на уровне так называемых структурных схем, обеспечивающих заданное преобразование дискретной информации. Основной задачей структурной теории автоматов является изучение композиции автоматов, то есть методов построения сложных автоматов из автоматов, являющихся более простыми.

Следует подчеркнуть, что структурная теория автоматов не ставит своей задачей отразить все свойства реально существующих автоматов. В ней, например, не учитываются переходные процессы в автоматах, вопросы надежности работы автоматов, физические свойства сигналов и т. д. В этом смысле структурная теория автоматов также остается в значительной мере абстрактной теорией, хотя она и отличается значительно меньшей степенью абстракции, чем собственно абстрактная теория автоматов.

При проектировании устройств преобразования информации, как правило, структурной теории автоматов предшествует абстрактная теория автоматов. Синтезируя реальные автоматы многие вопросы проще и эффективнее решать на уровне абстрактной теории. К числу таких вопросов относят определение необходимого объема памяти автомата (то есть числа его состояний), переходов в памяти, а также вопросы, относящиеся к минимизации числа состояний автомата. В то же самое время имеется ряд других вопросов – таких, например, как вопрос о композиции автоматов, – сама постановка которых уже выходит за рамки абстрактной теории автоматов. Таким образом, абстрактная и структурная теории автоматов не только взаимно дополняют друг друга, но могут иметь и собственные естественные области приложения.

Обычно в структурной теории автоматов сохраняется абстракция дискретного автоматного времени, однако, иногда приходится несколько изменять порядок отсчета временных интервалов (способ отсчета времени). В абстрактной теории автоматов входные и выходные сигналы относят к моменту перехода автомата из одного состояния в другое, а в структурной теории моменты перехода автомата из одного состояния в другое удобно считать *границами* интервалов, относящихся к одному и тому же значению автоматного времени.

При существующем способе отсчета времени функции переходов и выходов (автомата Мили) задаются следующим образом:

$$\begin{aligned} q(t+1) &= \delta(q(t), x(t)), \\ y(t) &= \lambda(q(t), x(t)), \quad (t = 0, 1, 2, \dots). \end{aligned}$$

Функции переходов и выходов автомата Мура приобретают вид:

$$\begin{aligned} q(t+1) &= \delta(q(t), x(t)), \\ y(t) &= \lambda(q(t)), \quad (t = 0, 1, 2, \dots). \end{aligned}$$

Очевидно, что эти формулировки законов функционирования автоматов нуждаются в несколько иной интерпретации, учитывающей особенности принимаемого теперь способа отсчета времени. В частности, следует учитывать два обстоятельства.

Первое обстоятельство состоит в том, что момент начала отсчета времени совпадает с нулевым моментом не только для состояний автомата, но также и для входных и выходных сигналов.

Это замечание в равной степени касается как автоматов Мура, так и автоматов Мили и обуславливает необходимость рассмотрения входных и выходных последова-

тельностью вида $x(0), x(1), \dots, x(t-1)$ вместо последовательностей вида $x(1), x(2), \dots, x(t)$. Подобный сдвиг во времени ни в какой мере не повлияет на правильность получаемых результатов.

Второе обстоятельство, относящееся только к автоматам Мура, связано с тем, что результатом воздействия входного слова $x(0), x(1), \dots, x(t-1)$ к начальному состоянию автомата Мура следует считать выходное слово $y(1), y(2), \dots, y(t)$, а не $y(0), y(1), \dots, y(t-1)$. Это связано с тем, что для автоматов Мура выходной сигнал, индуцированный каким-либо входным сигналом $x(t)$, при новом способе отсчета времени относится не к моменту t появления сигнала $x(t)$, а к непосредственно следующему за ним моменту времени $t + 1$. Кроме того, при предлагаемом способе отсчета времени пустое слово оказывается всегда представленным в автоматах Мура выходным сигналом $y(0)$, в то время как ранее оно не могло быть представленным никаким выходным сигналом.

Частичные автоматы в структурной теории рассматриваются как такие автоматы, у которых функции переходов и выходов всюду определены, за исключением некоторых точек, в которых значения этих могут определяться иначе, т. е. иметь другую реализацию.

В отличие от абстрактной теории автоматов в структурной теории как входные, так и выходные каналы автоматов считаются состоящими из нескольких **элементарных входных** и, соответственно, **элементарных выходных каналов**. По всем элементарным каналам могут передаваться лишь так называемые **элементарные сигналы**. Набор всех возможных для данного автомата элементарных сигналов называется **структурным алфавитом** этого автомата. Структурный алфавит должен быть непременно конечным. При этом, природа букв (элементарных сигналов), составляющих структурный алфавит, не учитывается.

Также предполагается, что каждый элементарный канал (входной или выходной) подсоединяется к так называемому **узлу**. Узлы, к которым подсоединены элементарные входные каналы, называются **входными узлами автомата**, а узлы, к которым подсоединены элементарные выходные сигналы – **выходными узлами**.

В каждом автомате осуществляется определенная циркуляция элементарных сигналов. Элементарные входные сигналы поступают вначале на входные узлы, а затем по входным каналам поступают в автомат. Элементарные выходные сигналы по выходным каналам, поступают на выходные узлы, к которым эти каналы присоединены.

При графическом изображении автомата узлы обозначаются точками или маленькими **кружочками**, элементарные входные и выходные каналы – **сплошными линиями**, а сами автоматы – различными геометрическими фигурами (прямоугольниками, кругами и т. д.). Для того, чтобы отличить входные каналы от выходных направление передачи по ним отмечают стрелками (рис. 4.1).

Таким образом, под термином «автомат» в структурной теории автоматов понимается абстрактный автомат с явно заданными элементарными входными и выходными каналами и соответствующими им входными и выходными узлами.

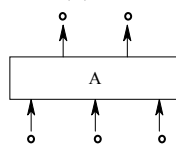


Рис. 4.1. Графическое изображение узла

При этом определенным образом нумеруются как входные, так и выходные узлы автомата, а входной и выходной сигналы задаются конечными упорядоченными наборами элементарных сигналов.

Наборы элементарных сигналов называются **векторами** в структурном алфавите, составляющие их элементарные сигналы – **компонентами вектора**, а число компонент – **размерностью** вектора. Нумерация компонент этих векторов соответствует нумерации входных и выходных узлов, то есть на i -й входной узел передается i -я компонента *входного вектора*, а на j -й выходной узел – j -я компонента *выходного вектора* (вектора, изображающего выходной сигнал).

В отличие от абстрактных входных и выходных сигналов векторные представления таких сигналов называются структурными (входными и выходными) сигналами. Переход от абстрактных входных и выходных сигналов к структурным называется **кодированием** соответствующих абстрактных сигналов в структурном алфавите автомата.

Вектор, получающийся в результате кодирования какого-либо абстрактного сигнала, обозначается обычно той же буквой, но жирным шрифтом (или чертой над буквой). При этом записи x обычно соответствует переменный входной структурный сигнал, а записи y – переменный выходной структурный сигнал.

Также считается, что на всех выходных узлах всякого автомата Мура выходные сигналы возникают в каждый момент автоматного времени, независимо от того, подаются ли какие-либо сигналы на его входные узлы или нет. В автомате Мили сигналы на выходных узлах появляются в тот или иной момент автоматного времени, то есть тогда и только тогда, когда в **тот же самый момент времени** сигналы поданы на все его входные узлы.

В некоторых случаях при построении структурного алфавита автомата в алфавит включается в качестве особого сигнала так называемый **естественный нулевой сигнал**, возникающий на изолированных (т. е. не присоединенных ни к какому каналу) узлах. Например, если элементарные сигналы представляют собой электрические импульсы различной величины, то естественным нулевым сигналом будет **отсутствие** каких бы то ни было импульсов в тот или иной момент автоматного времени. В случае включения в структурный алфавит таких естественных нулевых сигналов на выходных узлах автоматов Мили сигналы будут появляться и тогда, когда входные узлы этих автоматов не подсоединены ни к каким источникам сигналов.

Возможны и другие случаи (например, при представлении сигналов в виде уровней электрического потенциала), когда естественный нулевой сигнал не включается в структурный алфавит, то есть, иначе говоря, не рассматривается как один из возможных элементарных сигналов. В таких случаях для получения определенных (принадлежащих структурному алфавиту) элементарных сигналов на выходных узлах автоматов Мили, подсоединение входных узлов этих автоматов к источнику сигнала является обязательным.

4.2. Композиция автоматов и структурные схемы

Рассматривая способы композиции автоматов, условимся при рассмотрении той или иной системы автоматов считать, что все входящие в систему автоматы имеют один и тот же структурный алфавит и работают в одном и том же дискретном ав-

томатном времени. Введение общего автоматного времени для системы автоматов не означает, вообще говоря, отказ от рассмотрения асинхронных автоматов. Речь идет лишь о том, что совместная работа автоматов в системе определяет общее дискретное время для всех входящих в нее автоматов, отличное, вообще говоря, от того автоматного времени, в котором эти автоматы работали бы вне данной системы.

Общий способ композиции автоматов заключается в следующем. Пусть A_1, \dots, A_n ($n \geq 0$) – конечное множество автоматов. Произведем объединение этих автоматов в систему совместно работающих автоматов.

Введем в рассмотрение некоторое конечное множество других узлов, которые назовем внешними выходными узлами. Эти узлы предполагаются отличными от входных и выходных узлов рассматриваемых автоматов, которые в отличие от введенных внешних узлов будем называть внутренними. Чтобы подчеркнуть различие между двумя введенными типами узлов, внешние узлы называют иногда полюсами.

При графическом изображении системы автоматов внутренние узлы чаще всего обозначаются точками, а внешние – кружочками. При построении системы автоматов фиксируется определенная нумерация как для внешних входных, так и для внешних выходных узлов (полюсов).

Собственно композиция автоматов состоит в том, что в полученной системе, состоящей из данных автоматов A_1, \dots, A_n и внешних узлов, производится отождествление некоторых узлов (как внешних, так и внутренних).

Операция отождествления узлов с целью обеспечения совместной работы системы автоматов состоит в том, что элементарный сигнал, поступающий на один из узлов, входящих в множество отождествленных между собой узлов, попадает тем самым на все узлы этого множества. Реализация такой операции в электронных цифровых автоматах соответствует соединению этих узлов проводниками. В дальнейшем под отождествлением узлов будем также понимать **соединение** узлов друг с другом.

При графическом изображении отождествление узлов производится либо их фактическим совмещением, либо соединением их сплошными линиями (вообще говоря, ломаными и, возможно, проходящими через другие узлы).

В результате проведения операции отождествления (соединения) узлов, все узлы, входящие в данную систему, разобьются на попарно непересекающиеся множества соединенных между собою узлов. Некоторые из этих множеств могут быть и одноэлементными (состоящими из единственного узла).

После проведенных отождествлений система автоматов превращается в так называемую **схему**, или **сеть автоматов**. Будем считать, что автоматы, входящие в схему автоматов, работают совместно, если в каждый момент t автоматного времени ($t = 0, 1, 2, \dots$) на все внешние входные узлы схемы подается какой-либо набор элементарных сигналов, а со всех внешних выходных узлов схемы снимается получающийся на них набор элементарных выходных сигналов.

Предположим, что в каждый момент дискретного времени $t = 0, 1, 2, \dots$ структурный выходной сигнал схемы однозначно определяется поступившей к этому времени конечной последовательностью структурных входных сигналов, начальными состояниями входящих в схему автоматов и полученными при построении схемы соединениями узлов. В этом случае построенная схема может рассматриваться как некоторый автомат A , а схема будет называться структурной схемой этого автомата.

Таким образом, полученный описанным способом автомат A есть результат композиции исходных автоматов A_1, \dots, A_n . Входными узлами этого автомата служат внешние входные узлы схемы, а выходными узлами – внешние выходные узлы.

4.3. Условия корректности и правильности построения схем

Операция отождествления узлов не всегда определена однозначно. Поэтому существует много различных возможностей для композиции одних и тех же автоматов. Вместе с тем не всякое отождествление узлов приводит к схеме, которую можно рассматривать в качестве структурной схемы некоторого автомата. Для того, чтобы это можно было сделать, необходимо при отождествлении узлов соблюдать два условия, которые называются *условиями корректности построения схемы*.

Первое условие состоит в том, что в любой момент автоматного времени на всех внешних выходных узлах схемы должны появляться какие-то элементарные сигналы. Считая, что в схеме не должно быть заведомо лишних узлов (то есть таких узлов, которые можно исключить из схемы, не нарушая ее функционирования), условимся, что в любой момент времени на каждый узел схемы (как внешний, так и внутренний) поступает какой-либо элементарный сигнал.

Второе условие корректности построения схемы заключается в том, что неоднозначность элементарных сигналов в каком-либо узле схемы хотя бы в один момент времени будет считаться *недопустимой*.

Существует два источника неоднозначности элементарного сигнала в узле. Во-первых, неоднозначность может иметь место в случае, если к какому-нибудь узлу подсоединено одновременно несколько выходных узлов, являющихся источниками элементарных сигналов.

На практике присоединение к одному и тому же узлу нескольких выходных узлов часто не приводит к возникновению неоднозначности. Это бывает только тогда, когда среди элементарных сигналов определена некоторая *естественная упорядоченность*, в силу которой одновременный приход нескольких элементарных сигналов эквивалентен приходу лишь наибольшего из всех фактически пришедших сигналов. В этом случае будет иметь место *естественное разделение элементарных сигналов*. Такое естественное разделение наблюдается, например, в случае релейно-контактных схем, а также во многих электронных схемах с импульсным представлением элементарных сигналов.

Во-вторых, источником неоднозначности элементарных сигналов в узле могут быть так называемые *циклические цепи* или петли структурных схем. Условимся называть *цепью* конечную упорядоченную последовательность автоматов B_1, \dots, B_k , входящих в схему, у которых хотя бы один из выходных узлов каждого предыдущего автомата соединен с некоторым входным узлом следующего за ним автомата. Если к тому же хотя бы один из выходных узлов последнего автомата цепи соединен с каким-нибудь входным узлом первого автомата той же цепи, то цепь называется *циклической*. Обо всех узлах, входящих в цепи и петли, говорят как об узлах этой цепи или этой петли.

О цепи B_1, \dots, B_k говорят, что она начинается любым из входных узлов автомата B_1 и кончается любым из выходных узлов автомата B_k . Условимся также, что два отождествленных между собою узла a и b или один даже единственный узел составляют цепь (состоящую из пустого множества автоматов). В первом случае цепь *начи-*

нается любым из узлов a или b и *кончается* другим из этих узлов. Во втором случае цепь начинается и кончается узлом a . Циклические цепи, состоящие из пустого множества автоматов, называются *тривиальными* и в дальнейшем не рассматриваются.

Покажем на примере, каким образом может возникнуть неоднозначность в петле. Предположим, что каждый из трех автоматов Мили, входящих в цепь (рис. 4.2), обладает следующим свойством: структурный алфавит состоит из двух элементарных сигналов 0 и 1, и каждый автомат цепи переводит входной сигнал 0 в выходной сигнал 1, а входной сигнал 1 – в выходной сигнал 0; при этом входной и соответствующий ему выходной сигналы возникают в один и тот же момент времени.

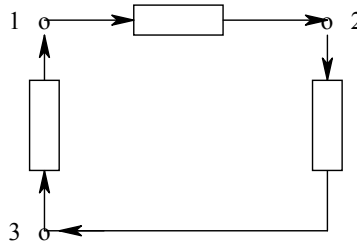


Рис. 4.2. Некорректно построенная цепь

Легко видеть, что при таком определении сигналов во всех входящих в цепь узлах будут возникать противоречия. Действительно, полагая, что если в какой-то момент времени в узле 1 сигнал 0, то в узле 2 в тот же самый момент времени будет сигнал 1, а в узле 3 – сигнал 0, следовательно, в узле 1 должен быть сигнал 1, хотя был сигнал 0, что и приводит к противоречию.

Петли, подобные только что рассмотренной, считают *некорректно построенными*. Некорректность такого рода не будет иметь места, если хотя бы один из входящих в петлю автоматов Мили заменить автоматом Мура. Действительно, автомат Мура реагирует на тот или иной входной сигнал выходным сигналом, возникающим на один элементарный промежуток времени позже, чем вызвавший его появление входной сигнал. Поэтому в данном примере сигнал 0, переданный в момент времени t в узел 1, вызвал бы в результате передачи его вдоль петли появление в этом узле сигнала 1 не в момент времени t , а в момент времени $t + 1$.

Рассмотрим, как построить схемы с соблюдением первого условия корректности.

Условие соблюдается автоматически, если в структурный алфавит включить естественный нулевой сигнал. В противном случае для соблюдения этого условия необходимо при проведении операции отождествления узлов выполнить определенные требования.

Прежде всего заметим, что все внешние входные узлы и все внутренние узлы, являющиеся выходными узлами автоматов Мура, по определению, получают элементарные сигналы во все моменты автоматного времени. Назовем все эти узлы *задающими узлами*. Сформулируем условие, обеспечивающее передачу элементарных сигналов от задающих узлов во все узлы автомата в каждый момент автоматного времени. Для этой цели учтем, что выходные сигналы у автоматов Мили, по определению, возникают одновременно с воздействием сигналов на все его входные узлы. Соответствующие условия можно сформулировать следующим образом:

1.1. Для любого узла схемы должна иметься цепь, состоящая из некоторого множества (может быть пустого) автоматов Мили, начинающаяся каким-либо задающим узлом и кончающаяся данным узлом. Назовем это условие *первым условием правильности построения схемы*.

1.2. Любой узел схемы должен быть отождествлен (соединен) не более чем с одним внешним входным или внутренним выходным узлом.

1.3. Любая правильная (содержащая не менее одного автомата) петля в схеме должна содержать в своем составе хотя бы один автомат Мура.

Схема, в которой выполнены все три условия: 1. 1, 1. 2, 1. 3, называется **правильной схемой**, а операция, состоящая в построении такой схемы – **правильной композицией автоматов**. Выполнение этих условий позволяет сформулировать условие 1.4.

1.4. Всякая правильная схема может рассматриваться как структурная схема некоторого автомата. Правильная композиция задает некоторый класс операций на множестве автоматов. Рассмотрим некоторые частные случаи таких операций более подробно.

Первый случай – операция пересоединения внешних узлов (композиция пустого множества автоматов).

В этом случае схема предполагается состоящей только из внешних (входных и выходных) узлов. Из условий 1. 1 и 1. 2 непосредственно следует, что каждый выходной узел в такой схеме должен быть соединен в точности с одним входным узлом. Схема представляет собой автомат Мили без памяти (с одним состоянием). Компонентами структурного выходного сигнала автомата служат компоненты его структурного входного сигнала, взятые, вообще говоря, в другом порядке и, быть может, повторенные по несколько раз (рис. 4.3).

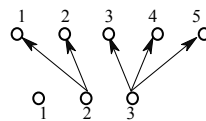


Рис. 4.3. Пересоединение внешних узлов

Автомат, полученный в результате этой композиции, сопоставляет структурному входному сигналу $x = (\alpha, \beta, \gamma)$ структурный выходной сигнал $y = (\beta, \beta, \gamma, \gamma, \gamma)$.

Второй случай – операция подстановки входов в автомате. Пусть имеется произвольный автомат A с m входными и n выходными узлами. Строится система, состоящая из автомата A и m внешних входных и из n выходных узлов, i -й выходной узел автомата A отождествляется с i -тым внешним выходным узлом ($i = 1, \dots, n$). Каждый входной узел автомата A отождествляется в точности с одним внешним входным узлом. Полученный в результате композиции автомат действует так же, как и исходный автомат A при условии, что на автомат A подается не исходный структурный входной сигнал x , а структурный входной сигнал, полученный из него в результате некоторой подстановки его компонент (рис. 4.4).

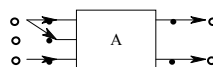


Рис. 4.4. Постановка входов в автомате

Третий случай – операция подстановки выходов. Определяется по аналогии с предыдущим случаем.

Четвертый случай – операция суперпозиции автоматов. Рассматриваются два автомата A и B , при этом у автомата A имеется m входных и n выходных узлов, а у автомата B – p входных и r выходных узлов. Суперпозиция этих автоматов заключается в построении системы, состоящей из автоматов A и B , m внешних входных и r внешних выходных узлов посредством соединения i -го внешнего входного узла с i -м входным узлом автомата A ($i = 1, \dots, m$), j -го выходного узла автомата A с j -м входным узлом автомата B ($j = 1, \dots, n$) и k -го выходного узла автомата B с k -м внешним выходным узлом ($k = 1, \dots, r$).

Пятый случай – операция объединения автоматов. Пусть имеется любое конечное множество автоматов A_1, \dots, A_p . Каждый автомат A_i имеет m_i входных и n_i выходных узлов ($i = 1, \dots, p$). Строится система, состоящая из данных автоматов, у которых $m_1 + m_2 + \dots + m_p$ внешних входных узлов и $n_1 + n_2 + \dots + n_p$ внешних выходных узлов. При этом каждый из внешних входных узлов соединяется в точности с одним из входных узлов автоматов A_1, \dots, A_p . Точно такое же отождествление, по принципу взаимно однозначного соответствия, осуществляется между выходными узлами данных автоматов и внешними выходными узлами.

В дальнейшем будут рассматриваться в основном правильные схемы. Однако правильные схемы не всегда исчерпывают всех корректно построенных схем и это позволяет в ряде случаев пользоваться схемами, не принадлежащими к числу правильных. Отсюда следуют два важнейших случая.

Во-первых, очевидно, что в случае наличия в структурном алфавите естественного нулевого сигнала можно не соблюдать условие правильности 1.1 и, тем не менее, получать корректно построенные схемы. Условие 1.1 в таких схемах будет выполненным после введения так называемого нулевого автомата.

Нулевым автоматом называют автомат Мили с одним входным и одним выходным узлом, отличающийся тем, что на его выходном узле при любом сигнале на входном узле появляется нулевой сигнал.

При наличии в схеме узлов, для которых условие 1.1 не выполняется, можно, не изменяя условий работы схемы, вставить в нее добавочные цепи из нулевых автоматов, соединяющие задающие узлы со всеми такими узлами. После такой операции условие 1.1 будет выполнено.

Во-вторых, в случае, когда имеется естественное разделение элементарных сигналов, можно не соблюдать условие правильности 1.2 и, тем не менее, получать корректно построенные схемы. Однако в этом случае необходимо интерпретировать указанные схемы, чтобы выполнялось условие 1.2.

Этой цели можно достичь, введением понятия *разделяющего автомата Мили*, или просто разделения. В разделяющем автомате Мили на единственный выходной узел автомата передается всегда самый большой элементарный сигнал из числа входных сигналов, поданных в то же самое время на его входные узлы. Если в схеме с естественным разделением сигналов имеется соединение какого-нибудь узла с несколькими внешними входными или внутренними выходными узлами $\alpha_1, \dots, \alpha_k$, то можно считать этот узел соединенным с выходным узлом разделяющего автомата, входные узлы которого соединены с узлами $\alpha_1, \dots, \alpha_k$. При таком дополнении схемы ее работа не изменится, но условие 1.2 будет уже соблюдаться.

Прежде чем сформулировать условие 1.5 рассмотрим состояния автомата A , которые могут быть получены в результате композиции некоторых автоматов A_1, \dots, A_k . Состояниями такого автомата A можно считать упорядоченные наборы

(q_1, \dots, q_k) состояний всех автоматов A_1, \dots, A_k . Полученный отдельный набор обозначается \mathbf{q} и называется **структурным состоянием** автомата A . В дальнейшем будем предполагать, что в результате композиции автоматов, кроме указанных комбинаций уже имевшихся состояний, других состояний не возникает. Тогда справедливо следующее условие.

1.5. Число состояний автомата, полученных в результате композиций автоматов A_1, \dots, A_n , равно произведению чисел состояний всех этих автоматов.

4.4. Канонический метод структурного синтеза автомата

4.4.1. Основная задача теории структурного синтеза автоматов

Основной задачей теории структурного синтеза автоматов является нахождение общих приемов построения структурных схем автоматов на основе композиции автоматов, принадлежащих к заранее заданному конечному числу типов автоматов. Более точно эта задача может быть сформулирована следующим образом.

2.1. Пусть задано некоторое конечное множество автоматов в одном и том же структурном алфавите Z . Назовем эти автоматы элементарными автоматами и предположим, что таких элементарных автоматов имеется неограниченное число экземпляров. Пусть также задан произвольный конечный автомат A в том же самом структурном алфавите Z . Необходимо найти алгоритм, позволяющий по заданному автомату A строить некоторую композицию элементарных автоматов так, чтобы полученный в результате композиции автомат индуцировал отображение, продолжающее отображение, индуцируемое автоматом A .

Следует отметить, что далеко не при всяком выборе системы элементарных автоматов эта задача имеет решение. Однако, если решение имеется (для произвольного конечного автомата A), то считают, что заданная система элементарных автоматов структурно полна или может иметь ослабленную структурную полноту.

Система элементарных автоматов **ослабленно структурно полна**, если для любого отображения φ , индуцируемого конечным автоматом, можно найти такую композицию элементарных автоматов, что получаемый в результате этой композиции автомат индуцирует отображение, которое продолжает либо само отображение φ , либо отображение, полученное из отображения φ в результате применения к нему операции выравнивания длин слов. При этом предполагается, что автомат, индуцирующий отображение φ , задан в том же структурном алфавите, что и все элементарные автоматы. Заметим также, что операция выравнивания длин слов, применяемая к отображению φ , являющемуся автоматным отображением, заключается в дописывании равного числа пустых букв справа к входным словам и слева к выходным словам отображения φ .

В настоящее время почти нет эффективных методов (существенно более простых, чем метод перебора всех вариантов) решения основной задачи структурного синтеза при любом выборе структурно полных систем элементарных автоматов. Наиболее часто применяют так называемый **канонический метод** структурного синтеза автоматов, пригодный для структурно полных систем элементарных автоматов некоторого специального вида.

Канонический метод структурного синтеза оперирует с элементарными автоматами, которые делят на два больших класса. Первый класс составляют элементарные **автоматы с памятью** (т. е. автоматы, имеющие более одного внутреннего состояния); такие автоматы называются элементами памяти или запоминающими элементами. Второй класс составляют **автоматы без памяти** (т. е. автоматы с одним внутренним состоянием), которые принято называть **комбинационными**, или **логическими элементами**.

При принятых допущениях в результате композиции логических элементов всегда получают автоматы без памяти или, как их иначе называют, комбинационные схемы. Каждая комбинационная схема (автомат без памяти) характеризуется векторной функцией выходов, устанавливающей зависимость структурного выходного сигнала $\mathbf{x}(t)$ от структурного входного сигнала $\mathbf{y}(t)$ в один и тот же момент автоматного времени:

$$\mathbf{y}(t) = \lambda \mathbf{x}(t),$$

где λ – векторная функция выходов.

Задание такой функции эквивалентно заданию системы обычных (скалярных) функций, устанавливающих зависимость каждой компоненты вектора $\mathbf{y}(t)$ от компонент (в общем случае всех) вектора $\mathbf{x}(t)$.

По аналогии с основной задачей структурного синтеза автоматов может быть сформулирована задача **структурного синтеза комбинационных схем**.

2.2. Пусть задано некоторое конечное множество логических элементов (элементов без памяти) в одном и том же структурном алфавите Z' ; предположим, что таких логических элементов имеется неограниченное число экземпляров. Требуется найти общий конструктивный прием (алгоритм), позволяющий по любому конечному автомату A без памяти в структурном алфавите Z' осуществить некоторую композицию заданных логических элементов так, чтобы полученная в результате композиции комбинационная схема имела ту же самую векторную функцию выходов, что и заданный автомат A .

В некоторых случаях при определенном выборе системы логических элементов сформулированная задача не имеет решения. В другом случае, когда решение имеется (для произвольного конечного автомата без памяти), считают, что заданная система логических элементов **функционально полна**.

Обычно на практике векторная функция выходов λ исходного для комбинационного синтеза автомата без памяти A задается не во всех точках, так как в некоторых точках значения функции не определены (безразличны). В таком случае комбинационная схема, получаемая при решении задачи комбинационного синтеза, должна иметь векторную функцию выходов, совпадающую с функцией λ во всех точках, в которых значения функции λ определены.

Сущность канонического метода структурного синтеза автоматов заключается в том, чтобы свести задачу структурного синтеза произвольных автоматов к задаче структурного синтеза комбинационных схем (автоматов без памяти). При этом специальным образом производится выбор запоминающих элементов. В качестве запоминающих элементов выбираются автоматы Мура, обладающие **полной системой переходов** и **полной системой выходов**.

2.3. Полнота системы переходов в автомате Мура означает, что для любой упорядоченной пары состояний этого автомата найдется входной сигнал, переводящий

первый элемент этой пары во второй. Иначе говоря, если $\delta(q, x)$ – функция переходов автомата, то для полноты системы переходов в этом автомате необходимо и достаточно, чтобы для любой пары (q, b) его состояний уравнение $b = \delta(q, x)$ было бы разрешимым относительно входного сигнала x .

2.4. Полнота системы выходов в автомате Мура означает, что каждому состоянию автомата соответствует свой собственный выходной сигнал, отличный от выходного сигнала, соответствующего любому другому состоянию. Иначе говоря, для полноты системы выходов автомата Мура необходимо и достаточно, чтобы его сдвинутая функция выходов $y = \lambda(q)$ осуществляла взаимно однозначное отображение множества состояний автомата на множество всех его выходных сигналов.

2.5. В автомате Мура с полной системой выходов можно отождествить внутренние состояния с выходными сигналами автомата. Тогда один и тот же (структурный) алфавит употребляется не только для обозначения (кодирования) входных и выходных сигналов, но и для кодирования внутренних состояний этого автомата.

Для автоматов с полной системой переходов целесообразно ввести следующее определение.

2.6. Функцией входов автомата A с полной системой переходов и заданной функцией переходов $\delta(q, x)$ называется функция $x = \mu(q, b)$ (обычно неоднозначная), заданная на упорядоченных парах состояний автомата A . Для каждой такой пары (q, b) в качестве значения функции входов $\mu(q, b)$ выбирается (непустое) множество всех тех входных сигналов x , для которых $\delta(q, x) = b$.

Функции входов конечных автоматов задаются *таблицами входов*. В таблице входов на пересечении q -й строки b -го столбца помещается множество входных сигналов, вызывающих переход автомата из состояния q в состояние b .

Таким образом, в реальных цифровых автоматах, как правило, употребляются запоминающие элементы, представляющие собой автоматы Мура с полной системой переходов и с полной системой выходов. При этом ограничения, вводимые каноническим методом структурного синтеза, с практической точки зрения являются несущественными.

4.4.2. Теорема о структурной полноте

Рассмотрим справедливость следующего утверждения, которое назовем теоремой о структурной полноте.

2.7. Всякая система элементарных автоматов считается **структурно-полной системой**, если в этой системе имеется автомат Мура с нетривиальной памятью, обладающий полной системой переходов и полной системой выходов и какой-нибудь функционально полной системой логических элементов. Существует общий конструктивный прием (канонический метод структурного синтеза), позволяющий в данном случае свести задачу структурного синтеза произвольных конечных автоматов к задаче структурного синтеза комбинационных схем.

Для доказательства теоремы 2.7 синтезируем произвольный конечный автомат A . Выберем запоминающие элементы A_1, \dots, A_p , обладающие полными системами переходов и выходов, причем такие, что произведение чисел их состояний не меньше, чем число состояний автомата A . Каждому состоянию q автомата A поставим в соответствие конечную последовательность (q_1, \dots, q_p) состояний автоматов A_1, \dots, A_p так,

что различным состояниям автомата A ставятся в соответствие различные последовательности. Назовем этот процесс **кодированием состояний** автомата A .

После кодирования состояний (выполняемого произвольным образом) возникают структурные состояния автомата A .

Учитывая 2.5 (см. п.4.4.1), считаем эти структурные состояния векторами в структурном алфавите; при этом компонента q_i в структурном состоянии (q_1, \dots, q_p) заменяется групповой компонентой структурного выходного сигнала v_i запоминающего элемента A_i , соответствующего состоянию q_i .

Обозначим структурное состояние (v_1, \dots, v_p) через v . Структурное состояние v можно рассматривать и как структурный выходной сигнал объединения автоматов A_1, \dots, A_p . Через u обозначим структурный входной сигнал этого объединения, который назовем **структурным входным сигналом памяти автомата A** .

Кодирование состояний автомата A позволяет функцию переходов и закон функционирования автомата представить в векторном виде

$$v(t+1) = \delta(v(t), x(t)). \quad (1)$$

Переход автомата A из состояния $v(t)$ в состояние $v(t+1)$ складывается из соответствующих переходов автоматов A_1, \dots, A_p . Каждый последующий переход происходит под действием структурного сигнала $u_i(t)$, подаваемого на вход автомата A_i и определяемого с помощью функции входов μ_i этого автомата ($i = 1, \dots, p$). Структурные сигналы $u_i(t)$ естественным образом объединяются в структурный входной сигнал $u(t)$ памяти автомата A , а из функций μ_i строится объединяющая их (содержащая их в качестве своих компонент) векторная функция

$$\mu(v(t), v(t+1)) = u(t). \quad (2)$$

Подставляя в формулу (2) значение $v(t+1)$ из формулы (1), получим

$$u(t) = \mu(v(t), \delta(v(t), x(t))) = \zeta(v(t), x(t)). \quad (3)$$

Определяемая этой формулой векторная функция $\zeta(v, x)$ называется (векторной) **функцией возбуждения** автомата A или **функцией входов его памяти**. Она определяет, какой структурный входной сигнал $u(t)$ в любой момент времени t должен быть подан на запоминающие элементы памяти автомата A , находящегося в состоянии $v(t)$, если на внешние входные узлы автомата подается в тот же самый момент времени структурный входной сигнал $x(t)$.

Благодаря совпадению всех рассмотренных сигналов во времени сигнал $u(t)$ можно рассматривать как структурный выходной сигнал некоторой комбинационной схемы, отличающейся тем, что ее структурный входной сигнал получается в результате объединения структурного входного сигнала $x(t)$ автомата A и **структурного выходного сигнала $v(t)$ его памяти**. Реализуя эту схему в виде композиции заданных логических элементов, очевидны все те переходы, которые предусматриваются функцией переходов автомата A .

Построенная комбинационная схема называется **схемой обратных связей** автомата A , а уравнение

$$u = \zeta(v, x), \quad (4)$$

определяющее реализуемую этой схемой векторную функцию выходов, – каноническим (векторным) уравнением автомата A . При переходе к компонентам это уравнение распадается на **систему канонических уравнений** автомата A .

Заметим, что функция возбуждения $\zeta(v, x)$ автомата является неоднозначной функцией, поскольку неоднозначными, в общем случае, являются функции входов μ_i запоминающих элементов A_i ($i = 1, \dots, p$). Неоднозначностью этой функции можно

воспользоваться для получения возможно более простой схемы обратных связей в автомате.

Значительные возможности упрощения схем обратных связей и определяемых ниже схем выходов заключены в выборе рационального способа кодирования состояний синтезируемого автомата. В выборе рационального, с указанной точки зрения, кодирования состояний автомата состоит так называемая **проблема кодирования**, являющаяся одной из трудных проблем структурной теории автоматов. В настоящее время решение этой проблемы в каждом конкретном случае сопряжено, как правило, с перебором большого числа различных вариантов кодирования состояний.

4.4.3. Комбинационная часть автомата. Синтез схемы автомата

При построении схемы обратных связей необходимо учитывать то, что в каждый момент времени t в соответствии с законом функционирования автомата A обеспечивается образование структурного выходного сигнала $y(t)$. В зависимости от того, является ли заданный автомат A автоматом Мили или Мура, образование его выходного сигнала будет определяться либо законом $y(t) = \lambda(v(t), x(t))$, либо законом $y(t) = \lambda(v(t))$.

И в том, и в другом случае необходимый структурный выходной сигнал может быть обеспечен комбинационной схемой, которую также называют **схемой выходов** исходного автомата A . Схема, построенная на автоматах Мили, реализует векторную функцию $y(t) = \lambda(v, x)$, а на автоматах Мура – векторную функцию $y(t) = \lambda(v)$.

В обоих случаях схема выходов автомата может быть получена в результате композиции заданных логических элементов только на основе предположения о функциональной полноте системы этих элементов.

При структурном синтезе автомата обе построенные комбинационные схемы (схема обратных связей и схема выходов) объединяются в одну общую комбинационную схему, называемую комбинационной частью автомата, а запоминающая часть автомата, представляет собой объединение всех запоминающих элементов. С помощью такого объединения часто оказываются возможными дальнейшие упрощения схемы.

Структурная схема автомата, синтезированного в соответствии с каноническим методом структурного синтеза, имеет вид, изображенный на рис. 4.5.

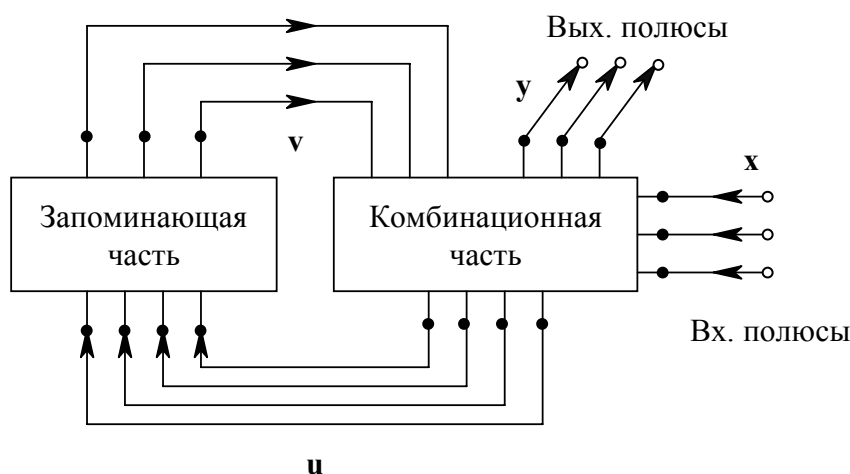


Рис. 4.5. Структурная схема автомата, синтезированного в соответствии с каноническим методом структурного синтеза

Очевидно, что эта структурная схема будет корректно построенной или правильной, если корректно построена или соответственно правильна его комбинационная часть (вместе с входными и выходными полюсами).

Таким образом, канонический метод синтеза полностью сводится к синтезу произвольных конечных автоматов, то есть к автоматам без памяти.

Применение канонического метода структурного синтеза рассмотрим на примере синтеза схемы автомата Мили А (в структурном алфавите (1, 2, 3)), заданного таблицами переходов и выходов 4.1 и 4.2.

Таблица 4.1

	q_1	q_2	q_3	q_4	q_5
1	q_2	q_1	-	q_3	-
2	q_5	q_2	q_1	q_5	-
3	q_4	-	q_3	-	q_4

Таблица 4.2

	q_1	q_2	q_3	q_4	q_5
1	(1,1)	(2,2)	-	(1,3)	-
2	(3,1)	(3,3)	(2,1)	(3,2)	-
3	(3,3)	(1,1)	(2,2)	-	(2,3)

В качестве элемента памяти выберем автомат Мура В, задаваемый таблицей переходов 4.3.

Таблица 4.3

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

Отметим, что структурные состояния i в автомате В есть также i ($i = 1, 2, 3$), тогда получим автомат Мура с полной системой переходов и с полной системой выходов.

Для кодирования состояний автомата А в структурном алфавите достаточно взять два элемента памяти, поскольку $3^2 > 5$. Систему кодирования автомата А выбираем произвольным образом, например, так:

$$q_1 = (1, 1), \quad q_2 = (2, 2), \quad q_3 = (3, 3), \quad q_4 = (1, 2), \quad q_5 = (1, 3).$$

После этого таблицу переходов (табл. 4.1) автомата А представим в виде таблицы 4.4 и выпишем таблицу входов элемента памяти В (табл. 4.5).

Таблица 4.4

	(1, 1)	(2, 2)	(3, 3)	(1, 2)	(1, 3)
1	(2, 2)	(1, 1)	-	(3, 3)	-
2	(1, 3)	(2, 2)	(1, 1)	(1, 3)	-
3	(1, 2)	-	(3, 3)	-	(1, 2)

Таблица 4.5

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

Используя таблицу 4.5 и таблицу переходов автомата А (табл. 4.1), построим векторную функцию возбуждения автомата А. Обозначим через $x = x$ внешний структурный входной сигнал автомата А; через $v = (v_1, v_2)$ – структурный выходной сигнал этого автомата, а через $u = (u_1, u_2)$ – структурный входной сигнал его памяти. Для за-

дания функции возбуждения автомата A нужно задать зависимость входных сигналов u_1 и u_2 от сигналов x , v_1 , v_2 .

Функции возбуждения задаются в виде таблицы, в которую явно выписываются значения компонент функции возбуждения на всех наборах элементарных сигналов, где эта функция задана. Таблицу для функции возбуждения целесообразно объединять с таблицей для векторной функции выходов автомата, задающей компоненты внешнего структурного выходного сигнала $y = (y_1, y_2)$ как функции элементарных сигналов x , v_1 , v_2 . Наборы элементарных сигналов x , v_1 , v_2 , на которых не определены ни функция возбуждения, ни функция выходов в таблицу не включаются.

Получаемую таким образом объединенную таблицу для функций возбуждения и выходов автомата условимся называть **функциональной таблицей** данного автомата. Она является исходной для синтеза соответствующей комбинационной схемы.

В рассматриваемом примере из таблиц переходов и выходов автомата A (табл. 4.1 и 4.2) и таблицы входов элемента памяти B (табл. 4.5) получаем функциональную таблицу автомата A (табл. 4.6).

Таблица 4.6

x	v_1	v_2	u_1	u_2	y_1	y_2
1	1	1	2	2	1	1
1	2	2	3	3	2	2
1	1	2	3	2	1	3
2	1	1	1	3	3	1
2	2	2	1	1	3	3
2	3	3	2	2	2	1
2	1	2	1	2	3	2
3	1	1	1	2	3	3
3	2	2	-	-	1	1
3	3	3	1	1	2	2
3	1	3	1	3	2	3

После построения этой таблицы задача структурного синтеза автомата A оказалась сведенной к задаче синтеза комбинационной схемы с тремя входными полюсами (x , v_1 , v_2) и четырьмя выходными полюсами (u_1 , u_2 , y_1 , y_2).

4.5. Кодирование состояний. Гонки в автомате

Кодирование состояний является одной из основных задач канонического метода структурного синтеза автоматов. Кодирование заключается в установлении взаимно-однозначного соответствия между множеством $Q = \{q_1, \dots, q_m\}$ состояний автомата и множеством R -компонентных векторов $\{K_1, \dots, K_m\}$, $K_m = (\ell_{m1}, \dots, \ell_{mR})$, где ℓ_{mR} – состояние r -го элемента памяти, $r = 1, \dots, R$. Если $\ell_{mR} \in \{0, 1\}$, т. е. алфавит состояний элементов памяти двоичный, тогда в качестве элементов памяти применим RS-триггеры, которые будем обозначать T_1, \dots, T_R .

Переход автомата из одного состояния в другое осуществляется за счет изменения состояний элементов памяти. Так, если автомат переходит из состояния q_m с кодом 0101 в состояние q_s с кодом 1001, то это означает, что триггер T_1 переходит из

состояния 0 в состояние 1, триггер T_2 – из состояния 1 в состояние 0, а состояния триггеров T_3 и T_4 не изменяются.

При функционировании автомата могут возникать так называемые состязания. Явление состязаний возникает вследствие того, что элементы памяти имеют различные времена срабатывания. Кроме того, различны также задержки сигналов возбуждения, поступающих на входные каналы элементарных автоматов по логическим цепям неодинаковой длины. Если при переходе автомата из одного состояния в другое должны изменить свои состояния сразу несколько запоминающих элементов, то между ними начинаются состязания. Тот элемент, который выиграет эти состязания, т. е. изменит свое состояние ранее, чем другие элементы, может через цепь обратной связи изменить сигналы на входах некоторых запоминающих элементов до того, как другие участвующие в состязаниях элементы изменят свои состояния. Это может привести к переходу автомата в состояние, не предусмотренное графом. Поэтому в процессе перехода из состояния q_m в состояние q_s под действием входного сигнала x_f (рис. 4.6) автомат может оказаться в некотором промежуточном состоянии q_k или q_ℓ в зависимости от того, какой элемент памяти выиграет состязание. Если затем при том же входном сигнале автомат из q_k или q_ℓ перейдет в состояние q_s , то такие состязания являются допустимыми или некритическими. Если же в этом автомате переход, например, из q_k в $q_j \neq q_s$, под действием того же сигнала x_f (рис. 4.7), то автомат может перейти в q_j , а не в q_s и правильность его работы будет нарушена. Такие состязания называются **критическими** или **гонками**.

При кодировании состояний гонки должны быть устранены. Кодирование с устранением гонок называется **противогоночным**.

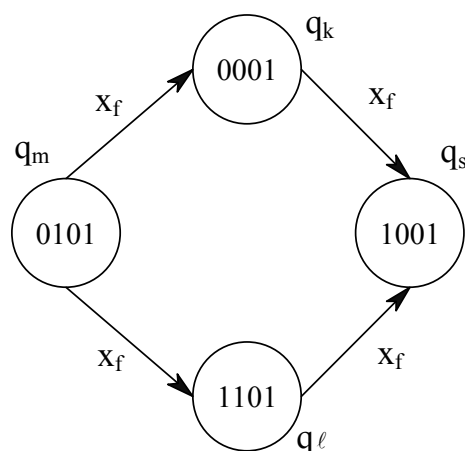


Рис. 4.6. Граф переходов автомата

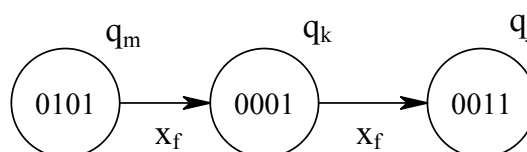


Рис. 4.7. Граф переходов автомата при наличии «гонок»

Один из способов ликвидации гонок состоит в тактировании входных сигналов автомата импульсами определенной длительности. Предполагается, что кроме входных каналов x_1, \dots, x_L имеется еще один канал p от генератора синхроимпульсов (ГСИ), по которому поступает сигнал $p = 1$ в момент прихода импульса и $p = 0$ при его отсутствии. В связи с этим входным сигналом на переходе (q_m, q_s) будет не x_f , а px_f . Тогда, если длительность импульса t_p меньше самого короткого интервала времени прохождения тактированного сигнала обратной связи по комбинационной схеме, то к моменту перехода в промежуточное состояние q_k (рис. 4.7) сигнал $p = 0$ и, следовательно, $px_f = 0$, что и исключает гонки.

Другой способ ликвидации гонок заключается во введении двойной памяти (рис. 4.8).

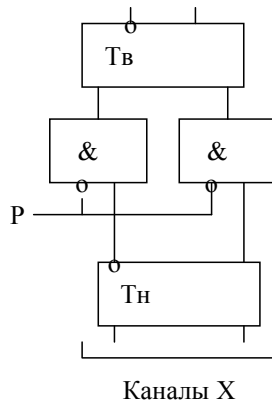


Рис. 4.8. Исключение «гонок» в автомате

В этом случае каждый элемент памяти дублируется, причем запись из нижнего элемента памяти в верхний происходит в момент отсутствия тактирующего импульса ($p = 0$). Сигналы обратной связи для получения функций возбуждения и функций выходов автомата снимаются с верхнего ряда триггеров. Таким образом, состязания могут возникать только между нижними триггерами и, пока p не станет равным нулю, сигналы обратной связи не изменятся. Тогда и входной сигнал rx_f также равен нулю, то есть гонки быть не может.

Наряду с аппаратными способами для устранения гонок используются специальные методы кодирования (**противогоночное кодирование**).

Пусть (α, β) и (γ, δ) — две пары двоичных кодов длины R . Пары (α, β) и (γ, δ) называются **развязанными**, если при некотором $1 \leq r \leq R$ r -й разряд кода принимает одно значение на паре (α, β) и противоположное на паре (γ, δ) . В противоположном случае пары двоичных кодов называются **связанными**.

Теорема (без доказательства) [1]. В автомате, состояния которого закодированы двоичными кодами конечной длины, гонки отсутствуют тогда и только тогда, когда для любых двух переходов (q_m, q_s) и (q_k, q_ℓ) , $q_s \neq q_\ell$, происходящих под действием одного и того же входного сигнала, соответствующие им пары кодов состояний развязаны.

Существует один частный способ кодирования — **соседнее кодирование** состояний автомата, при котором условие отсутствия гонок всегда выполнено.

При соседнем кодировании любые два состояния связанные дугой на графе автомата, кодируются наборами, отличающимися состояниями лишь одного элемента памяти.

Таким образом, имеются четыре способа устранения гонок: 1) двойная память; 2) рациональный выбор длительности синхроимпульса; 3) развязывание пар переходов; 4) соседнее кодирование.

4.6. Построение комбинационной схемы автомата

Пусть необходимо синтезировать частичный С-автомат A , заданный таблицей переходов (табл. 4.7) и отмеченный таблицей выходов (табл. 4.8).

Таблица 4.7

 $\delta: Q \times X \rightarrow Q$

	q_1	q_2	q_3
x_1	q_2	-	q_1
x_2	q_3	q_1	-
x_3	q_2	q_3	q_3

Таблица 4.8

 $\lambda_1: Q \times X \rightarrow Y$ $\lambda_2: Q \rightarrow U$

	u_1	u_2	u_1
	q_1	q_2	q_3
x_1	y_3	-	y_2
x_2	y_4	y_3	-
x_3	y_2	y_1	y_3

В качестве элементарных автоматов используем логические элементы И, ИЛИ, НЕ и автомат памяти П, функция переходов которого задана табл. 4.9.

Таблица 4.9

 $B \times Z \rightarrow B$

	b_1	b_2
z_1	b_1	b_2
z_2	b_2	b_1

Структурный входной и выходной алфавит состояний синтезируемого автомата и автомата памяти будем считать двоичными.

Перейдем к структурному представлению автоматов А и П, для чего закодируем их входные и выходные сигналы и состояния.

Так как в абстрактном автомате П два входных и два выходных сигнала (табл. 4.9), то в структурном автомате П будет один входной (ϕ) и один выходной (τ) каналы. После кодирования входных сигналов (табл. 4.10) и состояний (табл. 4.11) абстрактного автомата П, получим его таблицу переходов, представленную в табл. 4.12.

Таблица 4.10

 $Q \rightarrow \phi$

Q	ϕ
q_1	0
q_2	1

Таблица 4.11

 $B \rightarrow \tau$

q_3	τ
b_1	0
b_2	1

Таблица 4.12

 $\tau \times \phi \rightarrow \tau$

	0	1
0	0	1
1	1	0

Таблица 4.13

$\tau_{\text{исх}}$	ϕ	$\tau_{\text{пер}}$
0	0	0
0	1	1
1	1	0
1	0	1

Функция входов структурного автомата П приведена в таблице 4.13. Так как у абстрактного автомата А три состояния, то структурный автомат А будет иметь два элемента памяти Π_1 и Π_2 . Три абстрактных входных (x_1, x_2, x_3) и четыре выходных сигнала (y_1, y_2, y_3, y_4) потребуют два входных (q_1, q_2) и два выходных канала (ω_1 и ω_2). Необходим также еще один выходной канал (τ), чтобы реализовать получение двух выходных сигналов (u_1, u_2). В таблице 4.14 приведены результаты кодирования со-

стояний автомата А, в таблице 4.15 – результаты кодирования входных сигналов автомата А, а в таблицах 4.16 и 4.17 – кодирование выходных сигналов автомата А.

Таблица 4.14

Q	τ_1	τ_2
q_1	0	0
q_2	0	1
q_3	1	0

Таблица 4.15

X	q_1	q_2
x_1	0	0
x_2	0	1
x_3	1	0

Таблица 4.16

Y	ω_1	ω_2
y_1	1	0
y_2	0	0
y_3	1	1
y_4	0	1

Таблица 4.17

U	R
u_1	1
u_2	0

Заменяв в таблице 4.7 и таблице 4.8 состояния и соответствующие сигналы их кодами, получим таблицу переходов (табл. 4.18) и отмеченную таблицу выходов (табл. 4.19) структурного автомата А. Структурная схема данного автомата приведена на рис. 4.9.

Таблица 4.18

	00	01	11
00	01	-	00
01	11	00	-
10	01	11	11

Таблица 4.19

	1	0	1
	00	01	11
00	11	-	00
01	01	11	-
10	00	10	11

Таким образом, после выбора элементов памяти и кодирования состояний синтез С-автомата сводится к синтезу двух комбинационных схем КС1 и КС2, реализующих функции:

$$\begin{aligned} \omega_1(\tau_1, \tau_2, a_1, a_2), \quad \omega_2(\tau_1, \tau_2, a_1, a_2), \\ \varphi_1(\tau_1, \tau_2, a_1, a_2), \quad \varphi_2(\tau_1, \tau_2, a_1, a_2), \quad r(\tau_1, \tau_2). \end{aligned}$$

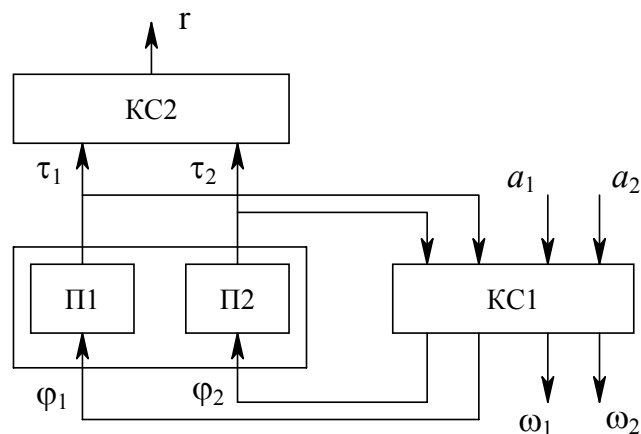


Рис. 4.9. Структурная схема автомата

В дальнейшем, непосредственно из таблицы 4.2.2 могут быть получены аналитические выражения ω_1 и ω_2 как дизъюнкции конъюнкций, соответствующих наборам переменных τ_1, τ_2, a_1, a_2 , на которых эти функции принимают значение единицы:

$$\begin{aligned} \omega_1 &= \tau_1 \tau_2 a_1 a_2 \vee \tau_1 \tau_2 a_1 a_2 \vee \tau_1 \tau_2 a_1 a_2 \vee \tau_1 \tau_2 a_1 a_2, \\ \omega_2 &= \tau_1 \tau_2 a_1 a_2 \vee \tau_1 \tau_2 a_1 a_2 \vee \tau_1 \tau_2 a_1 a_2 \vee \tau_1 \tau_2 a_1 a_2. \end{aligned} \quad (5)$$

Если каждый данный набор отождествить с его номером (например, набору 0001 соответствует 1, а набору 1110 – 14), то выражение (5) можно представить в виде

$$\omega_1 = 0 \vee 5 \vee 6 \vee 14, \quad \omega_2 = 0 \vee 1 \vee 5 \vee 14. \quad (6)$$

Также непосредственно из табл. 4.22 получим выражение для функции $r = r(\tau_1, \tau_2)$

$$r = \tau_1 \cdot \tau_2 \vee \tau_1 \tau_2. \quad (7)$$

Несколько сложнее получаются выражения функций возбуждения памяти. Рассмотрим, например, что будет с автоматом А, если он находился в состоянии 01, и на его вход поступил входной сигнал 10. Как видно из таблицы 4.18 (второй столбец, третья строка), автомат А из состояния 01 перейдет в состояние 11. Этот переход складывается из двух переходов элементарных автоматов памяти. Первый из них (Π_1) перейдет из состояния 0 в состояние 1, а второй (Π_2) – из состояния 1 в состояние 1 (останется в том же состоянии).

Переходы автоматов памяти происходят под действием сигналов функций возбуждения, поступающих на их входы. Для определения того, что нужно подать на вход автомата Π_1 , чтобы перевести его из 0 в 1, обратимся к функции входов автомата памяти (табл. 4.13). Как видно из нее (вторая строка) на вход автомата Π_1 необходимо подать сигнал 1. Аналогично, для перехода автомата Π_2 из 1 в 1 на его вход должен быть подан сигнал 0 (четвертая строка). Таким образом, при переходе автомата А из состояния 01 в состояние 11 на входы его элементов памяти должен поступить векторный сигнал функции возбуждения 10. Занесем этот результат на пересечение второго столбца и третьей строки таблицы функции возбуждения (табл. 4.20). Аналогичным образом, используя значения остальных переходов (табл. 4.21), заполним таблицу функций возбуждения памяти автомата А. Поскольку функция возбуждения памяти автомата зависит от тех же переменных τ_1, τ_2, a_1, a_2 , столбцы и строки табл. 4.20 и 4.18 отмечены одинаково.

Таблица 4.20

	00	01	11
00	01	-	11
01	11	01	-
10	01	10	00

Непосредственно из табл. 4.20 для единичных значений функций φ_1 и φ_2 имеем:

$$\begin{aligned} \varphi_1 &= \tau_1 \tau_2 \bar{a}_1 a_2 \vee \tau_1 \tau_2 a_1 \bar{a}_2 \vee \tau_1 \tau_2 \bar{a}_1 \bar{a}_2 = 1 \vee 6 \vee 12, \\ \varphi_2 &= \tau_1 \tau_2 \bar{a}_1 \bar{a}_2 \vee \tau_1 \tau_2 \bar{a}_1 a_2 \vee \tau_1 \tau_2 a_1 \bar{a}_2 \vee \tau_1 \tau_2 a_1 a_2 = 0 \vee 1 \vee 2 \vee 5 \vee 12. \end{aligned} \quad (8)$$

По выражениям (5), (7), (8) построим логическую схему (рис. 4.10).

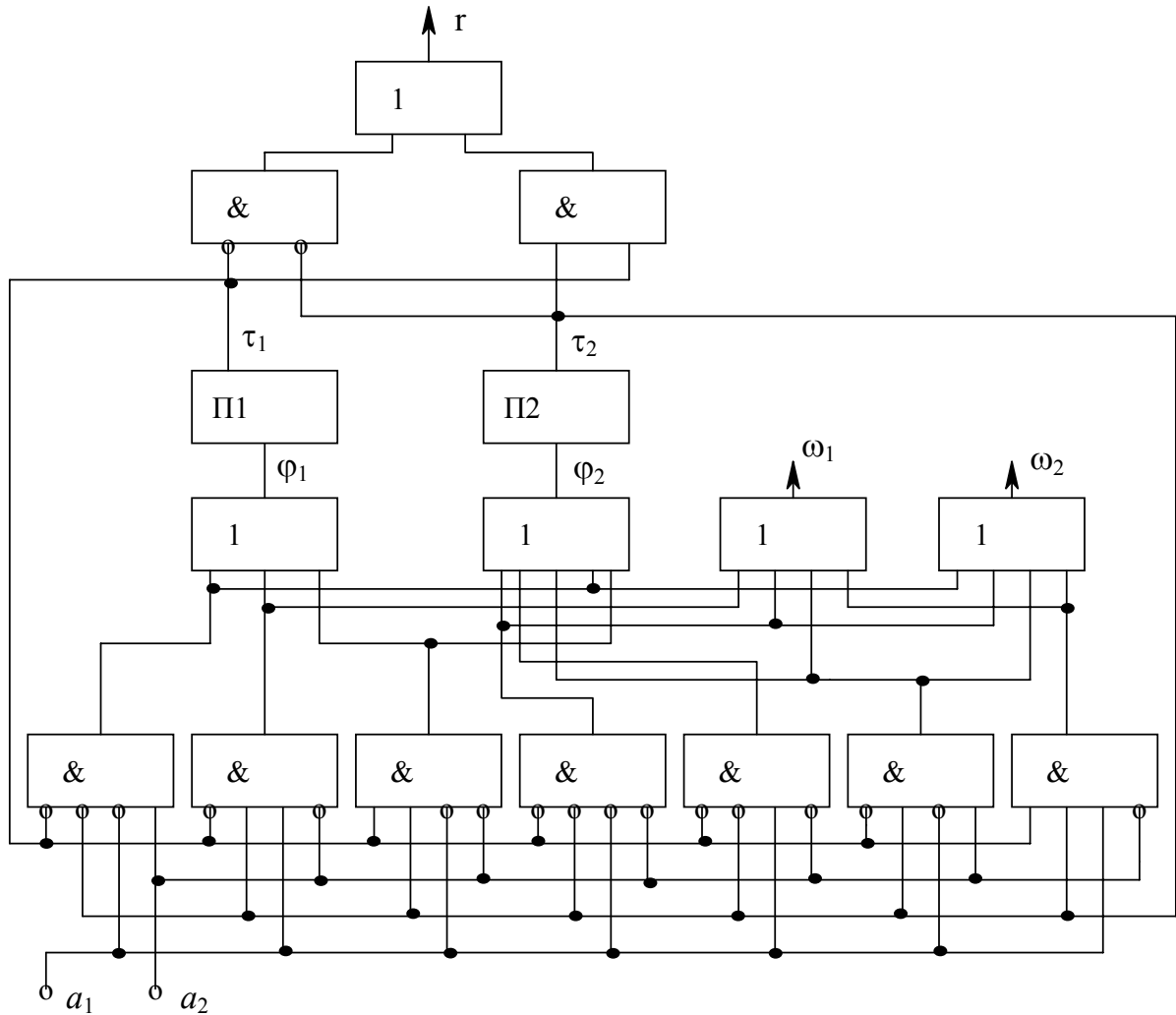


Рис. 4.10. Логическая схема автомата

Контрольные вопросы

1. Основные понятия структурной теории автоматов.
2. Композиция автоматов, структурные схемы.
3. Условия корректности и правильности построения схем.
4. Основные задачи теории структурного синтеза автоматов.
5. Теорема о структурной полноте.
6. Комбинационная часть автомата.
7. Кодирование состояний автомата.
8. Состояние элементов памяти.
9. Явление риска логических схем.

5. МИКРОПРОГРАММИРОВАНИЕ

Рассмотренные методы синтеза схем автоматов с памятью оказываются пригодными в основном для синтеза автоматов с относительно небольшим числом состояний. В случае синтеза сложных автоматов применению этих методов предшест-

вует обычно еще один этап, названный *этапом блочного синтеза*. Сущность этого этапа заключается в том, что на основе алгоритма функционирования, схема автомата разбивается на ряд отдельных участков, называемых блоками, и производится определение условий работы этих блоков. Одновременно с этим обычно производится циклирование работы отдельных блоков.

При синтезе очень сложных автоматов этап блочного синтеза может быть многоступенчатым. Общих методов блочного синтеза не существует.

Рассмотрим процесс блочного синтеза применительно лишь к одному классу автоматов, наиболее широко распространенному в настоящее время, а именно к классу универсальных микропрограммных автоматов (универсальных электронных вычислительных машин с программным управлением).

5.1. Принципы микропрограммного управления

Рассмотрим автоматы, в основу которых положен так называемый принцип микропрограммного управления, использующий операционно-адресную организацию управления алгоритмическим процессом.

Суть этого принципа заключается в следующем. Информация, с которой оперирует автомат, разделяется на две части:

- собственно на входную информацию;
- на информацию об алгоритме, который должен быть реализован автоматом.

Информация об алгоритме представляется программой, которая состоит из отдельных команд. Хотя всю информацию, воспринимаемую автоматом, можно считать одним словом, при командно-адресной организации управления эту информацию делят на более мелкие слова. В соответствии с принадлежностью информации к одному из двух описанных выше видов различают *информационные слова* (называемые в дальнейшем просто *словами* или *числами*) и *программные слова* (называемые обычно *командами* или *приказами*).

Чаще всего все слова (все команды) в автомате имеют одинаковую длину: обычно стремятся к тому, чтобы эта длина была одинаковой для информационных слов и команд. Каждое слово (информационное и программное) хранится в определенной части *памяти* (т. е. в определенной группе запоминающих элементов) автомата, называемой *ячейкой памяти*. Ячейки памяти, а следовательно и содержащиеся в них слова, нумеруются натуральными числами, называемыми *адресами* этих ячеек.

Каждая команда осуществляет элементарное преобразование информации, называемое операцией. Команда указывает операцию, которую необходимо выполнить, а объекты этой операции – операнды. Операция выполняется над одним или несколькими словами (информационными или программными), задаваемыми соответствующими адресами.

В соответствии с этим команда (программное слово) состоит из двух частей – *операционной* и *адресной*.

В операционной части содержится код операции (условный номер), указывающий на выполняемое данной командой действие, в адресной части – адреса (ячейки памяти) слов (операндов), над которыми должна выполняться данная операция.

Последовательность микрокоманд, выполняющих одну команду, образует микропрограмму. Обычно микропрограммы хранятся в специальной памяти микропрограмм («управляющей памяти»).

В управляющих автоматах с хранимой в памяти программой микропрограммы используются в явной форме, они программируются в кодах микрокоманд и в таком виде заносятся в память. Поэтому такой метод управления любым цифровым устройством называется микропрограммированием, а использующие этот метод управляющие блоки – микропрограммными управляющими устройствами.

Различают одноадресные, двухадресные, трехадресные и четырехадресные команды.

Например, при построении автоматов, реализующих вычислительные алгоритмы, к наиболее употребительным типам операций относят арифметические операции: сложение, вычитание, умножение, деление. Каждая из таких операций выполняется над двумя информационными словами (числами, операндами), а ее результатом является третье информационное слово. Для указания ячеек, в которых должны храниться эти слова, необходимо иметь три адреса, то есть целесообразно использовать трехадресные команды из соответствующей системы команд.

В наиболее сложных программах необходимо предусмотреть определенный порядок следования команд. Различают два вида порядка следования команд: естественный и принудительный. При естественном порядке после выполнения каждой очередной команды выполняется команда, расположенная в следующей по порядку ячейке памяти. В случае принудительного порядка осуществляется переход к следующей команде, по заданному адресу, указанному в адресной части текущей (выполняемой) команды.

5.2. Система команд автоматов, реализующих выполнение алгоритма

Обычный рабочий цикл программного автомата, выполняющего трех- или четырехадресные команды, состоит из следующих шагов:

- 1) выборка из памяти первого информационного слова А (по первому адресу команды);
- 2) выборка из памяти («чтение») второго информационного слова В (по второму адресу команды);
- 3) выполнение операции над выбранными словами А и В в соответствии с кодом операции выполняемой команды и получения результата операции, – некоторого слова С;
- 4) запись результата (т. е. слова С) в память (по третьему адресу команды);
- 5) выборка из памяти следующей команды (в случае трехадресной системы эта выборка производится из следующей по порядку ячейки памяти, в случае четырехадресной системы – из той ячейки памяти, которая указана в четвертом адресе выполняемой команды).

В ряде случаев удобно считать, что рабочий цикл автомата начинается с выборки из памяти той команды, которая должна выполняться на последующих шагах цикла.

Рабочий цикл автоматов в случае команд малой адресности (т. е. одноадресных или двухадресных) занимает лишь некоторую часть описанного цикла.

Так, в случае одноадресной системы команд с естественным порядком их следования в течение одного обычного рабочего цикла выполняется либо чтение, либо запись (в соответствии с кодом операции) информационного слова в памяти автомата (по адресу, указанному в команде) и выборке следующей по порядку команды. Описанный выше нормальный трехадресный цикл может быть выполнен, очевидно, в течение трех одноадресных циклов.

5.3. Набор операций автомата

Важной особенностью современных универсальных программных автоматов является наличие у них таких операций, которые позволяют изменять адресную часть команд программы, то есть менять порядок следования команд в зависимости от результатов, полученных после выполнения операций над информационными словами. Выполнение операций первого вида, называемых *операциями переадресации*, приводит к выполнению операций второго вида – *операций условного перехода*. Операцию переадресации можно задать трехадресной командой, в первом адресе которой указывается адрес переадресуемой команды, во втором – константа переадресации, (адрес переадресуемой команды), а на месте третьего адреса записывается адрес ячейки, указывающий на переадресованную команду. На практике удобно пользоваться различными константами переадресации, однако, в данном случае достаточно рассматривать переадресации лишь на ± 1 .

Операция условного перехода задается трехадресной командой. При выполнении этой команды, слово, выбранное по первому адресу команды, сравнивается с выбранным словом второго адреса. В случае несовпадения выбранных слов выбирается следующая по порядку команда, а в случае совпадения – команда, находящаяся в ячейке по адресу, указанному в третьем слове трехадресной команды условного перехода. Определенную таким образом операцию называют *операцией условного перехода по точному совпадению слов*.

Для построения универсальных программных автоматов большое значение имеют еще три операции, а именно операции пересылки, ввода и вывода.

Операция пересылки может быть реализована в виде двухадресной команды, при выполнении которой слово, выбранное из ячейки первого адреса команды, пересылается в ячейку по второму адресу команды. При использовании команд с большим числом адресов остальные адреса не учитываются.

Операция ввода. При выполнении этой операции информация (в виде последовательности слов), подаваемая на специальное *вводное устройство*, записывается в последовательно расположенные ячейки памяти (начиная с некоторой заданной ячейки). Обычно оказывается целесообразным использовать адресную часть команды для указания ячейки, в которую заносится первое вводимое слово.

Операция вывода состоит в выводе из автомата через специальное выводное устройство последовательно, слово за словом, содержимого всех ячеек памяти авто-

мата, адреса которых заданы в интервале чисел **a** и **b**. Эти числа указываются в двух адресах команды, реализующей операцию вывода.

5.4. Состав и назначение элементов блок-схемы

Блок-схема универсального программного автомата для реализации арифметических и логических операций может состоять из пяти основных блоков (рис. 5.1):

- запоминающего устройства (ЗУ);
- арифметико-логического устройства (АЛУ);
- устройства управления (УУ);
- вводного устройства (ввод);
- выводного устройства (вывод).

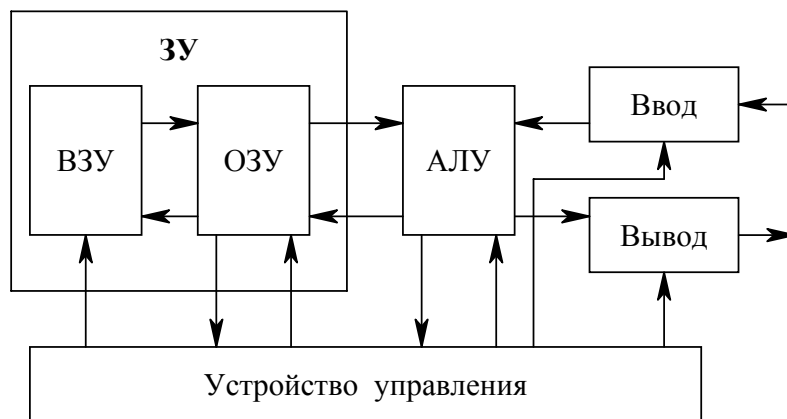


Рис. 5.1. Блоки универсального программного аппарата

Рассмотрим особенности построения таких универсальных программных автоматов.

При необходимости память универсальных автоматов включает два блока: оперативное запоминающее устройство (ОЗУ) и внешнее запоминающее устройство (ВЗУ).

Каждый из указанных блоков имеет свое особое назначение и выполняет определенные функции в работе автомата.

ОЗУ – предназначено для запоминания (записи) и хранения информации (информационных и программных слов) и характеризуется возможностью быстрого (оперативного) извлечения (чтения) записанной информации и пересылки ее в другие блоки в соответствии с сигналами, поступающими из устройства управления (УУ).

ВЗУ – предназначено для относительно длительного хранения информации и передачи в ОЗУ той информации, которая требуется в соответствии с сигналами, поступающими из УУ. Обычно обмен информацией между ОЗУ и ВЗУ осуществляется не посредством пересылки однозначных слов, а посредством передачи целых групп слов, называемых *массивами*.

АЛУ – предназначено для выполнения арифметических и логических операций над информационными словами, поступающими в него из ОЗУ по сигналам, выраба-

тываемым УУ. Результаты этих операций с помощью других сигналов УУ передаются на хранение в ОЗУ.

Устройство ввода осуществляет (по соответствующему сигналу УУ) ввод информации (информационных и программных слов), хранимой вне автомата на специальных накопителях. Данная информация поступает в ОЗУ либо непосредственно, либо после предварительного прохождения через АЛУ.

Устройство вывода служит для вывода информации на различные периферийные устройства.

УУ – координирует действия всех остальных устройств автомата.

В синхронных автоматах в состав устройства управления входит тактирующий генератор, задающий (с какой-либо постоянной частотой) моменты изменения состояний отдельных блоков автомата. Частота сигналов, выдаваемых этим генератором, называется *рабочей частотой* универсального автомата. В современных быстродействующих универсальных автоматах эта частота измеряется сотнями МГц.

5.5. Общий алгоритм функционирования

Обозначая направление передачи информации от одного блока к другому с помощью стрелок, можно представить блок-схему универсального программного автомата так, как это было изображено на рис. 5.1. Стрелки, показанные на этом рисунке от УУ к другим блокам, означают направление передачи соответствующих управляющих сигналов. Стрелка из ОЗУ в УУ соответствует передаче команд от ОЗУ к УУ, где они подвергаются расшифровке и исполнению.

Отметим, что хотя каждый канал передачи информации от одного блока к другому обозначен только одной линией, в действительности это может быть много линий, образующих ряд параллельных каналов.

Возможны два различных способа передачи информационных и программных слов между блоками автомата. Один способ позволяет передавать каждый разряд кода слова с использованием отдельного физического канала, называемого обычно шиной, так что передача всех разрядов кода производится одновременно.

При другом способе передачи слов все разряды кода передаются по одному и тому же физическому каналу (шине) последовательно, один за другим.

Программные автоматы, в которых осуществлен первый способ передачи кодов, называются параллельными, а автоматы, в которых имеет место второй способ передачи кодов, называются последовательными. Возможны, разумеется, и смешанные (параллельно-последовательные) системы передачи кодов в автомате.

5.6. Основные характеристики автоматов

Скорость работы универсальных программных автоматов измеряется обычно числом рабочих циклов, выполняемых автоматом в течение одной секунды. Часто различные рабочие циклы (например, циклы, соответствующие командам сложения и умножения) имеют различную длительность, поэтому при определении быстродействия автомата прибегают к подсчету среднего числа циклов (команд), выполняемых им в единицу времени. При этом в большинстве случаев не учитывают команды ввода, вывода и обмена информацией между ОЗУ и ВЗУ, так как предполагается, что машина (автомат) работает все время с ОЗУ.

Быстродействие универсального автомата, подсчитанное при условии работы автомата с ОЗУ и выраженное средним числом операций (команд), выполняемых автоматом в течение одной секунды, будем называть **номинальным быстродействием** этого автомата.

Номинальное быстродействие не определяет, как правило, реального времени, которое необходимо затратить, решая с помощью универсального автомата ту или иную задачу. Для подсчета реального времени решения задачи нужно учитывать время, затрачиваемое на ввод и на вывод информации, потери времени за счет обращения к ВЗУ (работающему значительно медленнее, чем ОЗУ), потери времени на многократное повторение решения для получения ответа с заданной степенью надежности и, наконец, средние потери времени за счет таких факторов, как профилактический ремонт и устранение возникающих в процессе работы неисправностей.

Также необходимо учитывать различную сложность операций, выполняемых автоматом. Устанавливая те или иные весовые коэффициенты для различных операций, получим возможность выражать все операции через какую-нибудь одну операцию (например, сложение), принимаемую в качестве **стандартной операции**.

Быстродействие универсального автомата, выраженное числом выполняемых им в единицу времени (секунду) стандартных операций будем называть **эффективным быстродействием** этого автомата.

Очевидно, что определенное таким образом эффективное быстродействие зависит от задачи, которая решается автоматом. Для определения среднего эффективного быстродействия автомата можно воспользоваться следующим приемом. Прежде всего выделяются: конечное множество M типовых задач R_1, \dots, R_n , решаемых автоматом A , эффективное быстродействие V_1, V_2, \dots, V_n автомата по каждой из этих задач и вероятности P_1, \dots, P_n введения в автомат каждой из данных задач R_1, \dots, R_n , (в процентном отношении времени, занимаемого каждой из них, от общего времени работы автомата A). Далее предполагается, что $P_1 + P_2 + \dots + P_n = 1$, тогда среднее эффективное быстродействие V автомата A на множестве задач M определяется по формуле:

$$\frac{1}{V} = \frac{P_1}{V_1} + \frac{P_2}{V_2} + \dots + \frac{P_n}{V_n}.$$

Среднее эффективное быстродействие представляет собою универсальный критерий эффективности программного автомата, ибо он определяет фактическую его производительность и характеризует усредненным образом все блоки автомата.

5.7. Устройство управления микропрограммным автоматом

Задачей устройства управления является, во-первых, управление последовательностью микроопераций в АЛУ и в ОЗУ, во-вторых, управление последовательностью собственных микроопераций. Для установления набора микроопераций УУ зафиксируем, прежде всего, количество и характер работы используемых в нем регистров.

Важнейшим из регистров УУ (и всего автомата в целом) является так называемый регистр команд (РК). Для трехадресных команд этот регистр «условно» можно разделить на четыре регистра:

- регистр операций (РО), регистрирующий (запоминающий) код операции выполняемой команды;
- регистры первого, второго и третьего адреса (PA_1 , PA_2 , PA_3), которые позволяют запомнить соответственно первый, второй и третий адреса, необходимые выполняемой команде.

Другой важной составной частью УУ является *счетчик команд* (СК), предназначенный для хранения адреса ячейки ОЗУ, из которой должна извлекаться очередная команда программы. Наконец, имеется еще *регистр микроопераций* (РМО), заменяемый иногда *счетчиком микротактов* (СМТ).

Счетчик микротактов реализуется всегда как циклический (замкнутый) счетчик, что же касается счетчика команд, то он может быть и незамкнутым.

Как следует из общего принципа программного управления, чтобы обеспечить управление работой всех устройств (включая и само устройство), УУ должно осуществлять автоматическую выборку команд из ОЗУ, а также их расшифровку и выполнение. С этой целью в набор микроопераций УУ должны быть включены следующие микрооперации:

- 1) передача кода (программного слова) из регистра числа ОЗУ на регистр команд;
- 2) передача первого адреса из РА в регистр адреса ОЗУ;
- 3) передача второго адреса из РА в регистр адреса ОЗУ;
- 4) передача третьего адреса из РА в регистр адреса ОЗУ;
- 5) посылка импульса в счетчик команд (увеличение номера команды на 1);
- 6) очистка регистра команд и регистра микроопераций;
- 7) передача третьего адреса из РА в счетчик команд;
- 8) передача содержимого счетчика команд в регистр адреса ОЗУ.

Восьмая микрооперация дает возможность реализовать операцию условного перехода.

Для организации управления последовательностью микроопераций строится конечный автомат М с регистровой памятью в виде регистра микроопераций. Выходные сигналы этого автомата представляют собой импульсы управления микрооперациями, посылаемые по одному или одновременно по нескольким из каналов управления микрооперациями (с изменением набора микроопераций меняется вообще говоря, и число этих каналов).

На вход автомата М поступают входные сигналы (сигналы обратной связи, подаваемые из АЛУ), представляющие собой выходные сигналы триггеров регистра операций (*операционные входы*). Кроме того, имеется один импульсный тактирующий вход, задающий *тактировку* (разбиение времени на микротакты) автомата М и всего универсального программного автомата в целом.

Для синхронного автомата импульсные тактирующие сигналы вырабатываются специальным синхронизирующим генератором, работающим с постоянной частотой.

Можно построить автомат М как асинхронный автомат, тогда после выдачи выходного сигнала для выполнения очередной микрооперации автомат переходит в состояние ожидания ответного сигнала от соответствующего устройства, свидетельствующего об окончании выполнения этой микрооперации. Ответные сигналы и будут играть роль тактирующего сигнала. Длительность микротактов при этом будет неодинаковой.

Задача организации управления микрооперациями любого универсального программного автомата Q есть задача синтеза соответствующего автомата M , который будет называться микропрограммным блоком.

Синтез микропрограммного блока в общем случае осуществляется с помощью методов синтеза автоматов, рассмотренных ранее.

5.8. Формирование адреса микрокоманд

В автоматах с программируемой логикой микрокоманды содержат адресную часть, позволяющую определять номера ячеек ПЗУ, к которым обращаются адресуемые микрокоманды. Способ адресации микрокоманд задает правило определения адреса следующей микрокоманды. Используются два основных способа адресации: принудительная и естественная адресация.

Принудительная адресация сводится к указанию в каждой микрокоманде адреса следующей микрокоманды. Этот способ реализован в управляющем автомате (рис. 5.2), который работает по микропрограмме, представляющей собой последовательность микрокоманд. Структура микрокоманды (МК) представлена на рис. 5.3.

Для хранения микрокоманд используются ПЗУ емкостью $(P+1)$ k -разрядных слов. Управляющий сигнал ЧТ инициирует операцию чтения слова $МК := +ПЗУ[A]$, в результате выполнения которой из ячейки ПЗУ с адресом A в регистр МК считывается микрокоманда. Выбранная из ПЗУ микрокоманда обрабатывается следующим образом. Поле Y дешифрируется ДШ Y и выходной управляющий сигнал y_i с дешифратора поступает в операционный автомат, возбуждая в нем выполнение заданной микрооперации.

Данная структура микрокоманды в каждом такте может выполнять не более одной микрооперации.

После выполнения микрооперации осуществляется переход к следующей микрокоманде, адрес которой определяется полем $A0$ или $A1$ в зависимости от значения поля X и логических условий x_1, \dots, x_L , формируемых в операционном автомате.

Если $X \neq 0$, то адрес A назначается равным $A0$ или $A1$ в зависимости от значения X_x , выделенного полем X : $A = A0$, если $X_x = 0$ или $A = A1$, если $X_x = 1$. Условно считается, что логическое условие x_0 тождественно нулю. Поэтому при $X = 0$ адрес $A = A0$. Это правило вычисления адреса следующей микрокоманды реализуется схемой, состоящей из дешифратора ДШХ, подсхемы И-ИЛИ и элемента НЕ, на выходе которых формируются управляющие сигналы a_0 и a_1 , инициирующие передачу $A := A0$ и $A := A1$ соответственно.

Таким образом к окончанию такта на адресной шине A будет сформирован адрес следующей микрокоманды, выбираемой из ПЗУ в очередном такте.

Перед началом работы управляющего автомата регистр микрокоманды МК устанавливается в нулевое положение, в результате чего адрес $A = 0$ и первой будет выбрана микрокоманда, хранимая в нулевой ячейке ПЗУ.

Запуск автомата приводится сигналом B , который переключит триггер T в состояние 1, в результате чего синхронизирующий сигнал C поступает на управляющий вход ЧТ, возбуждая в каждом такте процесс чтения микрокоманды из ПЗУ. В последнем такте реализации микропрограммы триггер T должен быть переключен в состояние 0.

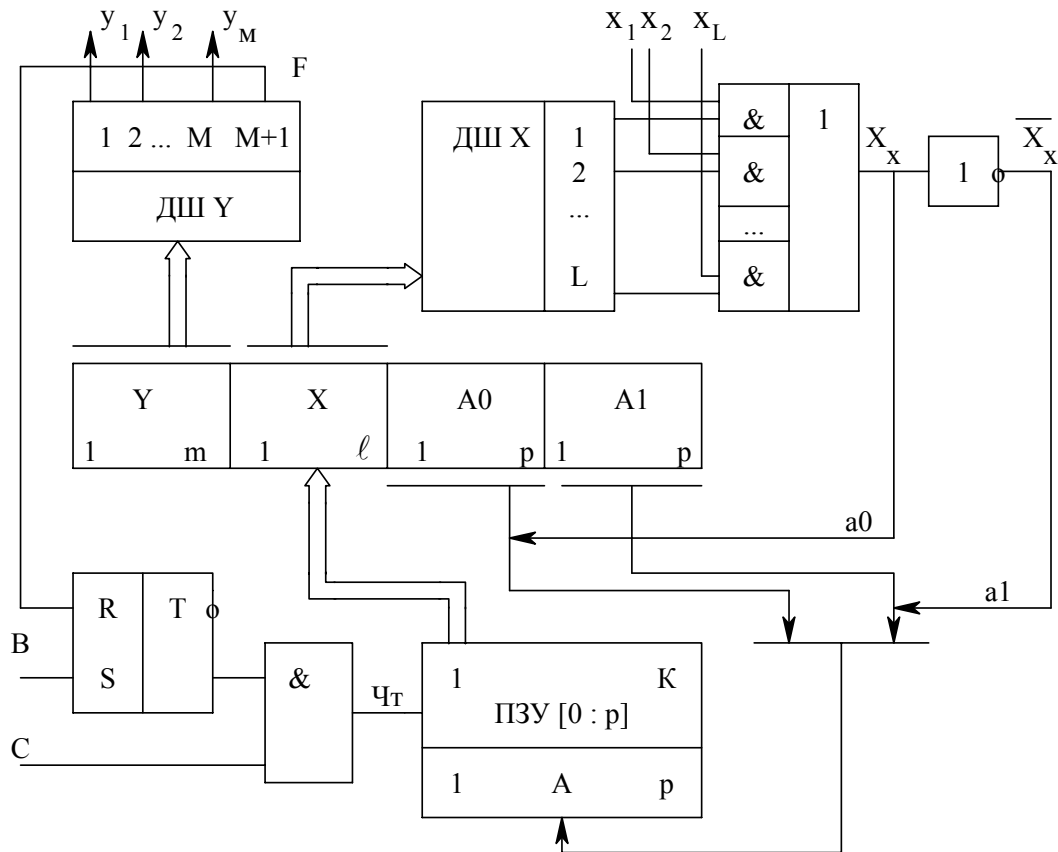


Рис 5.2. Управляющий автомат с принудительной адресацией и двумя адресными полями

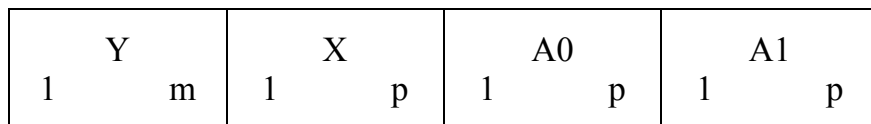


Рис 5.3. Структура микрокоманды

Для этого в системе микроопераций вводится дополнительная микрооперация Y_{m+1} , которая интерпретируется в виде сигнала F, останавливающего работу управляющего автомата.

Адрес следующей микрокоманды определяется в зависимости от кода X и значения X_x либо полем A0, либо полем A1.

С целью сокращения длины микрокоманды для формирования адреса следующей микрокоманды может отводиться единственное поле A. Если поле $X = 0$, то значение A, безусловно, определяет адрес следующей микрокоманды. Если $X \neq 0$, то адрес следующей микрокоманды равен $(A + X_x)$, где X_x – значение логического условия с номером X. В результате этого реализуется условный переход:

если $X_x = 0$, то к микрокоманде с адресом A;

если $X_x = 1$, то к микрокоманде с адресом $(A+1)$.

Указанный порядок формирования адреса реализуется схемой на рис. 5.4.

Исполнительный адрес $A1 = A + X_x$ формируется счетчиком КСЧ комбинационного типа.

Введение счетчика в схему уменьшает быстродействие автомата, поскольку длительность такта должна быть увеличена на время выполнения микрооперации счета в p -разрядном счетчике.

При естественной адресации адрес следующей микрооперации принимают равным увеличенному на единицу адресу предыдущей микрокоманды, т. е. если A – адрес выполняемой микрокоманды, то следующая микрокоманда выбирается из ячейки с адресом $(A + 1)$.

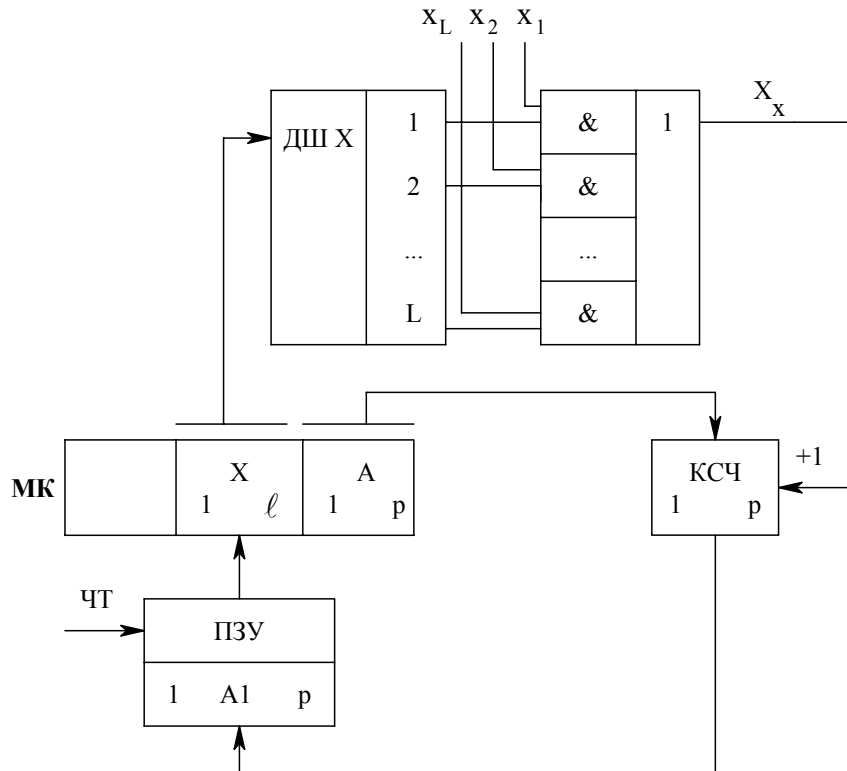


Рис. 5.4. Управляющий автомат с принудительной адресацией и одним адресным полем

Естественный способ адресации не требует введения адресного поля в каждую микрокоманду. Если микрокоманды следуют в естественном порядке, а процесс адресации реализуется счетчиком адреса микрокоманды, состояние которого увеличивается на единицу после чтения очередной микрокоманды. Следовательно, микрокоманды, которые задают функциональные преобразования, состоящие из набора микроопераций могут содержать только операционную часть с полями Y_1, Y_2, \dots, Y_M .

После выполнения микрокоманды с адресом A может возникнуть необходимость в переходе к микрокоманде с адресом $B \neq A+1$. Переход может быть безусловным или зависеть от текущего значения X_x . Условные переходы реализуются следующим образом:

если $X_x = 0$, то выполняется следующая микрокоманда с адресом $(A+1)$;

если $X_x = 1$, то следующей выполняется микрокоманда с адресом B .

При естественной адресации обычно используются микрокоманды двух типов: операционные и управляющие.

Операционная микрокоманда задает набор микроопераций Y_1, Y_2, \dots, Y_M и указывает на адрес следующей микрокоманды, равный $(A+1)$. Управляющие микрокоманды используются для изменения естественного порядка следования микрокоманд,

что сводится к выполнению безусловных и условных переходов. Управляющая микрокоманда содержит поле X , определяющее номер логического условия, и поле V , определяющее адрес следующей микрокоманды. Если $X = 0$, то адрес следующей микрокоманды безусловно равен V . Для выделения операционных и управляющих микрокоманд в управляющем слове вводится одноразрядное поле признака P , определяющего тип микрокоманды: если $P = 0$, то микрокоманда является операционной; если $P = 1$ – управляющей. Возможная структура рассмотренных управляющих слов изображена на рис. 5.5.

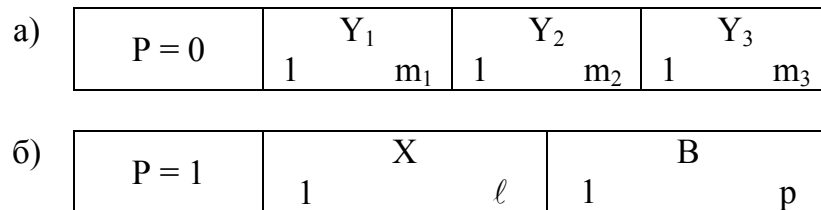


Рис. 5.5. Структура операционной (а) и управляющей (б) микрокоманды

Автомат, работающий с микрокомандами (рис. 5.5) строится по схеме (рис. 5.6).

Дешифраторы ДШ1, ДШ2, ДШ3, на выходе которых формируются управляющие сигналы y_1, y_2, \dots, y_m , стробируются сигналом P , принимающим значения 0 при выполнении операционной микрокоманды. ДШХ стробируется сигналом $P = MK(1)$, равным 1 при обработке управляющей микрокоманды. Адрес микрокоманды хранится и преобразуется на СЧА, с которым связаны микрооперации z_1 : $СЧА := СЧА + 1$ и z_2 : $СЧА := СЧА \div V$.

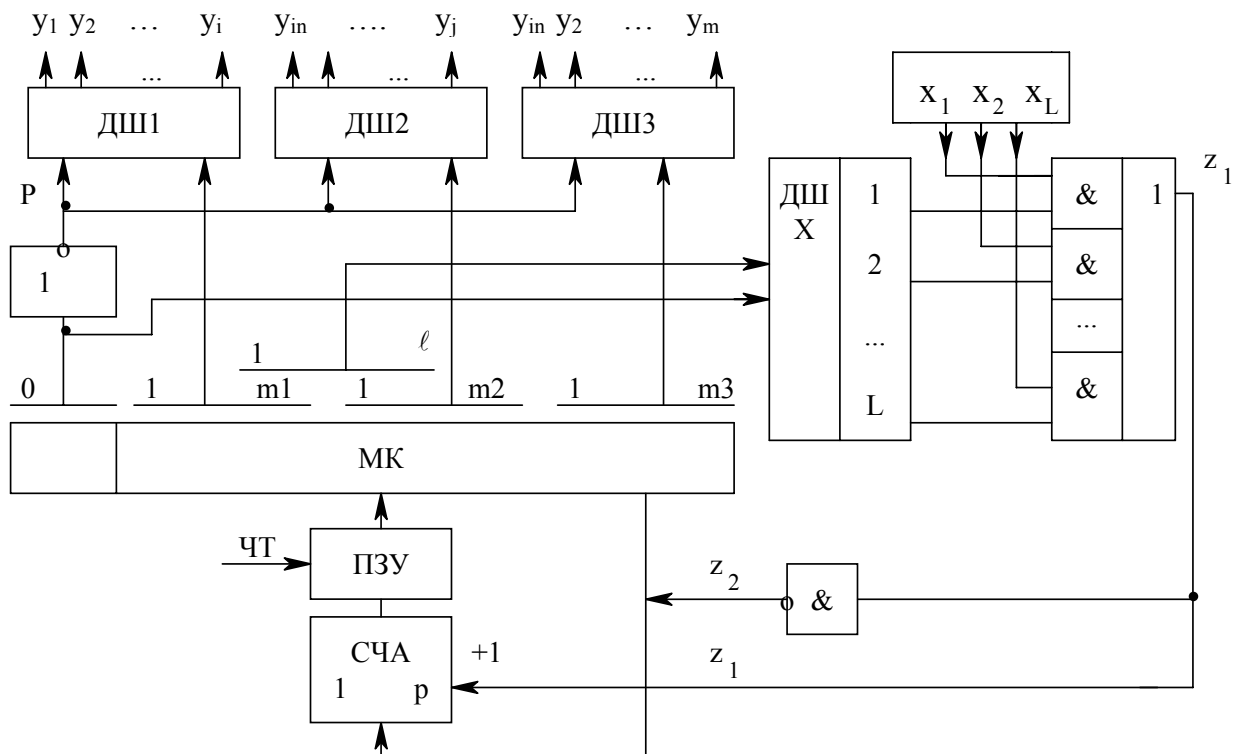


Рис. 5.6. Управляющий автомат с естественной адресацией

Сравнение вариантов реализации управляющих автоматов на основе ПЗУ: принудительная адресация и два адресных поля; принудительная адресация и одно адресное поле; естественная адресация показывает, что наименьшую разрядность ПЗУ обеспечивает вариант с использованием естественного способа адресации

Контрольные вопросы

1. Принцип микропрограммного управления автомата.
2. Система команд автоматов, реализующих вычислительные алгоритмы.
3. Набор операций автомата.
4. Состав и назначение элементов блок-схемы универсального микропрограммного автомата.
5. Общий алгоритм функционирования универсального микропрограммного автомата.
6. Основные характеристики автоматов.
7. Устройство управления микропрограммным автоматом.
8. Формирование адреса микрокоманд.

6. Проблемы отображения времени при проектировании

6.1. Модель тактируемого дискретного автомата

Сложное дискретное устройство будем рассматривать как композицию *управляющего* и *операционного* устройства. Операционное устройство в свою очередь является композицией *регистров* и *формеров*, которые опосредствуют взаимодействие регистров. **Формерами** называют любую комбинационную схему, предназначенную для преобразования информации.

Управляющее устройство – это композиция автоматов, каждый из которых также состоит из регистров и формеров. Выходные сигналы операционного устройства являются входными сигналами управляющего и, наоборот, выходные сигналы управляющего устройства являются входными сигналами операционного.

В качестве модели дискретного устройства, позволяющей рассмотреть временные явления, выберем модель дискретного автомата, изображенную на рис. 6.1.

Справа представлено операционное устройство (ОУ) в составе регистра R, разделенного на две ступени R1 и R2 и формеров на входах и выходах регистра R, слева – управляющий автомат (УА), реализованный на основе униблоков, что не имеет принципиального значения для последующего изложения. Существенным вначале будет лишь то, что элементы регистровой памяти автомата двухступенчатые.

Условимся структуру модели (рис. 6.1) представлять как бы многослойной. Иначе говоря, будем полагать, что в модели имеется не один регистр R, а некоторые системы двухступенчатых регистров, которые могут иметь произвольную разрядность, любые формеры на их входах и выходах, а между регистрами могут устанавливаться произвольные связи. Такой системой регистров, в частности, является операционное устройство вычислительной машины. При обращении к системе регистров пересылка информации с регистра адреса в регистр числа будет осуществляться через запоминающее устройство с соответствующими преобразованиями и задержкой.

Управляющий автомат

Операционное устройство

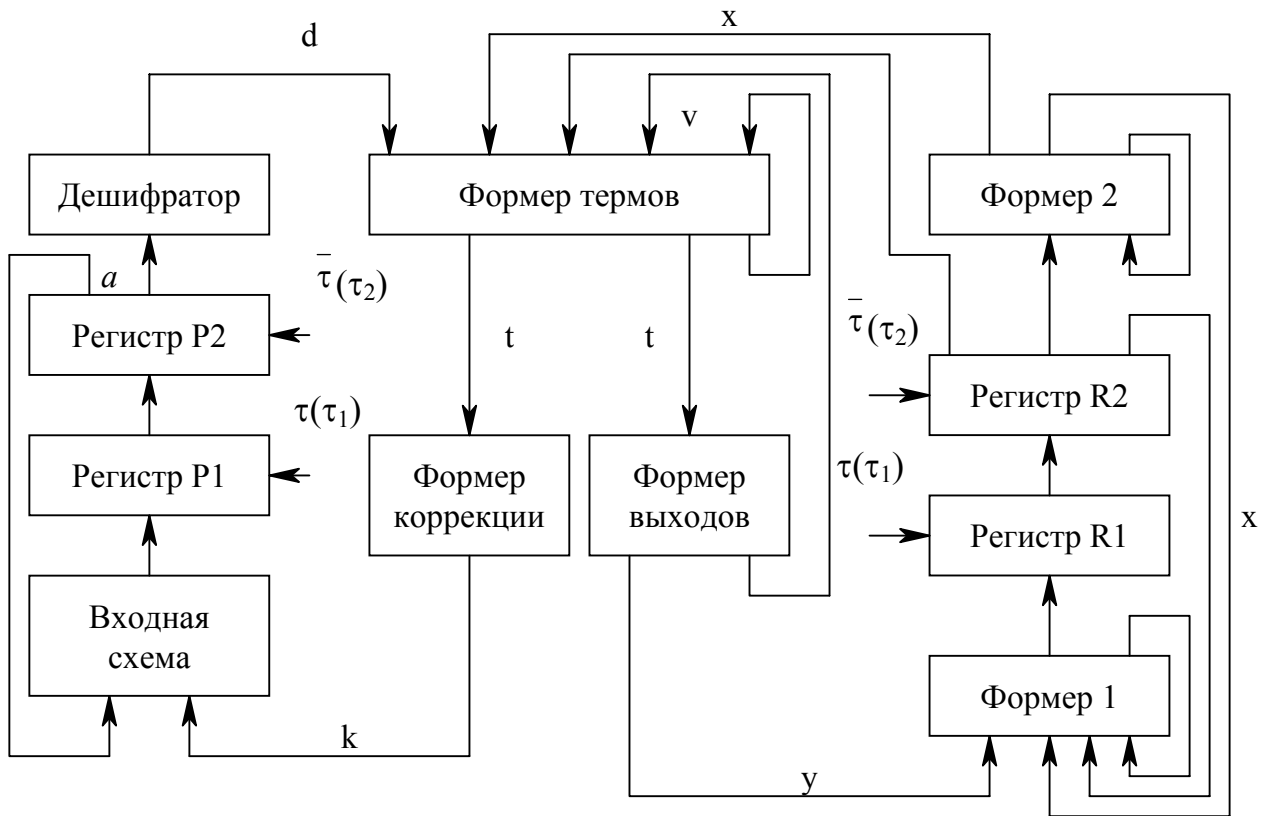


Рис. 6.1. Модель дискретного автомата

Будем также полагать, что в модели имеется не один автомат, а некоторое их множество. Каждый из автоматов может в качестве входных иметь сигналы с любого регистра и вырабатывать выходные сигналы, управляющие любым регистром операционного устройства. Автоматы могут работать в разное время и одновременно, автономно или во взаимодействии. При взаимодействии они обмениваются сигналами взаимодействия v , формирующимися либо в формере выходов, либо непосредственно в формере термов. В частности, таким множеством автоматов можно представить управляющее устройство вычислительной машины.

Как видно из рисунка 6.1, основным связывающим звеном, обеспечивающим взаимодействие всех частей дискретного устройства, являются формеры термов всех автоматов. Именно в них происходит взаимодействие информации, циркулирующей по двум основным видам цепей в системе. Это *цепи управления* – от выходов формеров термов через формеры коррекции и униблоки ко входам формеров термов и *цепи операций* – от формеров термов через формеры выходов и операционное устройство ко входам формеров термов.

Другие цепи прохождения информации, обусловленные в основном многослойностью изображенной структуры, часто оказываются как бы вставленными в виде отдельных звеньев в цепи операций, что увеличивает их общую длину, а значит и время прохождения сигналов по ним.

В то же время цепи управления не удлиняются цепями обратных связей в униблоках, но удлиняются при наличии сигналов v непосредственного взаимодейст-

вия автоматов. Однако, несмотря на это, более критичными к быстродействию, как правило, оказываются цепи операций.

6.2. Выбор параметров тактирующих сигналов

Рассмотрим применительно к описанной модели дискретного устройства (рис. 6.1) две наиболее распространенные системы тактирования, а именно тактирование одной серией сигналов τ и двумя сериями сигналов τ_1 и τ_2 , временные диаграммы которых показаны на рис. 6.2.

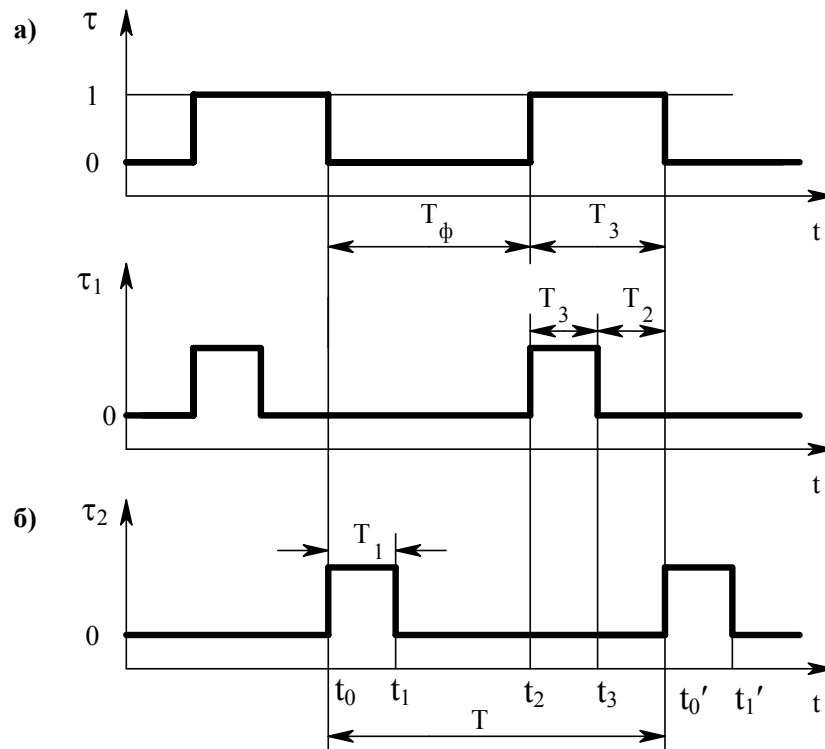


Рис. 6.2. Системы тактирования

Пусть системы тактирования (диаграммы (а) и (б) на рис. 6.2) имеют одинаковую длительность такта T . Период T представляет собой интервал времени $t_0 - t_0'$, через который характер процессов в устройстве повторяется. Моментом времени t_0 обозначим начало такта.

В начале такта при односерийной тактировке сигнал τ принимает нулевое значение, а при двухсерийной – сигнал τ_2 принимает единичное значение. В этот момент в устройстве происходит запоминание информации в регистрах P_2 и R_2 . Для устойчивой работы устройства длительность T_1 сигнала τ_2 должна обеспечивать надежное переключение триггеров в регистрах P_2 и R_2 .

Новые сигналы, появившиеся на выходах регистров P_2 и R_2 , начинают передаваться по цепям управления и цепям операций последовательно, проходя все встречающиеся на их пути формеры, пока не достигнут входов регистров P_1 и R_1 . Дальнейшее распространение сигналов по этим цепям приостанавливается в момент времени t_2 , в который при односерийной тактировке сигнал τ становится единичным, а при двухсерийной – единичным становится сигнал τ_1 . Эти сигналы откроют входы регистров P_1 и R_1 , и начинается запись новой информации в регистры.

Промежуток времени T_{ϕ} от момента времени t_0 до момента t_2 называется **периодом формирования** новой информации в данном такте. Он должен выбираться таким, чтобы переходные процессы выполнения всех одноктактных операций успели закончиться и на входах регистров P_1 и R_1 к моменту t_2 возникли стабильные сигналы. Таким образом, период формирования T_{ϕ} должен быть не меньше самой максимальной задержки сигнала во всех цепях управления и цепях одноктактных операций.

Интервал времени T_3 – это **период запоминания** информации, равный длительности сигнала τ и τ_1 . Поэтому длительность T_3 должна быть не меньше той минимальной длительности, при которой происходит надежное запоминание информации в регистрах P_1 и R_1 .

6.3. Сравнение способов тактирования автоматов

Очевидным преимуществом односерийной тактировки является то, что требуется только одна, а не две серии тактирующих сигналов. Кроме того, односерийная тактировка более быстродействующая по сравнению с двухсерийной. Так как если допустить, что при прочих равных условиях период запоминания T_3 сравниваемых способов тактирования одинаков, то получим, что при двухсерийной тактировке период T будет больше на интервал времени T_2 .

Однако двухсерийная тактировка более надежна чем односерийная, что определяется наличием в этой системе тактирования интервала времени T_2 между окончанием сигнала τ_1 и началом сигнала τ_2 с которого начинается новый такт. Действительно, при односерийной тактировке в момент смены тактов t_0' из-за различных задержек в цепях устройства сигналы τ и $\bar{\phi}$ в различных точках схемы могут иметь одновременно единичные значения. Поскольку сигнал τ , равный 1, фиксирует состояние устройства, сложившееся к концу предыдущего такта, а с сигналом $\bar{\phi}$, равным 1, начинается новый такт и формирование новой информации, то получается, что частично в некотором интервале времени устройство работает одновременно в режиме текущего такта и в режиме наступающего такта.

В такой ситуации из-за большого разброса задержек в разных цепях устройства может получиться, что информация, сформированная в режиме нового такта на одних элементах устройства, успевает зафиксироваться и на других элементах памяти, работающих еще в режиме предыдущего такта. При этом вновь записываемая информация стирает ту информацию, которая должна быть использована в новом такте, что, естественно, недопустимо.

Все это происходит из-за того, что при односерийной тактировке период формирования T_{ϕ} новой информации начинается непосредственно за периодом запоминания T_3 информации в предыдущем такте. При двухсерийной тактировке этого нет, т. к. в ней данные периоды разделены интервалом времени T_2 , который может быть любой длительности, необходимой для надежной работы устройства.

Отметим также, что в системах тактирования идеально прямоугольная форма сигналов не играет принципиальной роли, а введение ненулевой длительности фронтов сигналов приведет только к увеличению соответствующих интервалов времени.

6.4. Абсолютная и относительная шкала времени

В дискретных устройствах шкалой времени могут быть тактирующие серии сигналов, отмеченные метками на такой шкале. Метки на шкале, взятые сами по себе, безотносительно к какой-либо точке отсчета, образуют как бы абсолютную шкалу времени.

Под относительной шкалой понимаются те же метки, соотносящиеся с моментами пребывания автомата в определенных состояниях.

На рис. 6.3 приведена классификация входных сигналов, временные характеристики которых определяются либо по абсолютной шкале времени, либо по относительной.

В кружках, прямоугольниках и шестиугольниках приведены типы входных сигналов. Типы 1-9 – это сигналы, временные характеристики которых определяются по абсолютной шкале времени. При этом устанавливаются синхронность изменения сигнала относительно тактирующих сигналов и длительность интервала времени, на протяжении которого данный сигнал остается неизменным. Синхронность и длительность сигналов, а также характер изменения их фронтов легко определяются исходя из того, где и каким образом сигнал формируется. Синхронные сигналы характерны в основном для вычислительной техники, асинхронные чаще встречаются в устройствах дискретной автоматики.

Для использования относительной шкалы времени необходимо совместить начало отсчета с некоторым событием в дискретном устройстве и провести отчет тактов или долей тактов до некоторого другого события. За начало отсчета можно выбрать момент пребывания рассматриваемого автомата в таком состоянии, из которого легко проследить все интересующие события в дискретном устройстве (проектируемой системе).

Типы 10-16 – это входные сигналы, соотнесенные с относительной шкалой времени. Изменение этих типов сигналов приводят к изменениям в автомате, которым ставятся в соответствие времена его пребывания в некотором состоянии p .

Очевидно, что по определенной характеристике сигнала в абсолютной шкале времени можно определить его характеристику в относительной шкале.

Среди типов 10-15 только 13-й сигнал оказался ненадежным, требующим особой процедуры для устранения ненадежности. Вид геометрической фигуры показывает степень надежности данного типа сигналов при проектировании системы: сигналы в кружке в наибольшей степени обладают заданными качествами, в прямоугольнике – наименьшей, в шестиугольнике – в промежуточной степени.

Сплошные стрелки указывают на возможные варианты классификации типов сигналов, пунктирные – на возможные преобразования сигналов одного типа в сигналы другого типа. Таким образом, каждый тип сигналов будет иметь свои характеристики в абсолютной или в относительной шкале времени.

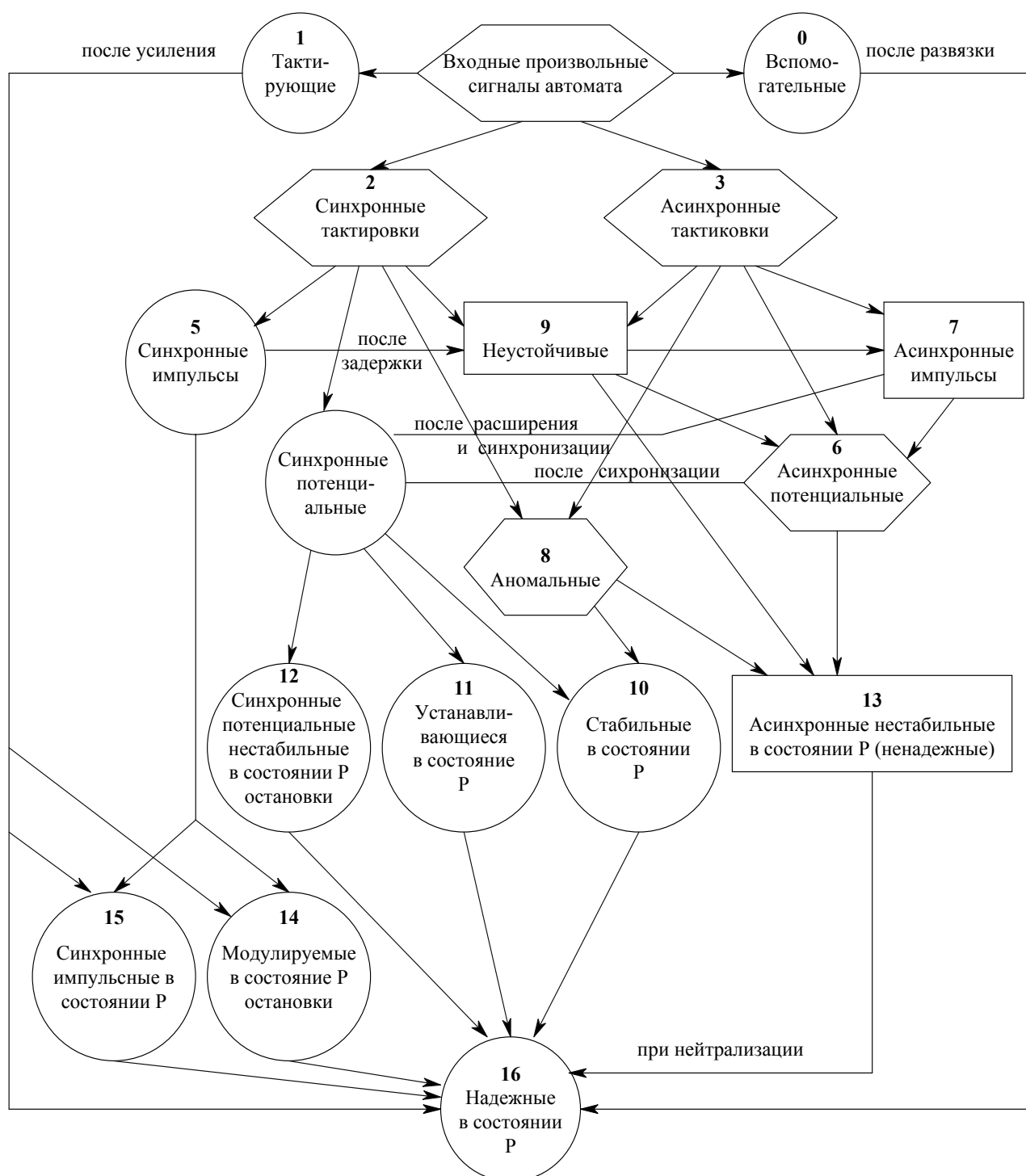


Рис. 6.3. Классификация входных сигналов автомата

6.5. Характеристики сигналов в абсолютной шкале времени

Входные сигналы, поступающие на автомат по входным каналам, можно поделить на четыре основных типа: вспомогательные, тактирующие, синхронные и асинхронные.

Тип 0. Вспомогательные сигналы. Это прежде всего различные напряжения питания автомата. Сигналами напряжения их можно считать потому, что в устройствах они могут использоваться по-разному, в том числе и как константы. В отношении их

надежности считают, что эти сигналы должны быть достаточной мощности и стабильности, без импульсных и высокочастотных полей. Соблюдение перечисленных характеристик достигается известными средствами, которые собирательно названы «развязкой». При соблюдении требований по развязке сигналы относят к разряду надежных (см. далее тип 16).

К вспомогательным относится также сигнал N начальной установки устройства. Он должен обладать такими временными и амплитудными характеристиками, которые бы надежно обеспечивали принудительную установку всех частей устройства в начальное состояние.

Тип 1. Тактирующие сигналы. Эти сигналы должны иметь соответствующую форму и амплитудные характеристики. Достигается это известными средствами, которые собирательно названы «усилением». Только после обеспечения требуемого усиления тактирующие сигналы можно отнести к разряду надежных.

В автоматах с двухступенчатыми триггерами тактирующие сигналы, как правило, поступают непосредственно на элементы памяти. Поэтому вначале эти сигналы не включают в автограмму автомата, то есть формальный синтез выполняют без них, а затем после выполнения этапа логического проектирования их вводят наряду со связями по питанию. Однако в автоматах с одноступенчатой памятью более удобно тактирующие сигналы с самого начала включить в условия автограммы наряду с другими входными сигналами. Кроме того, в условиях автограммы тактирующие сигналы могут входить и в качестве других сигналов (см. далее типы 14 и 15).

Тип 2. Синхронные сигналы. Это такие входные сигналы, изменение значения которых инициируется тактирующими сигналами. Самыми распространенными сигналами этого типа являются сигналы с выходов тактируемых триггеров. Другие сигналы, формируемые внутри синхронной системы, также считают синхронными.

Тип 3. Асинхронные сигналы. Это те сигналы, значения которых меняются независимо от тактирующих сигналов. Сигналы, поступающие в синхронную систему извне, обычно называют асинхронными. Большинство сигналов в системах дискретной автоматики относят к асинхронным.

В свою очередь синхронные и асинхронные сигналы также делятся на четыре типа в зависимости от их длительности, измеряемой интервалом времени с неизменным значением сигнала, и в зависимости от процесса изменения их значения. Процесс изменения сигналов называется *нормальным*, если он по длительности и монотонности сходен с процессом изменения тактирующих сигналов. *Ненормальным* называют процесс изменения сигналов либо существенно более медленный, чем у тактирующих сигналов, либо немонотонный, либо то и другое вместе. В представленной классификации у типов 4-7 нормальный процесс изменения, а у типов 8 и 9 – ненормальный.

Тип 4. Синхронные потенциальные сигналы. Это синхронные сигналы с нормальным процессом изменения, меняющие свое значение при одноступенчатых элементах памяти не чаще, чем один раз на протяжении одного такта, а при двухступенчатых элементах памяти не чаще, чем один раз на протяжении двух тактов. В основном это сигналы с выходов тактируемых триггеров. Они наиболее надежны и поэтому часто употребляются при проектировании синхронных систем. Сигналы типов 5-9 могут быть преобразованы в синхронные потенциальные сигналы различными средствами. Также этот тип сигналов является единственным источником сигналов типов 11-12 и часто основным источником сигналов типа 10.

Тип 5. Синхронные импульсные сигналы. Это такие сигналы с нормальным процессом изменения, единичное значение которых с небольшим (в доли такта) запаздыванием повторяет единичное значение тактирующих сигналов. Они получаются в результате стробирования потенциальных сигналов тактирующими, поэтому, в отличие от сигналов типа 4, они могут менять свое значение два раза за один такт.

В неизменном виде синхронные импульсные сигналы используются редко. Иногда их применяют в качестве входных сигналов типов 14 и 15 только в автоматах с одноступенчатыми элементами памяти, то есть когда они заменяют тактирующие сигналы. Наиболее часто эти сигналы с помощью специальных триггеров расширяются до потенциальных сигналов типа 4 и в таком виде применяются в любых автоматах. Это позволяет реализовывать устройства без совпадения двух импульсных сигналов, которое всегда нежелательно с точки зрения надежности. А поскольку в синхронных устройствах тактирующие (импульсные) сигналы так или иначе присутствуют, то применение других импульсных сигналов следует сводить к минимуму, допуская их разве что в исключительных случаях, несомненных с точки зрения надежности.

Тип 6. Асинхронные потенциальные сигналы. Такие сигналы относятся к сигналам с нормальным процессом изменения. Каждое изменение сигналов необходимо зафиксировать на тактирующем триггере, поэтому сигналы изменяются достаточно медленно, чтобы обеспечить на триггере их синхронизированность. При этом два следующих друг за другом изменения сигнала необходимо разделять несколько большим интервалом времени, чем такт в автоматах с двухступенчатой памятью и несколько большим двух тактов, чем в автоматах с одноступенчатой памятью. Если сигналы изменяются чаще этих интервалов времени, то они относятся к сигналам типа 7.

В неизменном виде асинхронные потенциальные сигналы используются в автомате в качестве сигналов типов 10 и 13. Обычно они после синхронизации преобразовываются в синхронные потенциальные сигналы типа 4.

Тип 7. Асинхронные импульсные сигналы. Это такие сигналы с нормальным процессом изменения, которые не могут быть отнесены к сигналам типа 6. Надежная синхронизация этого типа возможна только после предварительного его расширения на специальном триггере, в результате этого асинхронный импульсный сигнал преобразуется в асинхронный потенциальный сигнал.

Без преобразования в потенциальные сигналы асинхронный импульсный сигнал может использоваться в автомате лишь в качестве ненадежного сигнала типа 13.

Асинхронный сигнал, вызывающий сомнения относительно того, импульсный он или потенциальный, по требованию надежности должен рассматриваться как импульсный.

Тип 8. Аномальные сигналы. Это сигналы с ненормальным процессом изменения, установившееся значение которых остается стабильным в течение не менее двух тактов при двухступенчатых элементах памяти и в течение не менее четырех тактов при одноступенчатых элементах. Эти сигналы формируются в результате выполнения многотактных операций (например, сигнал переноса в сумматоре), ими являются также сигналы с кнопок и разных электромеханических датчиков и др.

Процесс изменения их значения либо слишком медленный, либо немонотонный, из-за чего с ними следует обращаться как с асинхронными, независимо от того, синхронно или асинхронно с тактирующими сигналами начинается процесс.

Непреобразованные аномальные сигналы могут использоваться в автомате как сигналы типов 10 и 13. В сигналы типа 4 они могут преобразоваться путем синхронизации на триггере в тех случаях, когда начало изменения сигнала может быть воспринято, как его окончательное изменение (например, в случае сигналов с кнопок).

Если же аномальный сигнал должен использоваться только после его полной стабилизации, то тогда должны быть приняты меры для блокировки этого сигнала в период его формирования, например, посредством сигнала типа 12.

Тун 9. Неустойчивые сигналы. Это сигналы, значения которых быстро изменяются из-за отсутствия достаточной задержки в цепи обратной связи «автомат – объект управления – автомат».

Неустойчивым входной сигнал x_1 автомата становится тогда, когда его значение обусловлено выходными сигналами y_1 , формируемыми в зависимости от значения сигнала x_1 .

В тактируемых устройствах основным источником неустойчивых сигналов являются одноступенчатые триггеры.

Введение достаточной задержки в любом звене цепи обратной связи превращает неустойчивый сигнал в сигнал одного из типов 4-7. В непосредственном виде неустойчивый сигнал может использоваться только в качестве сигнала типа 13.

6.6. Характеристики сигналов в относительной шкале времени

Тун 10. Сигналы, стабильные в состоянии p . Это сигналы, значение которых не меняется все время, пока автомат находится в данном состоянии p . Таковыми могут быть лишь сигналы типов 4, 6, 8. Сигналы стабильные в тех состояниях, в которых они используются в качестве входных, самые надежные и удобные при проектировании. В синхронных устройствах эти сигналы встречаются наиболее часто.

Тун 11. Сигналы, устанавливающиеся в состояние p . Это сигналы, которые начинают принимать новое значение одновременно с переходом автомата в состояние p .

Такие сигналы являются результатом операций, заканчивающихся в предшествующем такте, поэтому они могут использоваться сразу же в наступившем такте. Гарантией того, что они успеют произвести должное действие, является правильный выбор параметров тактирующих сигналов. Если эти сигналы являются результатом одноктактных операций, то в последующем такте они опять могут изменяться при условии, что элементы памяти автомата двухступенчатые. А при одноступенчатых элементах памяти устанавливающиеся в состояние p сигналы должны оставаться стабильными еще по крайней мере один такт.

Сигналы этого типа, как и типа 10, часто применяются. Ими могут быть только синхронные потенциальные сигналы типа 4.

Тун 12. Сигналы синхронные потенциальные нестабильные в состоянии p . Название этих сигналов полностью их определяет. Они рассматриваются только применительно к состояниям с остановкой, в которых автомат пребывает не менее двух тактов, без чего невозможно изменение сигнала на границе между тактами в то время, когда автомат находится в данном состоянии.

Введение таких сигналов в условие строк остановки позволяет их использовать как модулирующие, а в условия строк перехода их вводят для задержки перехода или для блокировки аномальных сигналов, пока значение последних еще не сформировалось. Сигналами типа 12 могут быть только сигналы типа 4.

Tun 13. Сигналы асинхронные нестабильные в состоянии р. Таковыми могут быть сигналы типов 6-9, не преобразуемые в синхронные. Этот тип сигналов, ненадежных в данном состоянии р, может быть применен в качестве входных сигналов автомата, только в условиях выполнения дополнительных предосторожностей, компенсирующих их ненадежность.

Tun 14. Сигналы, модулируемые в состоянии р. Это серии тактирующих (тип 1) или синхронных импульсных (тип 5) сигналов, которые модулируются с целью вырезания из серий пакетов сигналов требуемой конфигурации. Такие пакеты используются при композиции автоматов в качестве тактирующих сигналов, для отсчета времени, синхронизации и т. п.

Модулирование можно производить при пребывании автомата в данном состоянии р. Для этого достаточно записать символ модулируемого сигнала в качестве условия строки остановки в пункте р автограммы, а в качестве выходного – символ промодулированного сигнала. Реализация выражения последнего сигнала, полученного в результате формального синтеза автомата, и дает требуемый пакет первичных сигналов.

Такой пакет можно вторично модулировать, разбив его на субпакеты. Для этого в условие той же строки остановки надо дописать через конъюнкцию модулирующий сигнал типа 12 с временными параметрами, соответствующими конфигурации требуемых субпакетов. Аналогично, введя в то же условие еще один часто меняющийся сигнал, можно модулировать и субпакеты и т. д. Здесь открываются большие возможности для формализации сложных процессов взаимодействия в синхронных системах.

В качестве модулируемых сигналы типа 14 проходят только через схему термов, как бы едва задевая автомат. Поэтому в пункте р они присутствуют в прямом варианте и отсутствуют в инверсном. При определении формальной полноты пункта р их не следует учитывать, полагая их равными 1.

Tun 15. Сигналы асинхронные импульсные в состоянии р. Это импульсные сигналы, используемые в условиях автограммы при синтезе автоматов только с одноступенчатыми элементами памяти. Ими могут быть сигналы типов 1 или 5, причем сигналы типа 5 используются в условиях автограммы вместо тактирующих. Как правило, сигналами типа 5 осуществляется непосредственное взаимодействие автоматов между собой. Эти сигналы проходят через формеры термов и выходов двух (и более) взаимодействующих автоматов, отчего задержка их во времени увеличивается, а надежность уменьшается, и поэтому применение сигналов типа 5 нежелательно.

При соответствующей формулировке условий автограммы и ограничении глубины взаимодействия автоматов, например, двумя автоматами, иногда можно в виде исключения применять сигналы типа 5 в качестве сигналов типа 15 в биавтоматах и моноавтоматах.

Tun 16. Сигналы, надежные в состоянии р. Этот тип сигналов включает типы 10-12, 14, 15. Надежным в состоянии р будет называться сигнал такой стабильности, длительности и момента появления, что он может надежно произвести требуемое воздействие на автомат, находящийся в состоянии р. В противном случае сигнал считается ненадежным в состоянии р и, в соответствии с приведенной классификацией, должен быть отнесен к сигналам типа 13.

Из всех входных сигналов автомата наибольший интерес представляет надежность в данном состоянии р лишь тех сигналов, которые входят в условия пункта р,

т. е. значение которых существенно влияет на поведение автомата в состоянии p . Не существенные в состоянии p входные сигналы, с точки зрения формулировки пункта p и надежного функционирования автомата в данном состоянии p , могут иметь произвольные временные характеристики.

Контрольные вопросы

1. Модель тактируемого дискретного автомата.
2. Выбор параметров тактирующих сигналов.
3. Сравнение способов тактирования автомата.
4. Абсолютная и относительная шкала времени.
5. Система классификации входных сигналов.
6. Характеристики сигналов в абсолютной шкале времени.
7. Характеристики сигналов в относительной шкале времени.

7. СЕТИ ПЕТРИ

7.1. Назначение и общая характеристика сетей Петри

Среди многих методов описания и анализа дискретных параллельных систем выделяют подход, который основан на использовании сетевых моделей, относящихся к сетям специального вида, предложенных Карлом Петри для моделирования асинхронных информационных потоков в системах преобразования данных.

Сети Петри обеспечивают описание как алгоритмов и программ, так и собственно вычислительных систем и их устройств, а также порождаемых вычислительных процессов. Сети Петри используются для решения разнообразных задач анализа, синтеза и оптимизации. В том числе для решения прикладных задач и в основном задач, связанных с моделями и средствами параллельной обработки информации. В связи с этим много внимания уделяется изучению понятий сетей Петри, их связи с математическим аппаратом теории систем, теоретического программирования и т. п.

Среди приложений теории сетей Петри к задачам моделирования дискретных систем наибольшее развитие получили работы, связанные с попытками использовать аппарат сетей Петри, их модификации и обобщения для описания и изучения структурной динамики программ, в первую очередь – так называемых параллельных программ. Первый шаг к построению модели дискретной системы – абстрагирование от конкретных физических и функциональных особенностей ее компонентов. Компоненты системы и их действия представляются абстрактными событиями, например, исполнение оператора программы, прерывание в операционной системе и т. д.

Событие может произойти (реализоваться) один раз, повториться многократно, порождая конкретные действия (реализация события), или не произойти ни разу.

Совокупность действий, возникающих как реализация событий при функционировании дискретной системы, образует процесс, порождаемый этой системой. В общем случае одна и та же система может функционировать в одних и тех же условиях по-разному, порождая некоторое множество процессов, т. е. функционировать недетерминированно.

Реальная система функционирует во времени, событие происходит в некото-

рые моменты времени и длится некоторое время. В синхронных моделях дискретных систем события явно привязаны к определенным моментам или интервалам времени, в которые происходит одновременное изменение состояния всех компонентов системы, то есть изменение общего состояния системы. Смена состояний происходит последовательно.

В сетях Петри обычно отказываются от введения в модели дискретных систем времени и тактированных последовательностей изменений состояний, заменяя их причинно-следственными связями между событиями (асинхронные модели). Если возникает необходимость осуществить привязку ко времени, то моменты или интервалы времени представляют как события. Таким образом синхронные системы могут описываться в терминах асинхронных моделей. Отказ от времени приводит к тому, что события в асинхронной модели рассматриваются как элементарные (неделимые, мгновенные) или как составные, имеющие некоторую внутреннюю структуру, образованную из подсобытий.

Взаимодействие событий в сложных асинхронных системах имеет, как правило, сложную динамическую структуру. Эти взаимодействия будут описываться более просто, если указывать не непосредственные связи между событиями, а те ситуации, при которых данное событие может реализоваться. При этом глобальные ситуации в системе формируются с помощью локальных операций, называемых условиями реализации событий.

Условие имеет емкость: условие не выполнено (емкость = 0), выполнено (емкость = 1), условие выполнено с n -кратным запасом (емкость = n , где n – целое положительное число). Условие соответствует таким ситуациям в моделируемой системе, как наличие данных для операции в программе, состояние некоторого регистра в ЭВМ и т. п. Определенные сочетания условий разрешают реализоваться некоторому событию (предусловия события), а реализация события изменяет некоторые условия (постусловия события), т. е. события взаимодействуют с условиями, а условия – с событиями.

Таким образом, предполагается, что для решения указанных задач достаточно представлять дискретные системы как структуры, образованные из элементов двух типов: *событий* и *условий*.

В сетях Петри события и условия представлены абстрактными символами из двух непересекающихся алфавитов, называемых соответственно **множеством переходов** и **множеством позиций**. В графическом представлении сетей переходы изображаются «барьерами» или планками (t), а позиции – кружками (P) (рис. 7.1).

Условия и события-переходы связаны отношением непосредственной зависимости, которое изображается с помощью направленных дуг, ведущих из позиции в переходы и из переходов в позиции. Позиции, из которых ведут дуги на данный переход, называются его *выходными позициями*. Позиции, на которые ведут дуги из данного перехода, называются его *входными позициями*.

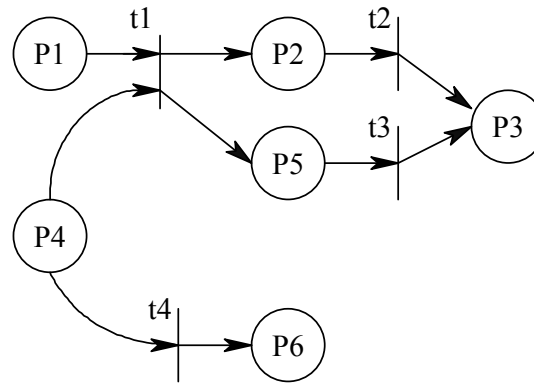


Рис. 7.1. Графическое изображение сети Петри

Выполнение условий изображается разметкой соответствующей позиции, а именно: помещением числа n *фишек (маркеров)* в соответствующее место, где $n > 0$ – емкость условия. Например:

$P \circ$ - условие p не выполнено;

$P \odot_3$ - условие p имеет емкость 3;

$P \odot$ - условие p выполнено;

$P \odot_5$ - условие p имеет емкость 5.

Неформально работу сети можно представить как совокупность локальных действий, которые называются *срабатываниями* переходов. Они соответствуют реализациям событий и приводят к изменению разметки позиций, т. е. к локальному изменению условий в системе.

Таким образом, сети Петри позволяют перейти от понятий абстрактной асинхронной системы к динамической структуре из событий и условий. В общей теории сетей сеть Петри рассматривается как один из способов сетевого моделирования систем. Вводятся обобщенные сетевые модели. Их единую основу образует понятие неинтерпретированной ориентированной сети из условий и событий, которая описывает только статическое строение системы. Если сеть Петри описывает функциональную схему моделируемой системы, то работа сети моделирует процесс, происходящий при функционировании системы.

7.2. Структура и способы представления сетей Петри

Моделирующие возможности сетей Петри и их эффективность объясняются прежде всего тем, что сеть Петри – это интеграция графа и дискретной динамической системы. Она может быть статической или динамической моделью представляемого с ее помощью объекта. При этом отсутствие строго фиксированного аналитического подхода при определении отношения вход-выход сети делает эту систему алгоритмически неопределенной как и для имитационных моделей. Особенную роль сети Петри играют при моделировании параллельных процессов. Учитывается также преимущество этих сетей – удобство их программирования на ЭВМ.

Применение сетей Петри не ограничивается, только, моделированием процессов и динамических систем. Они с успехом используются и в теоретическом программировании при решении задач функциональной спецификации, верификации программного обеспечения, организации вычислительных процессов, управления.

Выделяют четыре типа задач исследования объектов с помощью сетей Петри:

- 1) интерпретация (программирование объекта), связанная с адекватным представлением моделируемого объекта соответствующей сетью Петри;
- 2) программирование модели в конкретной операционной среде;
- 3) исследование модели;
- 4) кросс-трансляция с языка сетей Петри на языки программирования.

Возможности сетей Петри ограничиваются для автоматных сетей и маркированных графов, которые допускают один вход и один выход для переходов и позиций соответственно. Введение в рассмотрение мультипликативности для дуг и количества фишек в позициях позволяет моделировать не только функционирование параллельных систем, но и динамику объемов ресурсов.

7.2.1. Структура сетей Петри

Сеть Петри состоит из 4-х элементов: множество позиций P , множество переходов T , входная функция I и выходная функция O . Входная и выходная функции связаны с переходами и позициями. Входная функция I отображает переход t_j в множество позиций $I(t_j)$, называемых входными позициями перехода. Выходная функция O отображает переход t_j в множество позиций $O(t_j)$, называемых выходными позициями перехода.

Структура сети Петри определяется ее позициями, переходами входной и выходной функции.

Определение:

Сеть Петри S является четверкой $S = (P, T, I, O)$, где

$P = \{p_1, p_2, \dots, p_n\}$ – конечное множество позиций, $n \geq 0$;

$T = \{t_1, t_2, \dots, t_m\}$ – конечное множество переходов, $m \geq 0$. Множество позиций и множество переходов не пересекаются, $P \cap T = \emptyset$;

$I : T \rightarrow P^\infty$ есть входная функция – отображение из переходов в комплекты позиций;

$O : T \rightarrow P^\infty$ есть выходная функция – отображение из переходов в комплекты позиций.

Мощность множества P есть число n , а мощность множества T есть число m . Произвольный элемент P обозначается символом p_i , $i = 1, \dots, n$, а произвольный элемент T – символом t_j , $j = 1, \dots, m$.

Позиция p_i является **входной позицией** перехода t_j в том случае, если $p_i \in I(t_j)$; p_i является **выходной позицией**, если $p_i \in O(t_j)$.

Входы и выходы переходов представляют собой **комплекты** позиций. **Комплект** является обобщением множества, в которое включены многократно повторяющиеся элементы – тиражированные элементы. Использование комплектов, а не множеств для входов и выходов перехода позволяет позиции иметь кратный вход либо кратный выход для соответствующего перехода. **Кратность** входной позиции p_i для перехода t_j есть число появлений позиции во входном комплекте перехода $\#(p_i, I(t_j))$. Аналогично кратность выходной позиции p_i для перехода t_j есть число появлений позиции в выходном комплекте перехода $\#(p_i, O(t_j))$. Если входная и выходная функции являются множествами (а не комплектами), то кратность каждой позиции есть либо 0, либо 1.

Входные и выходные функции используются для отображения позиций в комплекты переходов или наоборот для отображения переходов в комплекты позиций.

Определим, что переход t_j является входом позиции p_i , если p_i есть выход t_j . Переход t_j есть выход позиции p_i , если p_i есть вход t_j .

7.2.2. Графы сетей Петри

Наглядность сетевого моделирования систем существенно повышается, если использовать теоретико-графовое представление сети Петри в виде двудольного ориентированного **мультиграфа**.

В соответствии с данным подходом структура сети Петри представляет собой совокупность позиций и переходов, а граф сети – два типа узлов, соединенных дугами (стрелками).

Кружок O является позицией, а планка $|$ – переходом. Ориентированные дуги соединяют позиции и переходы, при этом одни дуги направлены от позиций к переходам, а другие – от переходов к позициям. Дуга, направленная от позиции p_i к переходу t_j , определяет позицию, которая является входом перехода. Кратные входы в переход указываются кратными дугами из входных позиций в переход. Выходная позиция указывается дугой от перехода к позиции. Кратные выходы также представлены кратными дугами.

Сеть Петри есть мультиграф, так как он допускает существование кратных дуг от одной вершины графа к другой. Следует добавить, что так как дуги являются направленными, то это ориентированный мультиграф. Вершины графа можно разделить на два множества (позиции и переходы) таким образом, что каждая дуга будет направлена от элемента одного множества (позиций или переходов) к элементу другого множества (переходов или позиций); следовательно, такой граф является двудольным ориентированным мультиграфом. В дальнейшем для простоты будем называть его просто графом сети Петри.

Определение. Граф G сети Петри есть двудольный ориентированный мультиграф $G = (V, A)$, где

$V = \{v_1, v_2, \dots, v_s\}$ – множество вершин;

$A = \{a_1, a_2, \dots, a_r\}$ – комплект направленных дуг;

$a_i = (v_j, v_k)$, где $v_j, v_k \in V$.

Множество V может быть разбито на два непересекающихся подмножества P и T , таких, что $V = P \cup T$ и $P \cap T = \emptyset$. Кроме того для любой направленной дуги $a_i \in A$, если $a_i = (v_j, v_k)$, при условии, что тогда либо $v_j \in P$ и $v_k \in T$, либо $v_j \in T$ и $v_k \in P$.

Для демонстрации эквивалентности двух представлений сети Петри: структуры сети Петри и графа сети Петри (рис. 7.2 и 7.3) – покажем, каким образом можно преобразовать одно представление сети в другое.

Предположим, что дана структура сети Петри $S = (P, T, I, O)$ с $P = \{p_1, p_2, \dots, p_n\}$ и $T = \{t_1, t_2, \dots, t_m\}$. Тогда граф сети Петри можно определить следующим образом. Пусть $V = P \cup T$. Определим A как комплект направленных дуг, такой что для всех $p_i \in P$ и $t_j \in T$

$$\#((p_i, t_j), A) = \#(p_i, I(t_j));$$

$$\#((t_j, p_i), A) = \#(p_i, O(t_j)),$$

тогда $G(V, A)$ есть граф сети Петри, эквивалентный структуре сети Петри $S = (P, T, I, O)$.

Структура сети Петри $S(P, T, I, O)$, где

$P \{p_1, p_2, p_3, p_4, p_5\},$
 $T \{t_1, t_2, t_3, t_4\},$
 $I(t_1) = \{p_1\}, I(t_2) = \{p_2, p_3, p_5\}, I(t_3) = \{p_3\}, I(t_4) = \{p_4\},$
 $O(t_1) = \{p_2, p_3, p_5\}, O(t_2) = \{p_5\}, O(t_3) = \{p_4\}, O(t_4) = \{p_2, p_3\}.$
 Рис. 7.2. Структура сети Петри $C(P, T, I, O)$

На рис. 7.2 структура сети Петри представлена в виде четверки, которая состоит из множества позиций (P), множества переходов (T), входной функции $I : T \rightarrow P^\infty$ и выходной функции $O : T \rightarrow P^\infty$.

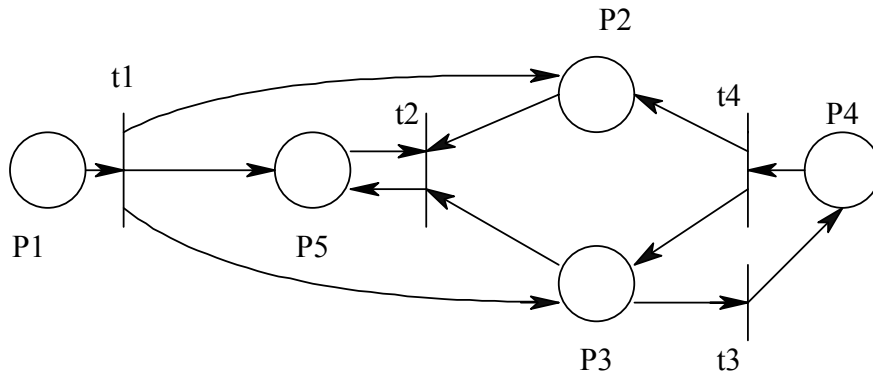


Рис. 7.3. Граф сети Петри $G(V, A)$

Таким образом, граф сети Петри, изображенный на рис. 7.3 эквивалентен структуре сети Петри на рис. 7.2.

7.2.3. Маркировка сетей Петри

Маркировка μ сети Петри заключается в присвоении фишек (маркеров) позициям сети Петри. Маркировка сети Петри есть функция, отображающая множество позиций P в множество неотрицательных целых чисел N .

$$\mu : P \rightarrow N.$$

Если маркированная сеть Петри $M = (C, \mu)$ есть совокупность структуры сети Петри $C = (P, T, I, O)$ и маркировки μ , то она может быть записана в виде $M = (P, T, I, O, \mu)$.

На рис. 7.4 представлена маркированная сеть Петри.

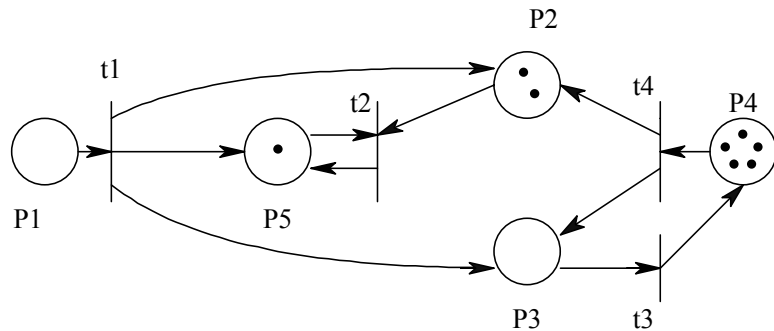


Рис. 7.4. Маркированная сеть Петри. Маркировка $(0, 2, 0, 5, 1)$

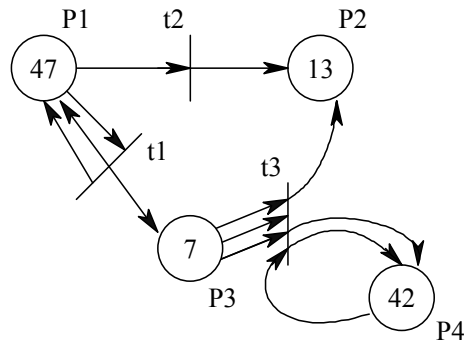


Рис. 7.5. Маркированная сеть Петри с очень большой маркировкой (47, 13, 7, 42)

7.2.4. Работа сетей Петри

Работа сети Петри заключается в перемещении фишек из одних позиций в другие. Фишки находятся в кружках и управляют выполнением переходов сети. Сеть Петри выполняется посредством запусков переходов. Переход запускается удалением фишек из его входных позиций и образованием новых фишек, помещаемых в его выходные позиции.

Переход может запускаться только в том случае, когда он разрешен. Переход называется *разрешенным*, если каждая из его входных позиций имеет число фишек по крайней мере равное числу дуг из позиции в переход. Кратные фишки необходимы для кратных входных дуг. Фишки во входной позиции, которые разрешают переход, называются его *разрешающими фишками*. Например, если позиции p_1 и p_2 служат входами для перехода t_4 , тогда t_4 разрешен, если p_1 и p_2 имеют хотя бы по одной фишке. Для перехода t_7 с входным комплектом $\{p_6, p_6, p_6\}$ позиция p_6 должна обладать по крайней мере тремя фишками, для того чтобы t_7 был разрешен.

Определение.

Переход $t_j \in T$ в маркированной сети Петри $S = (P, T, I, O)$ с маркировкой μ разрешен, если для всех $p_i \in P$

$$\mu(p_i) \geq \#(p_i, I(t_j)).$$

Переход запускается удалением всех разрешающих фишек из его входных позиций и последующим помещением в каждую из его выходных позиций по одной фишке для каждой дуги. Кратные фишки создаются для кратных выходных дуг.

Например, переход t_3 с $I(t_3) = \{p_2\}$ и $O(t_3) = \{p_7, p_{13}\}$ разрешен всякий раз, когда в p_2 будет хотя бы одна фишка. Переход t_3 запускается удалением одной фишки из позиции p_2 и помещением одной фишки в позицию p_7 и в p_{13} (его выходы). Дополнительные фишки в позиции p_2 не влияют на запуск t_3 (хотя они могут разрешать дополнительные запуски t_3). Переход t_2 , в котором $I(t_2) = \{p_{21}, p_{23}\}$ и $O(t_2) = \{p_{23}, p_{25}, p_{25}\}$ запускается удалением одной фишки из p_{21} и одной фишки из p_{23} , при этом одна фишка помещается в p_{23} и две в p_{25} (т. к. p_{25} имеет кратность, равную двум).

Запуск перехода в целом заменяет маркировку μ сети Петри на новую маркировку μ' . Так как можно запустить только разрешенный переход, то при запуске перехода количество фишек в каждой позиции всегда остается неотрицательным. Запуск перехода никогда не удалит фишку, отсутствующую во входной позиции. Если какая-либо входная позиция перехода не обладает достаточным количеством фишек, то переход не разрешен и не может быть запущен.

Определение. Переход t_j в маркированной сети Петри с маркировкой μ может быть запущен всякий раз, когда он разрешен. В результате запуска разрешенного перехода t_j образуется новая маркировка μ' , определяемая следующим соотношением:

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j)).$$

7.3. Анализ сетей Петри

С помощью сетей Петри можно моделировать широкий класс систем, представляя должным образом взаимодействие различных процессов, которые могут возникнуть в системе. Как отмечалось ранее, наиболее часто сети Петри применяют при моделировании систем, включающих параллельные действия.

Моделирование системы (устройства) на основе сетей Петри предполагает проведение тщательного анализа, который должен привести к глубокому пониманию поведения моделируемой системы. Таким образом, необходимо рассмотреть методы анализа и свойства сетей Петри. Для этого рассмотрим типы задач, которые могут решаться с применением сетей Петри. Цель анализа сети Петри – получение ответа на вопрос о конкретной сети Петри.

7.3.1. Безопасность сети Петри

Одно из важнейших свойств сети Петри, которая должна моделировать реальное устройство, – безопасность. Позиция сети Петри является безопасной, если число фишек в ней никогда не превышает 1. Сеть Петри безопасна, если безопасны все позиции сети.

Определение. Позиция $p_i \in P$ сети Петри $S = (P, T, I, O)$ с начальной маркировкой μ является *безопасной*, если $\mu'(p_i) \leq 1$ для любой $\mu' \in R(C, \mu)$. Сеть Петри безопасна, если безопасна каждая ее позиция.

Безопасность – очень важное свойство для устройств аппаратного обеспечения. Если позиция безопасна, то число фишек в ней равно 0 или 1. Следовательно, позицию можно реализовать одним триггером.

В первоначальном определении сети Петри можно считать безопасными, поскольку переход не мог быть запущен, если не все из выходных позиций были пусты (а кратные дуги не были разрешены). Это объяснялось интерпретацией позиции как условия. Условие, будучи логическим высказыванием, может быть либо истинно (представляется фишкой в позиции), либо ложно (представляется отсутствием фишки). При этом кратные фишки не имеют никакой интерпретации. Таким образом, если интерпретировать сети как условия и события, маркировка каждой позиции должна быть безопасной.

Если позиция не является кратной входной или кратной выходной для перехода, ее можно сделать безопасной. К позиции p_i , которую необходимо сделать безопасной, добавляется новая позиция p_i' . Переходы, в которых p_i используется в качестве входной или выходной, модифицируются следующим образом:

если $p_i \in I(t_j)$ и $p_i \notin O(t_j)$, тогда добавить p_i' к $O(t_j)$;

если $p_i \in O(t_j)$ и $p_i \notin I(t_j)$, тогда добавить p_i' к $I(t_j)$.

Цель введения этой новой позиции p_i' – представить дополнительное условие (p_i пуста). Следовательно p_i и p_i' находятся в следующей зависимости: p_i имеет фишку, только если p_i' не имеет фишки и наоборот. Любой переход, удаляющий

фишку из p_i должен помещать фишку в p_i' , а всякий переход, удаляющий фишку из p_i' , должен помещать фишку в p_i . Начальная маркировка также должна быть модифицирована для обеспечения того, чтобы только одна фишка была либо в p_i , либо в p_i' . Такая принудительная безопасность возможна лишь для позиций, которые в начальной маркировке являются безопасными, и входная, и выходная кратность которых равна 0 или 1 для всех переходов. Позиция, имеющая для некоторого перехода выходную кратность 2, будет получать при его запуске две фишки и, следовательно, не может быть безопасной. На рис. 7.6 простая сеть Петри (а) преобразована в безопасную (б).

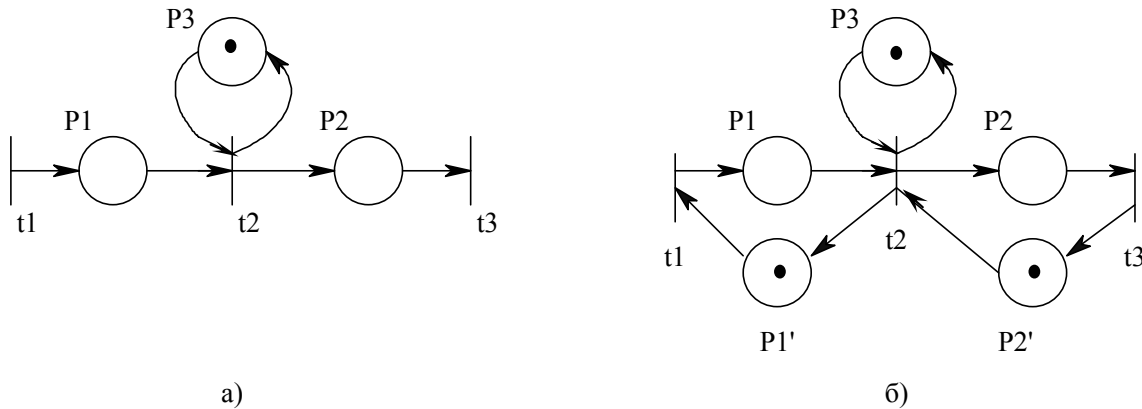


Рис. 7.6. а) сеть Петри, не являющаяся безопасной; б) безопасная сеть Петри, эквивалентная сети а)

7.3.2. Анализ живучести (сохранения) сетей Петри

Сети Петри можно использовать для моделирования систем распределения ресурсов. Например, сеть Петри может моделировать запросы распределения и освобождения устройств ввода-вывода в вычислительной системе. В этих системах некоторые функции могут представлять ресурсы. Группа из трех постстрочно печатающих устройств представляется позицией, имеющей в начальной маркировке три фишки. Запрос постстрочно-печатающего устройства – это переход, для которого данная позиция является входной; затем устройство освобождается переходом, для которого позиция постстрочно печатающих устройств является выходной.

Сети Петри такого типа должны обладать свойством живучести или сохранения. Для этого фишки, представляющие ресурсы, не должны создаваться или уничтожаться, общее число фишек в сети должно оставаться постоянным.

Определение. Сеть Петри $C = (P, T, I, O)$ с начальной маркировкой μ называется **строго сохраняющей**, если для всех $\mu' \in R(C, \mu)$:

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i).$$

Строгое сохранение требует, чтобы число входов в каждый переход должно равнялось числу выходов $|I(t_j)| = |O(t_j)|$. Если это условие не будет выполняться, то запуск перехода изменил бы число фишек в сети.

Рассмотрим график сети Петри (рис. 7.7), которая не является строго сохраняющей. В этой сети запуск перехода t_1 или t_2 уменьшает число фишек на 1, а запуск переходов t_3 или t_4 добавит фишку к маркировке. Эту сеть можно преобразовать в сеть строго сохраняющую (рис. 7.8).

Сеть Петри должна сохранять ресурсы, которые она моделирует. Однако взаимно однозначного соответствия между фишками и ресурсами нет.

Фишка может представлять собой один программный счетчик или какой-нибудь другой элемент или несколько ресурсов сразу. В этом случае фишка используется для создания кратных фишек путем запуска перехода с большим числом выходов, чем входов.

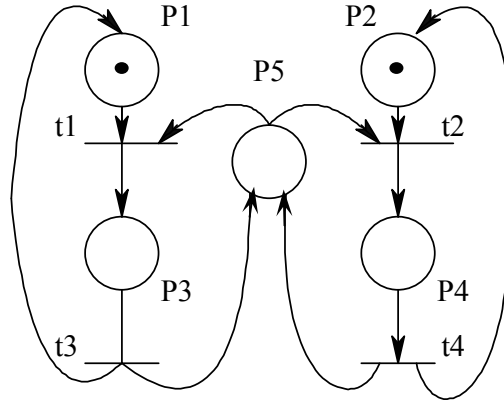


Рис. 7.7. Сеть Петри, не являющаяся строго сохраняющей

В общем случае следует определить **взвешивание** фишек. Взвешенная сумма для всех маркировок должна быть постоянной. Фишкам, не являющимся важными, можно присвоить вес 0; другим фишкам можно присвоить веса 1, 2, 3 или другое целое число. Фишка определяется ее позицией в сети. Следовательно, веса связаны с каждой позицией сети.

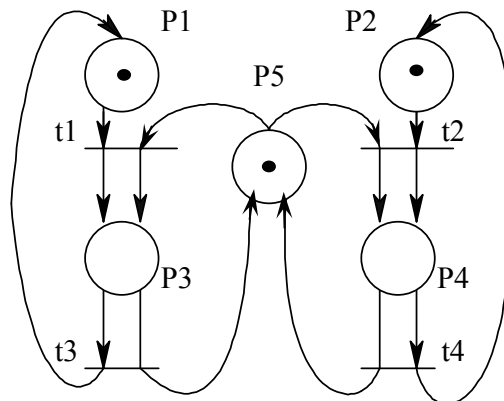


Рис. 7.8. Строго сохраняющая сеть Петри, эквивалентная сети, изображенной на рис. 7.7

Вектор взвешивания $w = (w_1, w_2, \dots, w_n)$ определяет вес w_i для каждой позиции $p_i \in P$.

Определение. Сеть Петри $C = (P, T, I, O)$ с начальной маркировкой μ называется сохраняющей по отношению к вектору взвешивания w , где $w = (w_1, w_2, \dots, w_n)$, при

$n = |P|$, $w_i > 0$, если для всех $\mu' \in R(C, \mu)$:

$$\sum_i w_i \cdot \mu'(p_i) = \sum_i w_i \cdot \mu(p_i).$$

Строго сохраняющая сеть Петри является сохраняющей по отношению к вектору взвешивания $(1, 1, \dots, 1)$. Все сети Петри являются сохраняющими по отношению к вектору взвешивания $(0, 0, \dots, 0)$.

7.3.3. Активность сети Петри

Рассмотрим систему, включающую два различных ресурса q и r и два процесса a и b . Если оба процесса нуждаются в обоих ресурсах им необходимо будет совместно использовать ресурсы. Для выполнения этого требуется, чтобы каждый процесс запрашивал ресурс, а затем освобождал его. Предположим, что процесс a сначала запрашивает ресурс q , затем ресурс r и, наконец, освобождает и q , и r . Процесс b работает аналогично, но сначала запрашивает r , а затем q (рис. 7.9).

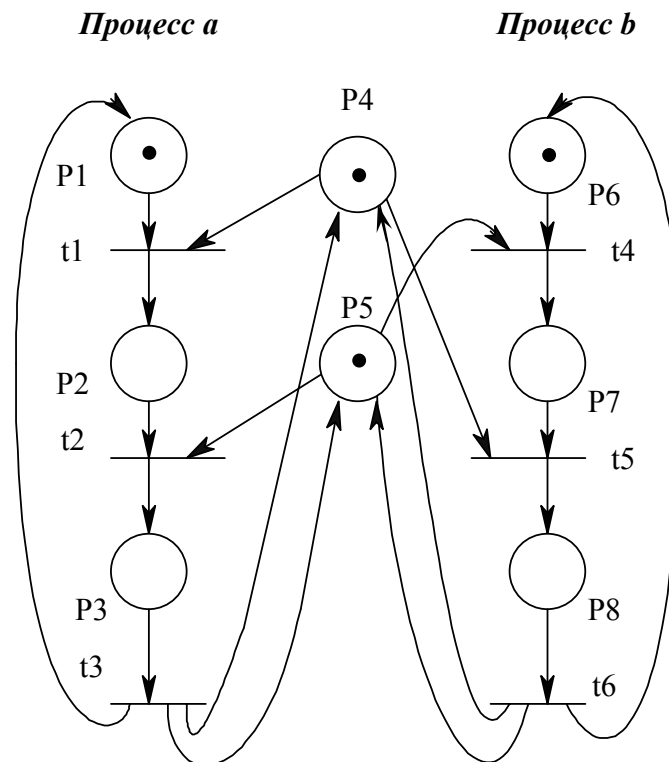


Рис. 7.9. Распределение ресурсов для случая двух процессов (a и b) и двух ресурсов (q (моделируется $P4$) и r (моделируется $P5$))

Начальная маркировка помечает ресурсы q ($P4$) и r ($P5$) доступными и указывает на готовность процессов a и b . Одним выполнением этой сети является $t_1 t_2 t_3 t_4 t_5 t_6$; другим $t_4 t_5 t_6 t_1 t_2 t_3$. Ни одно из этих выполнений не приводит к тупику. Однако рассмотрим последовательность, которая начинается переходами t_1, t_4 , а процесс a обладает ресурсом q , чтобы получить r ; процесс b обладает ресурсом r , чтобы получить q . Система заблокирована, то есть никакой процесс продолжаться не может.

Тупик в сети Петри – это переход (или множество переходов), которые не могут быть запущены. В сети Петри (рис. 7.9) тупик возникает, если нельзя запустить перехо-

ды t_2 , t_3 . Переход называется **активным**, если он не заблокирован (нетупиковый).

7.3.4. Достижимость и покрываемость

Определение. Задача достижимости. Для данной сети Петри $C = (P, T, I, O)$ с маркировкой μ и маркировки μ' определить: $\mu' \in R(C, \mu)$.

Задача достижимости – основная задача анализа сети Петри. Многие задачи анализа могут быть сформулированы в терминах такой задачи достижимости.

Множество достижимости сети Петри представляет собой дерево достижимости. Пусть имеется сеть Петри, представленная на рис. 7.10. Ее начальная маркировка – $(1, 0, 0)$. В этой начальной маркировке разрешены t_1 и t_2 . Чтобы рассмотреть все множество достижимости, определим новые вершины в дереве достижимости для других достижимых маркировок, получающихся в результате запуска каждого из этих двух переходов. Начальной (корневой) является вершина, соответствующая начальной маркировке. Дуга, помеченная запускаемым переходом, приводит из начальной маркировки к каждой из новых маркировок (рис. 7.11). Это частичное дерево показывает все маркировки, непосредственно достижимые из начальной маркировки.

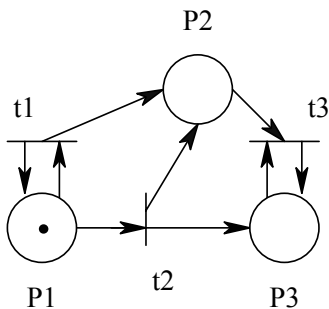


Рис. 7.10. Маркированная сеть Петри, для которой строится дерево достижимости

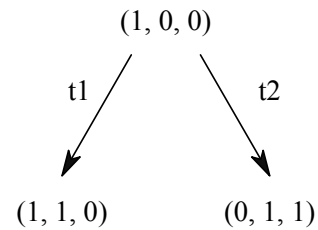


Рис. 7.11. Первый шаг построения дерева достижимости

Теперь необходимо рассмотреть все маркировки, достижимые из новых маркировок. Из маркировки $(1, 1, 0)$ можно снова запустить t_1 (получая $1, 2, 0$) и t_2 (получая $(0, 2, 1)$); из $(0, 1, 1)$ можно запустить t_3 (получая $0, 0, 1$). Это порождает дерево, изображенное на рис. 7.12. Начиная с трех новых маркировок, необходимо повторить этот процесс, порождая новые маркировки, которые нужно ввести в дерево, как показано на рис. 7.13.

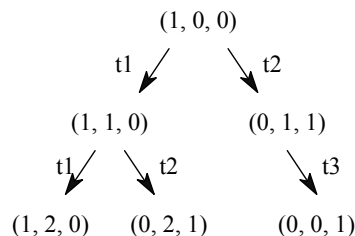


Рис. 7.12. Второй шаг построения дерева достижимости

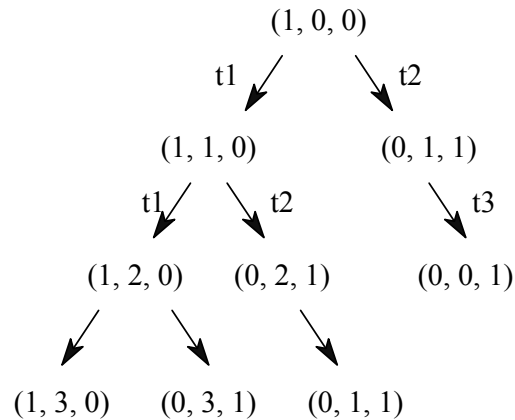


Рис. 7.13. Третий шаг построения дерева достижимости

Заметим, что маркировка $(0, 0, 1)$ пассивная; никакой переход в ней не является разрешенным, поэтому никакие новые маркировки из этой пассивной маркировки в дереве порождаться не будут. Кроме того, необходимо отметить, что маркировка $(0, 1, 1)$, порождаемая запуском t_3 в $(0, 2, 1)$ была уже порождена непосредственно из начальной маркировки запуском t_2 .

Если эту процедуру повторять снова и снова, каждая достижимая маркировка окажется порожденной. Однако получившееся дерево достижимости может оказаться бесконечным, так как будет порождена каждая маркировка из множества достижимости. Поэтому для любой сети Петри с бесконечным множеством достижимости соответствующее дерево также должно быть бесконечным. Даже сеть Петри с конечным множеством достижимости может иметь бесконечное дерево (рис. 7.14).

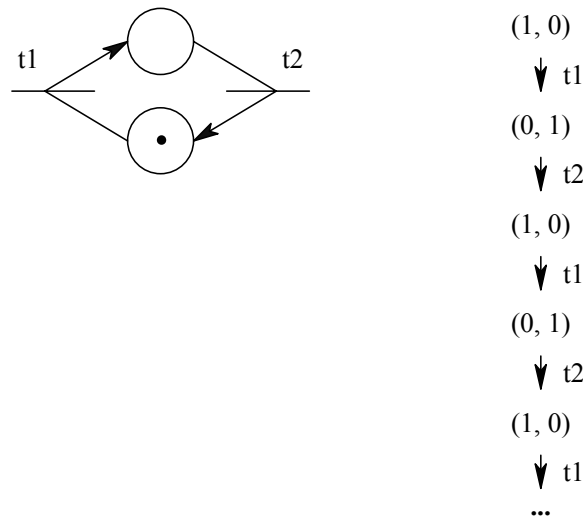


Рис. 7.14. Простая сеть Петри с бесконечным деревом достижимости

Дерево представляет все возможные последовательности запусков переходов. Всякий путь в дереве, начинающийся в корне, соответствует допустимой последовательности переходов.

Для получения дерева, которое можно считать полезным инструментом анализа необходимо найти средства ограничения его до конечного размера. Отметим, что если какое-то представление бесконечного множества конечно, то бесконечное множество маркировок должно отображаться в такое представление. В общем случае это

приведет к потере информации и, возможно, к тому, что некоторые свойства сетей Петри определить будет нельзя, но все зависит от того, как представление было получено.

Приведение к конечному представлению осуществляется несколькими способами. Прежде всего, необходимо использовать те средства, которые ограничивают введение новых маркировок (называемых *граничными* вершинами) на каждом шаге. Для этого могут вводиться пассивные маркировки, то есть маркировки, в которых нет разрешенных переходов. Такие пассивные маркировки называются *терминальными вершинами*. Другой класс маркировок – это маркировки, ранее встречавшиеся в дереве. Такие маркировки называются *дублирующими вершинами*: никакие последующие маркировки можно не рассматривать – все они будут порождены из места первого появления дублирующей маркировки в дереве. Таким образом, в дереве на рис. 7.13 маркировка $(0, 1, 1)$, получившаяся в результате выполнения последовательности t_1, t_2, t_3 , не будет порождать какие-либо новые вершины в дереве, поскольку она ранее встречалась в дереве в результате выполнения последовательности t_2 из начальной маркировки.

Каждая вершина может классифицироваться как граничная вершина, терминальная вершина, дублирующая вершина или как внутренняя вершина. *Граничными* являются вершины, еще не обработанные алгоритмом, а после обработки они могут стать терминальными, дублирующими или внутренними вершинами.

Алгоритм обычно начинается с определения начальной маркировки корнем дерева, т. е. граничной вершиной.

Пусть x – граничная вершина, которую необходимо обработать, тогда

1) если в дереве имеется другая вершина y , не являющаяся граничной, и с ней связана та же маркировка, $\mu(x) = \mu(y)$, то вершина x – дублирующая;

2) если для маркировки $\mu(x)$ ни один из переходов не разрешен, то x – терминальная вершина;

3) дуга, помеченная t_j , направлена от вершины x к вершине y . Вершина x переопределяется как внутренняя, вершина y – становится граничной.

Если все вершины дерева – терминальные, дублирующие или внутренние, то обработка данным алгоритмом останавливается.

Определение. Задача покрываемости. Для заданной сети Петри S с начальной маркировкой μ и маркировки μ' определить, существует ли такая достижимая маркировка $\mu'' \in R(S, \mu)$, что $\mu'' \geq \mu$. Маркировка μ'' покрывает μ' .

Это требование можно усложнить, если определять достижимость и покрываемость для множества маркировок, тогда можно перейти к задачам *достижимости множества* и *покрываемости множества*. Однако, если множество конечно, то такие задачи можно решить обычным многократным решением соответствующих задач достижимости и покрываемости для одной маркировки.

7.4. Моделирование алгоритмов с помощью сетей Петри

Моделирование алгоритмов с помощью сетей Петри имеет ряд особенностей. Одной из особенностей является свойственные сетям и их моделям параллелизм или одновременность. В модели сети Петри два разрешенных невзаимодействующих события могут происходить независимо друг от друга. Синхронизировать события, пока это не требуется моделируемой системе, нет необходимости. Но, когда синхрониза-

ция необходима, моделировать ее легко. Таким образом, сети Петри представляются идеальными для моделирования систем с распределенным управлением, в которых несколько процессов выполняется одновременно.

Другая важная особенность сетей Петри – это их асинхронная природа. В сети Петри отсутствует измерение времени. Обычно последовательности происходящих событий укладываются в различные интервалы времени, и это находит отражение в модели сети Петри, но в полной независимости от времени управления последовательностью событий. Структура сети Петри такова, что содержит в себе всю необходимую информацию для определения возможных последовательностей событий. Таким образом, событие «Завершение выполнения задания» должно следовать за соответствующим событием «Начало выполнения задания». Однако не требуется никакой информации, связанной с количеством времени, необходимым на выполнение задания.

Выполнение действий в сети Петри (или поведение моделируемой системы) рассматривается как последовательность дискретных событий. Порядок появления событий может быть одним из любых возможных, допускаемых основной структурой. Это приводит к явной недетерминированности в выполнении сети Петри. Если в какой-то момент времени разрешено более одного перехода, то любой из нескольких возможных переходов может стать «следующим» запускаемым. Выбор запускаемого перехода осуществляется недетерминированным образом, т. е. случайно. Эта особенность сети Петри отражает тот факт, что в реальной ситуации, где несколько действий происходят одновременно, возникающий порядок появления события неоднозначен: может возникнуть любая из множества последовательностей событий.

Основное положение теории относительности утверждает, что передача чего-либо не может быть мгновенной. Любая информация о возникновении события распространяется в пространстве со скоростью, ограниченной скоростью света c . Это означает, что если два события и произойдут одновременно, то порядок возникновения этих событий для двух разных наблюдателей может оказаться различным.

Для простоты обычно вводят следующее ограничение. Запуск перехода (и соответственно события) рассматривается как *мгновенное событие*, занимающее нулевое время, и возникновение двух событий одновременно невозможно. Моделируемое таким образом событие называется *примитивным*; примитивные события мгновенны и *неодновременны*.

Непримитивными называются такие события, длительность которых отлична от нуля. Они не являются неодновременными и, следовательно, могут пересекаться во времени. Так как выполнение большинства событий все же реально и длится некоторое время, то они являются непримитивными событиями и поэтому не могут должным образом моделироваться переходами в сети Петри. Однако непримитивное событие может быть представлено в виде двух примитивных событий: «Начало непримитивного события», «Конец непримитивного события» и условия «Непримитивное событие происходит» (рис. 7.15).

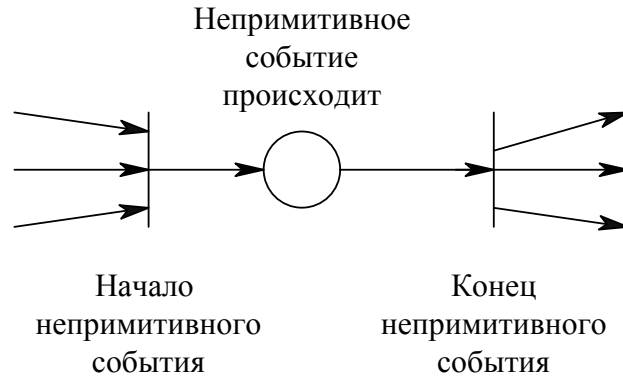


Рис. 7.15. Моделирование непримитивного события

При моделировании сложных вычислительных систем на нескольких иерархических уровнях сети Петри представляются в отдельные элементы сети целыми подсетями. Отдельными элементами в этом случае являются прямоугольники, представляющие непримитивные события, а планки представляют примитивные события. Такое моделирование вычислительной системы представлено на рис. 7.16.

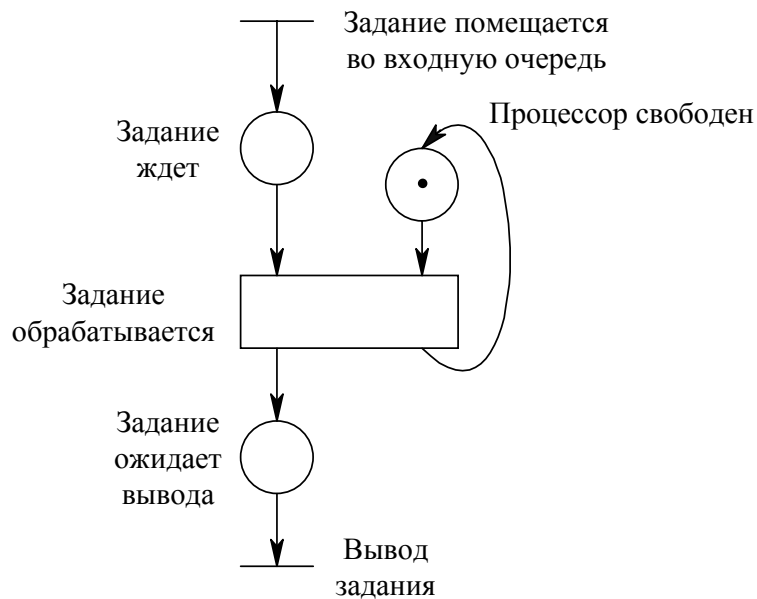


Рис. 7.16. Моделирование вычислительной системы с использованием непримитивного перехода

Недетерминированность и неодновременность запусков переходов при моделировании параллельной системы показываются двумя способами. Один из них представлен на рис. 7.17. В этой ситуации имеется два разрешенных перехода, которые в любом случае не влияют друг на друга. В число возможных последовательностей событий входит последовательность, в которой первым срабатывает один переход или последовательность, в которой первым будет запущен другой переход. Такая ситуация называется *одновременностью*.

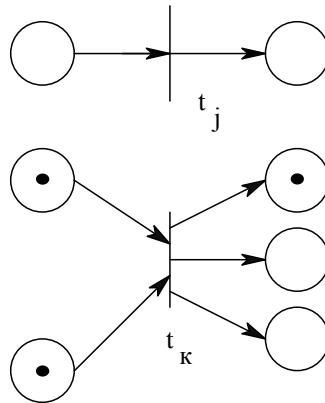


Рис. 7.17. Одновременность.

Эти два перехода могут быть запущены в любом порядке

Другая ситуация, в которой одновременное выполнение затруднено и которая характеризуется невозможностью одновременного возникновения событий, показана на рис. 7.18. Здесь два разрешенных перехода находятся в конфликте. Может быть запущен только один переход, так как при запуске он удаляет фишку из общего входа и запрещает другой переход.

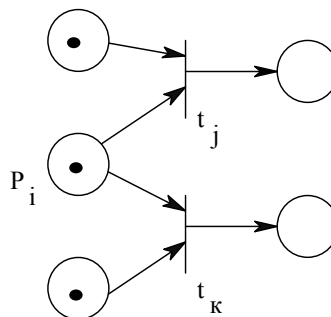


Рис. 7.18. Конфликт. Переходы t_j и t_k находятся в конфликте, т. е. запуск одного из них удаляет фишку из P_i и тем самым запрещает другой

Таким образом, рассмотренные ситуации требуют внимательного изучения моделируемых сетями Петри систем, чтобы правильно отображать их поведение.

Существуют определенные области, в которых сети Петри представляются идеальным инструментом для моделирования. К ним относят использование сетей Петри для моделирования аппаратного и программного обеспечения ЭВМ и других систем. При этом имеется возможность моделировать параллелизм довольно простым объединением подсистем, представленных сетями Петри, что делает сети Петри полезным инструментом моделирования сложной аппаратуры вычислительных систем. Вычислительные системы состоят из многих компонент, поэтому сети Петри также считают наиболее подходящим средством для представления таких систем.

Например, производительность вычислительных систем можно увеличить, если параллельно выполнять несколько функций. Тогда построение высокопроизводительной ЭВМ будет основано на использовании метода **конвейерной обработки**. Этот метод обработки подобен функционированию обычного сборочного конвейера и особенно удобен для работы с векторами и массивами. При моделировании работы конвейера применяют сети Петри. Конвейер представляется набором операций, кото-

рые могут выполняться одновременно. Когда операция k завершается, она передает свой результат операции $(k+1)$ и ожидает от операции $(k - 1)$ нового задания. Если каждая операция занимает t единиц времени и всего существует n операций, то завершение обработки одного операнда потребует nt единиц времени.

7.5. Расширенные сети Петри

В п. 7.4 отмечалось, что сети Петри могут быть использованы для моделирования самых различных систем, в том числе аппаратного и программного обеспечения ЭВМ. Очевидно, что сети Петри могут адекватно моделировать разные системы, однако могут существовать такие системы, которые нельзя должным образом моделировать сетями Петри, т. е. мощность моделирования сетей Петри имеет пределы.

Применение классических подходов и добавление дополнительных атрибутов позволили разработать сети различной целевой направленности, получившие название **расширенные**. Классификация расширенных сетей Петри приведена на рис. 7.19. Рассмотрим подробнее некоторые типы сетей Петри.

Ингибиторная сеть представляет собой сеть Петри, дополненную специальной функцией инцидентности $I_{IN} : P \times T \rightarrow \{0, 1\}$, которая вводит ингибиторные (запрещающие) дуги для тех пар (p, t) , для которых $I_{IN}(P, T) = 1$. Ингибиторные дуги связывают только позиции с переходами, на рисунках их изображают заканчивающимися не стрелками, а маленькими кружочками.



Рис. 7.19. Расширенная сеть Петри

Переход t в ингибиторных сетях может сработать, если каждая его входная по-

зиция, соединенная с переходом обычной дугой с кратностью $w(p, t)$ содержит не менее $w(p, t)$ фишек, а каждая входная позиция, соединенная с переходом t ингибиторной дугой (ее кратность всегда равна 1), имеет нулевую разметку.

Например, используя ингибиторную дугу, можно промоделировать оператор условного вычитания (если $x_i \neq 0$, то $x_i = x_i - 1$) и получить фрагмент ингибиторной сети (рис. 7.20).

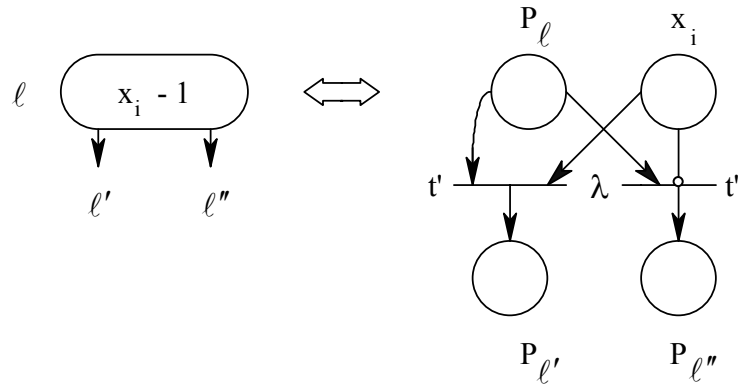


Рис. 7.20. Фрагмент ингибиторной дуги

Ингибиторные сети используются для разработки диагностических моделей средств вычислительной техники.

В приоритетных сетях вводят приоритеты срабатывания переходов. Если несколько переходов являются разрешенными, то срабатывает тот из них, который имеет наивысший приоритет. Такие сети используются для моделирования систем на уровне задач.

В структурированных сетях некоторые из переходов являются сложными. При их срабатывании запускается сеть другого уровня иерархии (рис. 7.21).

Срабатывание t_2 приводит к запуску сети другого уровня. Выполнение сложного перехода заключается в помещении во входную позицию по сети фишки. После выполнения сети фишка появляется в ее выходной позиции, затем формируются фишки в выходных позициях сложного перехода.

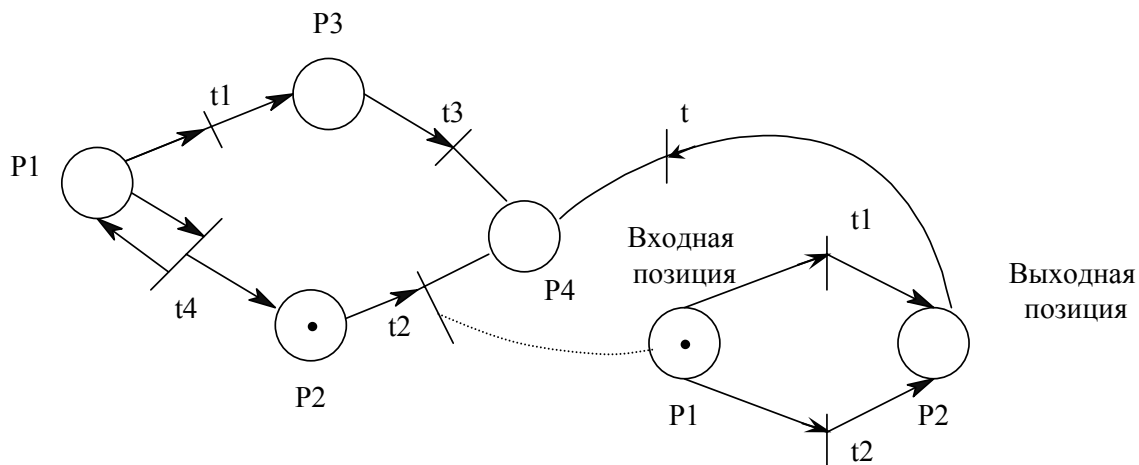


Рис. 7.21. Структурированная сеть Петри

Преобразование сети к виду, имеющему один вход и один выход, всегда возможно. Такие сети используются для моделирования модульных вычислительных

систем.

В цветных сетях вводится понятие цвета для фишек. В общем случае может быть n цветов. В вычислительной технике используются трехцветные сети ($n = 3$). Такие сети используются для моделирования аппаратных средств.

В сетях с изменяемой структурой кратность ребер не является постоянной.

В самомодифицируемых сетях кратность ребра может задаваться либо натуральным числом N , либо определяться количеством фишек, находящихся во входных позициях некоторого перехода.

Качественными характеристиками могут быть: отсутствие заикливаний в системе, достижение некоторого состояния системы (например, конечного).

Количественными характеристиками являются: время работы некоторого маршрута в программе, время прохождения сигнала в схеме и т. д.

Во временных сетях переходам ставится в соответствие их времена срабатывания, либо позициям ставится в соответствие времена нахождения фишек в позициях.

В стохастических сетях указанные характеристики являются вероятностными, т. е. вводится функция плотности вероятности времен срабатывания переходов или времен нахождения фишек в позициях.

Предикатные сети – это сети с логическим описанием состояния системы.

7.6. Сети Петри и регулярные языки

Методы анализа сетей Петри позволяют проводить исследования разрешимости таких основных задач, как **достижимость**. В частности, одна из областей, в которых рассматриваются вопросы достижимости – это теория формальных языков. Используя языки сетей Петри, можно перенести понятия и методы теории формальных языков на задачи для сетей Петри. И наоборот, методы сетей Петри могут оказаться весьма полезными для получения новых сведений о формальных языках.

Языком называется множество строк алфавита. В общем случае, языки могут быть бесконечными, следовательно, одной из проблем является представление языка. Для представления языков предложено два подхода. Один заключается в определении машины, которая порождает строку из языка, и любая строка из языка порождается ею. Другой подход подразумевает определение грамматики, которая указывает, как последовательным применением ее правил порождения получаются строки языка.

Ограничения, накладываемые на вид машин или грамматик, порождающих языки, определяют **классы языков**.

Традиционными классами языков являются: регулярный, контекстно-свободный, контекстно-связанный и языки типа 0, соответствующие конечным автоматам, магазинным автоматам, линейно-ограниченным автоматам и машинам Тьюринга. Каждый из классов языков порождается соответствующим классом автоматов. Это дает возможность установления связи теории сетей Петри с теорией формальных языков, чтобы определить класс языков сетей Петри как класс языков, порождаемых сетями Петри.

Рассмотрим в качестве примера конечные автоматы и регулярные выражения. Конечный автомат S есть пятерка (Q, A, δ, q_0, F) , где

Q – конечное множество состояний;

A – алфавит символов; A^* – множество всех строк из символов A , включая пустую строку: $A^* = A \cup \{\varepsilon\}$;

$\delta: Q \times A \rightarrow Q$ – функция переходов;

$q_0 \in Q$ – начальное состояние;

$F \subseteq Q$ – конечное множество заключительных состояний.

Функция переходов δ естественным образом обобщается на случай отображения из $Q \times A^*$ в Q . Язык $L(S)$, порождаемый конечным автоматом, – это множество строк над A^* , определяемое следующим образом:

$$L(S) = \{a \in A^* \mid \delta(q_0, a) \in F\}.$$

Всякому конечному автомату соответствует язык, а класс всех языков, порождаемых конечными автоматами, называется классом **регулярных** языков. Каждый автомат определяется своим языком. Если два конечных автомата имеют одинаковые языки, то они **эквивалентны**.

Основные понятия, используемые для получения регулярного языка по конечному автомату, применимы и к сетям Петри для образования теории языков сетей Петри.

В дополнение к сети Петри, определяемой множеством позиций и переходов (которые соответствуют множеству состояний и функций переходов автомата), необходимо определить **начальное состояние**, **алфавит** и **множество заключительных состояний**.

Начальное состояние сети Петри можно определить различными способами. Наиболее общепринятое определение – считать начальным состоянием произвольную маркировку μ .

Символы алфавита связаны с переходами, поэтому последовательность запусков переходов порождает строку символов языка. Связь символов с переходами осуществляется функцией **помечения**: $\delta: T \rightarrow A$.

Определение заключительных состояний сети Петри оказывает наибольшее влияние на язык сети Петри.

Одно из определений взято по аналогии с соответствующим понятием для автоматов – множество заключительных состояний F определяется как конечное множество заключительных маркировок.

Наиболее изученным классом формальных языков является класс регулярных языков. Эти языки порождаются регулярными грамматиками и конечными автоматами и характеризуются регулярными выражениями.

Всякий регулярный язык – это язык сети Петри. Доказательством этого служит то, что всякий регулярный язык порождается некоторым конечным автоматом, а всякий конечный автомат можно преобразовать в эквивалентную сеть Петри.

7.7. Преобразование конечного автомата в сеть Петри

Аппаратное обеспечение ЭВМ можно рассматривать на нескольких уровнях, и сети Петри могут моделировать каждый из этих уровней. На одном уровне ЭВМ построены из простых устройств памяти и вентилях (низкий уровень); на более высоком уровне в качестве основных компонент вычислительной системы используются функциональные блоки и регистры.

На низком уровне вычислительные системы могут быть описаны конечными автоматами. Автомат – это пятерка $(Q, X, Y, \delta, \lambda)$, где

Q – конечное множество состояний;

X – конечный входной алфавит;

Y – конечный выходной алфавит;

$\delta: Q \times X \rightarrow Q$ – функция следующего состояния;

$\lambda: Q \times X \rightarrow Y$ – функция выхода.

Автоматы часто представляют в виде графов переходов (рис. 7.22). В графе переходов состояния представляются кружками, являющимися вершинами. Дуга из состояния q_i в q_j , помеченная a / b означает, что находясь в состоянии q_i , автомат при входе a перейдет в состояние q_j , выдавая при этом символ b . Формально можно записать: $\delta(q_i, a) = q_j$ и $\lambda(q_i, a) = b$.

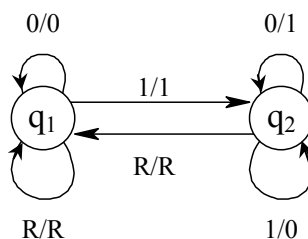


Рис. 7.22. Граф переходов для конечного автомата, вычисляющего дополнение до двух двоичного числа

Этот автомат преобразует двоичное число, представленное последовательностью бит, начиная с младшего, в дополнение до двух. В данном случае входной и выходной алфавиты состоят из трех символов: 0, 1, R. Автомат начинает работать в состоянии q_1 . Символ сброса (R) означает конец (или начало) числа и возвращает автомат в исходное состояние. Выход автомата для символа сброса является просто его повторением.

При представлении конечного автомата сетью Петри следует обратить внимание на связь между сетью Петри и внешними воздействиями. Моделирование взаимодействия с внешними воздействиями можно реализовать многими способами. В данной задаче моделируем это взаимодействие с помощью специального множества позиций. Позициями будут представлены каждый входной и выходной символы.

Допустим, что в сеть Петри помещается фишка в позицию, соответствующую входному символу, а затем фишка, появившаяся в позиции, соответствующей выходному символу, то есть удаляется из сети (рис. 7.23).

В качестве альтернативного подхода к моделированию входов и выходов сети могут быть использованы *переходы*. Для определения следующего входного символа следует выбирать входной переход и запускать его. Сеть Петри ответит (в конце концов) запуском соответствующего перехода из множества выходных переходов, связанного с соответствующим выходом. Затем будет запущен новый входной переход и т. д. (рис. 7.24).

Задав представление позиций, соответствующих символам входа и выхода, можно завершить построение модели системы конечных состояний. Текущее состояние отмечается фишкой, все остальные позиции пусты.

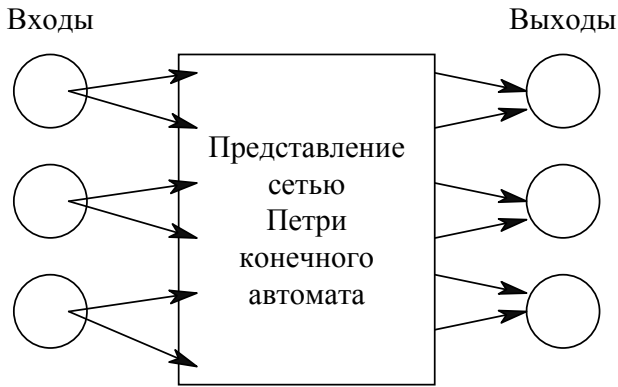


Рис. 7.23. Общий подход к моделированию

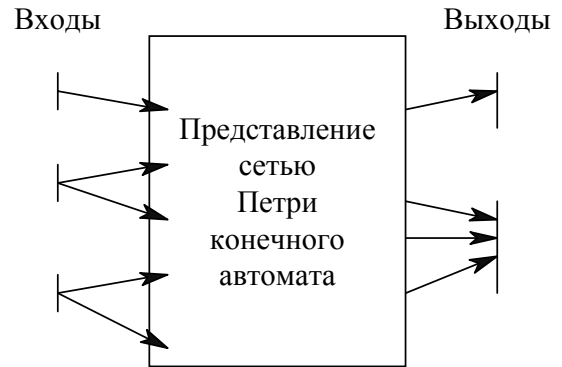


Рис. 7.24. Альтернативный подход

Теперь для определения переходов из состояния в состояние можно ввести **переходы сети** следующим образом. Для каждой пары (состояние и выходной символ) определяем переход, входными позициями которого являются позиции, соответствующие состоянию и входному символу, а выходными позициями – позиции, соответствующие следующему состоянию и выходу (рис. 7.25).

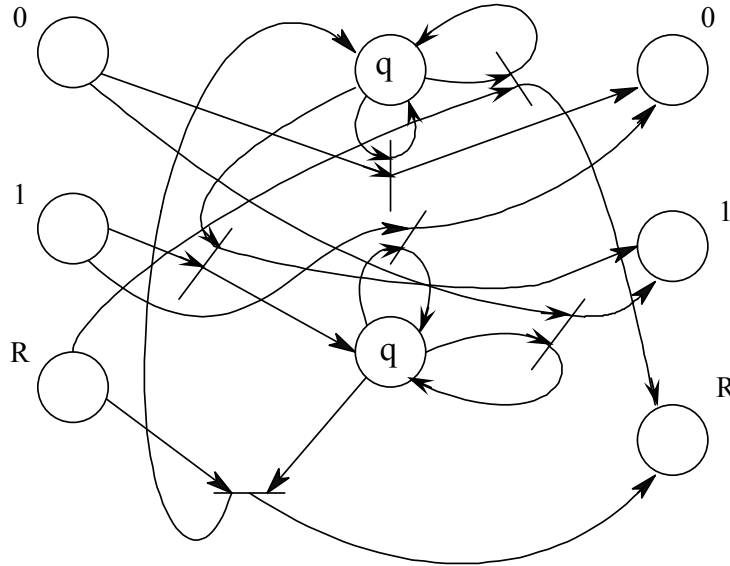


Рис. 7.25. Сеть Петри, эквивалентная автомату на рис. 7.22

Таким образом, для конечного автомата $(Q, X, Y, \delta, \lambda)$ определена сеть Петри (P, T, I, O) следующим образом:

$$P = Q \cup X \cup Y, \quad T = \{t_q, x \mid q \in Q \text{ и } x \in X\},$$

$$I(t_q, x) = \{q, x\},$$

$$O(t_q, x) = \{\delta(q, x), \lambda(q, x)\}.$$

Полученная сеть Петри является моделью конечного автомата.

7.8. Разработка модели сложения двух чисел с плавающей запятой

Модель сложения двух чисел с плавающей запятой разрабатывается в следующей последовательности:

- 1) выделить экспоненты обоих чисел;
- 2) сравнить экспоненты и изменить, если необходимо, должным образом порядок большей и меньшей экспонент;
- 3) сдвинуть точку в числе с меньшей экспонентой для их уравнивания;
- 4) сложить дроби;
- 5) нормализовать результат;
- 6) проверить экспоненту на переполнение и сформировать экспоненту и дробь результата.

Каждый из этих шагов может быть выполнен отдельным вычислительным блоком, где каждый отдельный операнд передается от блока к блоку для выполнения операции сложения. Очевидно, что необходимо выполнять до шести сложений одновременно.

Координацию работы различных блоков можно осуществлять несколькими способами. Обычно управление конвейерной обработкой является *синхронным*, то есть: время, отпущенное на выполнение каждого шага конвейера t , постоянно и фиксировано. В каждые t единиц времени результат каждого блока перемещается по конвейеру, чтобы стать входом для следующего блока. Однако обработка может быть приостановлена без необходимости при условии, если требуемое время будет изменяться от блока к блоку, а также внутри данного блока для различных входов. Например, для выполнения шага нормализации результата при сложении чисел с плавающей запятой может потребоваться различное количество времени в зависимости от того, на сколько разрядов необходимо произвести нормализующий сдвиг и в какую сторону.

В *асинхронном* конвейере в среднем процесс обработки может быть ускорен, если ввести сигнал завершения каждого шага конвейерной обработки к готовности передать свой операнд и получить новый.

Результат шага конвейерной обработки может быть послан на следующий шаг $(k + 1)$, как только шаг k выполнен, а блок $(k + 1)$ свободен.

Рассмотрим произвольный шаг конвейерной обработки. Пусть требуется поместить результат на входы и выходы в то время, когда они используются. Такая ситуация предполагает наличие регистров: каждый блок использует значение своего входного (буферного) регистра для вычисления значения выходного (буферного) регистра. Очевидно, что пока входной регистр блока не будет очищен путем пересылки содержимого во входной регистр следующего блока, новое входное значение не может появиться в его входном регистре. Таким образом, для управления блоком k конвейера необходимо знать, когда выполняются следующие условия:

- входной регистр заполнен;
- входной регистр пуст;
- выходной регистр заполнен;
- выходной регистр пуст;
- блок занят;
- блок свободен;

- пересылка осуществлена.

На рис. 7.26 и 7.27 показано, как можно промоделировать асинхронный конвейер такого типа. На рис. 7.26 приведена блок-схема конвейера, моделируемого сетью Петри (рис. 7.27).

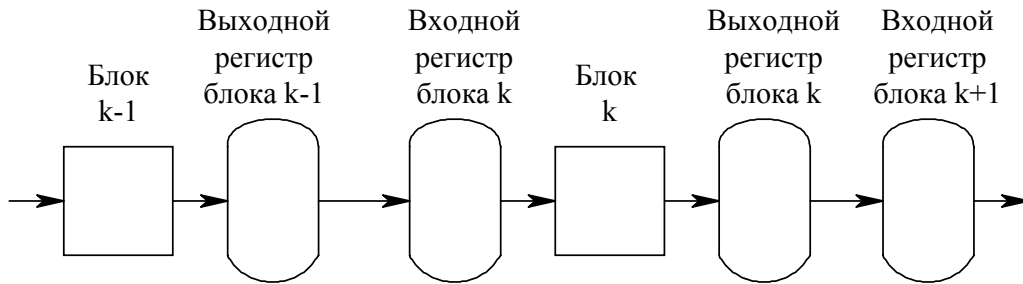


Рис. 7.26. Блок-схема устройства управления асинхронной ЭВМ с конвейерной обработкой

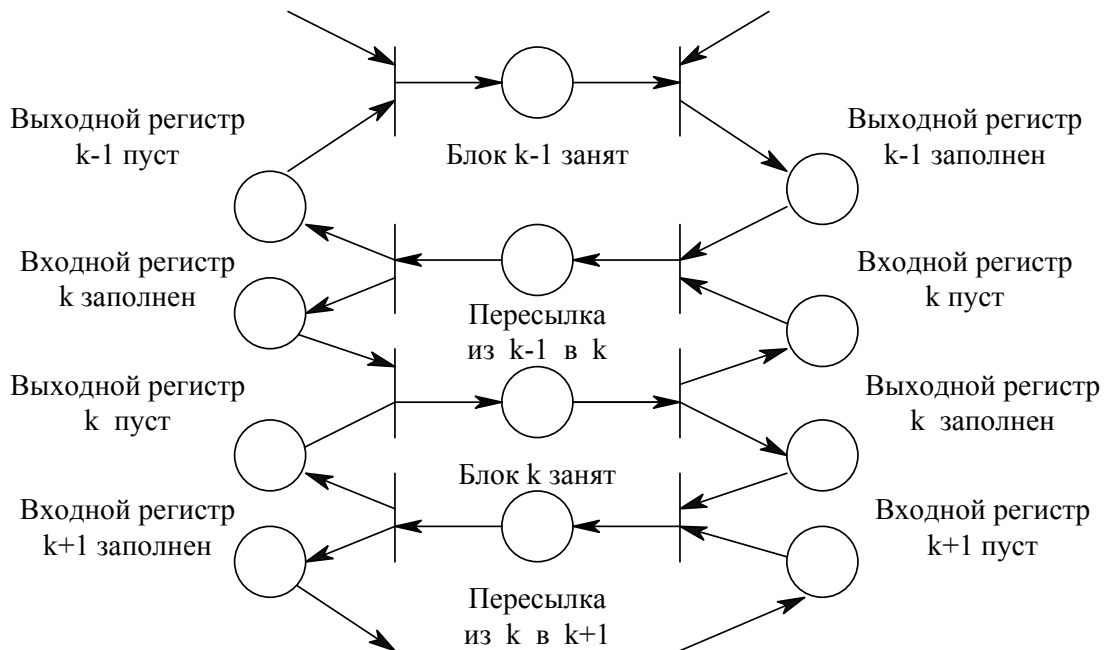


Рис. 7.27. Модель сети Петри устройства управления асинхронной ЭВМ с конвейерной обработкой

Отметим, что эта модель моделирует реальную работу блоков конвейера как непримитивных событий. Это позволяет нам проигнорировать на данном уровне конкретные детали того, что происходит в блоках, и сосредоточиться на их правильном взаимодействии. Каждая операция также может промоделироваться сетью Петри. Затем сети Петри для каждого блока можно объединить в обобщенную сеть Петри (рис. 7.27) моделируемой системы. Такая возможность моделирования системы на нескольких различных уровнях абстракции, т. е. иерархическим образом, может быть весьма полезна.

Контрольные вопросы

1. Основные характеристики сетей Петри.
2. Способы представления сетей Петри.
3. Расширенные сети Петри.
4. Маркировка сетей Петри.
5. Работа сети Петри.
6. Построение графов сетей Петри.
7. Моделирование алгоритмов с помощью сетей Петри.

ЗАКЛЮЧЕНИЕ

Понятие структурной теории автоматов, методы структурного синтеза автоматов и методы анализа сетей Петри, оказываются мощным инструментом моделирования динамических систем. При этом в каждом конкретном случае целесообразно применение одной из их модификаций, заключающейся для традиционных приложений во введении ограничений, накладываемых на функционирование элементов сети.

Особый интерес представляют вопросы построения и исследования моделей в системах искусственного интеллекта, для которых необходимо создание специальной интеллектуальной надстройки, ориентированной на более универсальный и гибкий аппарат. В качестве такового аппарата может использоваться формализм алгебраических сетей Петри. В этом направлении есть много нерешенных проблем, в том числе и вопросы применения способов программирования алгебраических сетей. В целом сети Петри представляют собой перспективную область исследований, в которой соединяются, обогащая друг друга, усилия специалистов самых разных направлений.

Приложение 1

Теория автоматов относится к одному из разделов кибернетики, объектом исследования которого является информация. Для описания процессов преобразования информации в автоматах используется специальный математический аппарат логической алгебры и различных ее модификаций, в том числе логико-алгебраических моделей (моделей логической алгебры). Автоматы обычно описываются на основе лингвистической концепции (построения автомато-лингвистических моделей) и теоретико-множественного подхода (построенная модель рассматривается как некоторая совокупность операций над множествами). Автоматные и лингвистические модели строятся на базе теории формальных грамматик.

Согласно лингвистической концепции автомат представляется в виде некоторого устройства, определяющего допустимость подаваемых на его вход слов в соответствии с заложенными в него правилами. Поэтому автоматы могут быть интерпретированы как распознающие устройства, которые определяют допустимость входных слов с позиций заложенных в них грамматик.

1. Грамматика

Язык. Язык задается следующим образом:

- 1) фиксируется алфавит как набор символов (букв);
- 2) определяются правила, как из букв образовывать выражения (слова).

Правила, определяющие предложения языка, называют его *грамматикой*, а комбинации символов, образующие грамматические единицы, – *фразами языка*.

Среди фраз выделяют имена, предложения и функторы. *Имя* называет некоторый объект. *Предложение* выражает утверждение. *Функтор* – это средство соединения фраз с целью образования других фраз.

Основные виды функторов: операторы (преобразуют имена в имена); глаголы (преобразуют имена в предложения); коннекторы (преобразуют предложения в предложения). Некоторое употребление функторов представлено в таблице.

Таблица

Употребление функторов

Функтор	Обозначение	Смысловое значение
Бинарные коннекторы	\rightarrow	Если _____ 1, то _____ 2
	\rightarrow \leftarrow или и	_____ 1 тогда и только тогда, когда _____ 2 _____ 1 или _____ 2 _____ 1 и _____ 2
Унарные глаголы	\vdash \dashv	_____ утверждается _____ отвергается
Бинарные глаголы	\equiv $=$ \leq \subseteq	_____ 1, есть то же самое, что _____ 2 _____ 1 равно _____ 2 _____ 1 предшествует _____ 2 _____ 1 включено в _____ 2

Функтор	Обозначение	Смысловое значение
Унарные операции	\neg	(операция отрицания)
Бинарные операторы	\supset	______1 имплицирует ______2 (операция импликации)
	\Leftrightarrow	$(\text{_____}_1 \supset \text{_____}_2) \wedge (\text{_____}_2 \supset \text{_____}_1)$ (операция эквиваленции)
	\vee	______1 ad ______2 (сложение, дизъюнкция)
	\wedge	______1 con ______2 (умножение, конъюнкция)

При $n = 1, 2, 3$ n -арный функтор называется соответственно унарным, бинарным, тернарным.

В автоматах для задания преобразования информации применяются различные формальные языки:

- язык регулярных функций;
- язык исчисления предикатов;
- язык временных логических функций;
- язык логических алгоритмов и т. д.

Пример. В формальных системах структура языка обычно связана с высказываниями, подразумеваемыми смыслом языка. Под *высказыванием* понимают всякое утверждение, о котором можно вполне определенно и объективно сказать, истинно оно или ложно. Однако наряду с высказываниями встречаются выражения, грамматически имеющие форму высказываний, но содержащие предметные переменные некоторых множеств. Такие выражения называются *предикатами* или *функциями-высказываниями*. Подобные выражения можно получить из любых высказываний, заменив в них обозначения предметов предметными переменными множеств, к которым принадлежат эти предметы. Поэтому под *предикатом* (praedicatum, означает «сказуемое») также понимают переменное высказывание, истинное значение которого зависит от параметров (переменных).

Например, предложение «2 – простое число» есть высказывание. Если «2» заменить предметным переменным n множества натуральных чисел, то полученное выражение « n – простое число» грамматически имеет ту же форму, но не является высказыванием.

Словарь. Конечное множество элементов называют *словарем*, элементы словаря – *символами*, а последовательности символов словаря – *цепочками* или *предложениями*. Тогда *языком* будет называться множество предложений.

Пример. Пусть V – словарь, L – язык над словарем V , а V^* – множество всех возможных цепочек, составленных из символов словаря V .

Тогда, если $V = \{a, b, c\}$, получим $V^* = \{\varepsilon, a, b, c, aa, ab, cba, csa, \dots\}$, где ε – пустая цепочка, т. е. цепочка, вовсе не содержащая символов. Очевидно, что хотя V конечно, V^* – бесконечное счетное множество. Язык – это просто некоторое подмножество V^* : $L \subseteq V^*$. Всего языков над словарем V (как подмножеств счетного множества V^*) бесконечное число.

Алфавит. Если V^* – множество, то и словарь V также представляет собой множество. Поэтому множество V часто называют *алфавитом*, а любое множество цепочек $L \subseteq V^*$ называют *формальным языком*, определенным на алфавите V .

Итерации алфавита V

Пусть B и C – некоторые подмножества множества V^* , а V_1 и V_2 – цепочки этого множества V^* .

Произведением множеств B и C (не путать с декартовым произведением, элементами которого в данном случае были бы не цепочки, а упорядоченные пары цепочек. Обозначается \times , см. далее в п. 3) называется множество D цепочек, являющихся конкатенацией (слиянием) цепочек из B и C , т. е.

$$D = BC = \{V_1 V_2 \mid V_1 \in B, V_2 \in C\}.$$

Итерацией алфавита V называют множество всех цепочек алфавита

$$V^* = \bigcup_{n=0}^{\infty} V^n,$$

где V^* – множество всех цепочек над алфавитом V , включая пустую цепочку ε ; V^n – степень алфавита, определяемая как $V^n = V^{n-1} V$, для каждого $n \geq 1$, если множество, состоящее из пустой цепочки ε , обозначить через V^0 .

Усеченной итерацией алфавита V называют

$$V^+ = \bigcup_{n=1}^{\infty} V^n,$$

где V^+ – множество всех цепочек над алфавитом V , без пустой цепочки ε .

Ассоциативные исчисления

Пусть даны пары цепочек $(P_1, Q_1), (P_2, Q_2), \dots, (P_n, Q_n)$ из $V^* \times V^*$. Если V_1 и V_2 – цепочки множества V^* , то цепочку V_1 называют подцепочкой цепочки V_2 , когда существуют такие цепочки X и Y из V^* , что $V_2 = X V_1 Y$.

Соотношения Туэ. Соотношениями Туэ называют правила, согласно которым любой цепочке $V_1 = X P_i Y$ из множества V^* будет ставиться в соответствие цепочка $V_2 = X Q_i Y$ из того же множества V^* ($i = 1, 2, \dots, n$). Такие соотношения и приводят к так называемым ассоциативным исчислениям.

Направление и порождение. Соотношения Туэ являются двусторонними, если цепочка V_1 является смежной по отношению к цепочке V_2 , и наоборот, цепочка V_2 является смежной по отношению к цепочке V_1 . Однако бывают соотношения, в которых вводится *направление*.

Соотношения, содержащие направление, называют *полусоотношениями Туэ* или *продукциями* и обозначают: $(P_1 \rightarrow Q_1), (P_2 \rightarrow Q_2), \dots, (P_n \rightarrow Q_n)$. Например, если имеется набор продукций цепочек V_1 и V_2 , то говорят, что V_1 порождает V_2 , или говорят, что цепочка V_2 непосредственно порождается из цепочки V_1 , и обозначается это как

$$V_1 \Rightarrow V_2,$$

при условии существования таких цепочек X и Y , что $V_1 = X P_i Y$, $V_2 = X Q_i Y$, а $(P_i \rightarrow Q_i)$ – продукция из данного набора.

Правило. Правило (или продукция) – это упорядоченная пара цепочек символов (α, β) . В правилах очень важен порядок цепочек, поэтому их часто записывают в виде $\alpha \rightarrow \beta$. Такая запись читается как « α порождает β » или « β по определению есть α ».

Например, P – множество правил (продукций) грамматики вида $\alpha \rightarrow \beta$, где $\alpha \in V^+$, $\beta \in V^*$.

2. Знаки, знакосочетания и кванторы

Знаки. К знакам относят: логические знаки (\vee, \neg, \dots); буквы; специальные знаки ($=, \in, \dots$).

Знакосочетания. Знакосочетание – последовательность знаков, написанных рядом друг с другом, причем некоторые знаки, отличные от букв, могут быть соединены линиями, идущими над строкой и называемыми *связями*.

Термами называют знакосочетания, составленные по некоторым правилам математической теории. *Термы* – это знакосочетания, изображающие объекты (*предметы*), а *соотношения* – формулы, изображающие *утверждения*, которые можно делать об этих предметах.

Примеры:

1) *предметы* могут изображать буквы или сводиться к одной букве (если задано условие, что «знакосочетание A есть буква»);

2) если B – *утверждение*, то $\neg B$, так называемое *отрицание* этого утверждения B , также есть *утверждение*, которое читается: «не B » (если задано условие, что в последовательности существует знакосочетание B , предшествующее A , такое, что A есть $\neg B$);

3) если A – *соотношение*, то «не (A) » можно писать вместо $\neg A$;

4) если A и B – *соотношения*, то можно писать « (A) или (B) » вместо $\vee AB$, а также $(A) \Rightarrow (B)$ вместо $\Rightarrow AB$;

5) если A – алфавит, то под *примитивным термом* понимается любая конечная последовательность знаков из алфавита A вида $\varphi(x_1, \dots, x_m)$, где φ обозначает m – местный функтор ($m = 1, 2, \dots$), а x_1, \dots, x_m – свободные индивидуальные переменные.

Квантор. Если A – знакосочетание и x – буква, то знакосочетание $(\exists x)A$ обозначается (читается) «существует такое x , что A ». Знакосочетание $(\forall x)A$ читается «для всех x A » или «каково бы ни было x , A ». Символы \exists и \forall соответственно *квантор существования* и *квантор общности*. (\exists, \forall – перевернутые первые буквы английских слов Exist – «существует» и All – «все»).

Например, пусть запись $A(x)$ означает, что x обладает свойством A , тогда запись $\forall x A(x)$ обозначает утверждение «все x обладают свойством A », а запись $\exists x A(x)$ означает, что «существует по крайней мере один предмет x , обладающий свойством A ».

Символ $(\exists x)$ также называется квантором существования по переменной x , его читают: «существует x такой, что ...» или «для некоторого x ...». Символ $(\forall x)$ называется квантором общности по переменной x , его читают: «для всех x » или «для каждого x ».

3. Множество

Множество. Под *множеством* (синонимы: совокупность, класс, система) понимается некоторая совокупность объектов, которые называются *элементами* этого множества.

Способы задания множеств:

1) с помощью перечисления элементов, которые записываются внутри фигурных скобок через запятую. Например, $X = \{x_1, x_2, x_3\}$;

2) с помощью характеристического свойства – общего свойства объектов, из которых образовано множество: $X = \{x \mid x \text{ обладает свойством } Q\}$, где вертикальная

черта после x означает «таких, что», т. е. X – множество элементов таких, что они обладают свойством Q .

Множество X называется *подмножеством* множества Y (иначе – X входит в Y), если и только если каждый элемент множества X принадлежит множеству Y . Обозначение: $X \subseteq Y$ или $Y \supseteq X$.

Например, если каждый элемент множества X принадлежит множеству Y , то записывают $X \subset Y$ и говорят, что X является подмножеством множества Y . Отношение \subset называется включением.

Операции над двумя множествами X и Y :

1. Объединение $X \vee Y$.
2. Пересечение $X \wedge Y$.
3. Дополнение \bar{X} и \bar{Y} .
4. Разность $X \setminus Y$.
5. Произведение $X \times Y$.

Декартовым или прямым *произведением* $X \times Y$ называется множество всех упорядоченных пар, первый и второй компоненты которых принадлежат соответственно множествам X и Y . Множество X называется первым, а множество Y вторым сомножителем произведения X и Y . Меняя ролями множества X и Y , получаем декартово произведение $Y \times X$, которое отлично от декартова произведения $X \times Y$, если $X \neq Y$.

Любое подмножество $P \subseteq X \times Y$ произведения множеств X и Y называется *бинарным соответствием* из X в Y . При этом множество X называется *областью отправления*, а множество Y – *областью прибытия* соответствия P .

Иногда для определения операции произведения множеств используют понятие *кортежа (вектора)*. Вместе с понятием кортежа вводится понятие *компонента* (координаты).

Регулярные и нерегулярные множества. Класс множеств цепочек над конечным словарем V называют *регулярными множествами*.

Пусть V_1 и V_2 – множества цепочек, тогда на этих множествах цепочек можно определить следующие *операции*:

1. Объединение V_1 и $V_2 = \{\alpha \mid \alpha \in V_1\}$ или $\alpha \in V_2$.
2. Конкатенация (произведение, склеивание): $V_1 \cdot V_2 = \{\alpha\beta \mid \alpha \in V_1, \beta \in V_2\}$.

Знак конкатенации обычно опускается.

Например, $V_1 = \{abc, ba\}$, $V_2 = \{b, cb\}$. $V_1 V_2 = \{abcb, abccb, bab, bacb\}$.

Обозначим через V^n произведение n множеств V , т. е. $V^n = VV \dots V$ при $V^0 = \{\varepsilon\}$, где ε – пустая цепочка.

Например, $V_1 = \{abc, ba\}$, $V_1^2 = \{abcbabc, abcbba, baba, baabcb\}$.

3. Итерация: $V^* = V^0 \vee V^1 \vee V^2 \vee \dots$

Например, $V = \{a, bc\}$, $V^* = \{\varepsilon, a, bc, aa, abc, bc, bc, bca, aaa, aabc, \dots\}$.

Правила для класса регулярных множеств над конечным словарем V :

1. \emptyset – регулярное множество;
2. $\{\varepsilon\}$ – регулярное множество;
3. $(\forall a \in V) \{a\}$ – регулярное множество;
4. Если A и B – регулярные множества, то регулярны:
 - объединение $A \vee B$;
 - конкатенация AB ;

- итерации A^* и B^* .

Если множество не может быть построено конечным числом правил для регулярных множеств, то оно *нерегулярно*.

Примеры:

1) регулярные множества: $\{ab, ba\}^* \{a, a\}$; $\{b\} (\{c\} \vee \{\alpha, ab\}^*)$;

2) нерегулярные множества: $\{a^n b^n \mid n > 0\}$; $\{\alpha \mid \text{в цепочке } \alpha \text{ количества вхождения символов } a \text{ и } b \text{ совпадают}\}$.

4. Функция

Функция. Если X и Y – два множества, то отображением множества X в множество Y называют *функцию* f , у которой область отправления (равная области определения) равна X , а область прибытия равна Y . Можно также и по-другому утверждать (читать):

1) «функция f определена на X и принимает свои значения в Y »;

2) «пусть f есть отображение X в Y »;

3) «пусть дано отображение $f: X \rightarrow Y$ » или проще: «пусть $f: X \rightarrow Y$ ».

Термины *отображение*, *функция*, *преобразование* всегда будут иметь одинаковый смысл, а запись

$$f: X \rightarrow Y$$

будет указывать (читается), что « f – отображение, определенное на множестве X со значениями из множества Y ».

Множество X называется *областью определения* или *доменом* функции f . Множество Y называется *областью значения* или *диапазоном* функции f .

Функция f , определенная на множестве X и принимающая значения из множества Y , называется также *отображением* X в Y .

Домен и диапазон называют соответственно *первой* и *второй проекциями* f и обозначаются $pr_1 f$ и $pr_2 f$.

Отображение f множества X в множество Y называется *отображением* X на Y , если $pr_2 f = Y$. Отображение f множества X в множество Y называется *взаимно-однозначным*, если образами двух любых различных элементов множества X являются различные элементы множества Y .

Пример. Пусть X – некоторое множество элементов информации, представленных тем или иным образом, Y – другое множество элементов информации, а f – функция преобразования. Тогда преобразователь информации можно представить устройством, реализующим отображение $f: X \rightarrow Y$ одного множества на другое.

Предметный указатель

Абстрактная теория	11, 33, 36	Машина Тьюринга	40
Автомат	18	Место	59
- абстрактный	33, 51, 53	- конечное	59
- асинхронный	11	- начальное	59
- второго рода	12	- основное	61
- детерминированный	37, 45	- предосновное	61
- дискретный	10	- разделяющее	59
- конечный	9, 37	Моноид	15
- Мили	12	Продукция	13, 17, 18
- Мура	12	Распознающая машина	18
- недетерминированный	40, 45	Регулярное событие	57
- первого рода	12	Словарь терминальный	26
- синхронный	11	- нетерминальный	26
- цифровой	10	Соотношения Туэ	16
Алфавит	10, 14	Структурная теория	12
- входной	37	Таблица переходов	32, 33
- выходной	37	- выходов	32
- состояний	37	- совмещенная	33
Входной сигнал	11	Условие автоматности	52, 53, 78
Выходной сигнал	11	Функция выходов	32
- канал	11	- заключительного состояния	73, 76
Граматики	13	- переходов	32
- контекстно-свободные	19	Цепочка	14
- непосредственных составляющих	19	- правильная	13
- порождающие	14	- пустая	14
- распознающие	14	- смежная	16
- формальные	10	Частичный автомат	48, 51
Граф автомата	34	Частичное отображение	30
Двоичное кодирование	32	Язык	19, 20, 39
Итерация	15	- контекстно-свободный	46, 47
Каноническое множество	56	- программирования	13
Канонический способ отметок	59		
Класс	16, 17		
Комбинационный синтез	13		
Конкатенация	14		

Библиографический список

1. Алгебраическая теория автоматов, языки и полугруппы. / Под ред. М. А. Арбиба М.: Статистика, 1975. – 120 с. , ил.
2. Аперидические автоматы. /Под ред. В. И. Варшавского. – М.: Наука, 1976. – 423 с.
3. Баранов С. И. Синтез микропрограммных автоматов. Л.: Энергия, 1979. –232 с., ил.
4. Букреев И. Н. и др. Микроэлектронные схемы цифровых устройств. – М. : Сов. радио, 1975. – 368 с.
5. Гилл А. Линейные последовательностные машины. – М. : Наука, 1974. – 288 с.
6. Гинзбург С. Математическая теория контекстно–свободных языков. – М. : Мир, 1970. –328 с.
7. Гладкий А. В. Формальные грамматики и языки. – М. : Наука, 1973. – 368 с.
8. Глушков В. М. и др. Логическое проектирование дискретных устройств. – Киев: Наукова думка, 1987. – 264 с.
9. Глушков В. М. Синтез цифровых автоматов. – М. : Физматгиз, 1962. – 476 с. , ил.
10. Закревский А. Д. Алгоритмы синтеза дискретных автоматов. – М. : Наука, 1971. – 511 с.
11. Каган Б. М. Электронные вычислительные машины и системы. – М. : Энергоатомиздат, 1985. – 306 с.
12. Котов В. Е. Сети Петри. – М. : Наука, 1984. – 158 с.
13. Кудрявцев В. Б. Введение в теорию автоматов. – М. : Наука, 1985. – 319 с. , ил.
14. Кузин Л. Т. Основы кибернетики. Т. 2. Энергия, 1979. – 584с.
15. Лазарев В. Г. , Пийль Е. И. Синтез управляющих автоматов. – М. : Энергия, 1978. – 408 с.
16. Леснин А. А. и др. Сети Петри в моделировании и управлении. – Л. : Наука, 1989. – 133 с.
17. Логическое проектирование БИС /Под ред. В. А. Мищенко– М. : Радио и связь, 1984. – 311 с.
18. Мелихов А. Н. Ориентированные графы и конечные автоматы. – М. : Наука, 1971. – 416 с.
19. Периодические автоматы / Под ред. В. И. Варшавского. – М. : Наука, 1976. – 178 с.
20. Питерсон Д. Теория сетей Петри и моделирование систем. – М. : Мир, 1984. – 264 с.
21. Пospelов Д. А. Логические методы анализа и синтеза схем. – М. : Энергия, 1974. – 368 с.
22. Рабинович З. Л. Основы теории элементных структур ЭВМ. – М. : Радио и связь, 1982. – 279 с.
23. Савельев А. Я. Прикладная теория цифровых автоматов. –М. : Высшая школа, 1987. –272с. , ил.
24. Чирков М. К. Основы общей теории конечных автоматов. – Л. : ЛГУ, 1975. –280 с., ил.

Учебное издание
Захаров Николай Григорьевич
Рогов Виктор Николаевич
Синтез цифровых автоматов
Учебное пособие
Редактор М. В. Леонова
Подписано в печать 30.10.2003. Формат 60×84/16.
Бумага офсетная. Печать трафаретная. усл. печ. л. 8,00.
Уч.-изд. л. 8,00. Тираж 100 экз. заказ.
УлГТУ
432027, г. Ульяновск, ул. Сев. Венец, д. 32.
Типография УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, д. 32.