МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Санкт-Петербургский
государственный университет аэрокосмического приборостроения

А. М. Лупал

ТЕОРИЯ АВТОМАТОВ

Учебное пособие

Санкт-Петербург
2000

УДК 519.7(075)
ББК 32.815
Л85

Лупал А. М.

Л77 Теория автоматов: Учеб. пособие/ СПбГУАП. СПб., 2000. 119 с.: ил.
ISBN 5–8088–0044–7

В пособии приводятся основные понятия теории алгоритмов, раскрывается связь между алгоритмами и вычислительными машинами и различия между процессами, протекающими в машинах Тьюринга и автоматах фон Неймана. Рассматриваются также основы теории конечных автоматов, формальные методы проектирования автоматов на основе абстрактного и структурного синтеза, явление состязаний элементов памяти и гонок в структурном автомате, методы кодирования состояний автоматов и синхронизация автоматов. Приводятся примеры модификаций элементарных абстрактных и структурных автоматов в асинхронном и синхронизируемом исполнении.

Учебное пособие предназначено для студентов дистанционной формы обучения по специальности “Вычислительные системы, комплексы и сети” и может быть использовано студентами дневного и вечернего факультета, обучающимися по этой же специальности.

Рецензенты:
кафедра автоматики и процессов управления
Санкт-Петербургского государственного электротехнического университета;
кандидат технических наук доцент *Л. А. Чугунов*

Утверждено
редакционно-издательским советом университета
в качестве учебного пособие

ISBN 5–8088–0044–7

1. КИБЕРНЕТИКА – НАУКА ОБ УПРАВЛЕНИИ

1.1. Создание кибернетики

Кибернетика – это наука об общих законах получения, хранения и передачи информации, т. е. о процессах, которыми характеризуется интеллектуальная деятельность как естественная, так и искусственная. Кибернетика явилась фундаментом, на котором выросла вся современная вычислительная техника и другие науки.

Впервые представление об искусственном интеллекте появилось в научно-фантастической литературе, в частности американский ученый и писатель Айзек Азимов в 1974 году писал: “Я придумал в 1942 году три закона робототехники и написал на их основе несколько десятков рассказов, относясь ко всем этим роботам и искусственным интеллектам как к экзотическим плодам воображения. Я считал, что робот (или вычислительная машина, или искусственный интеллект) необходим человечеству. Затем я обнаружил, что, в конце концов, вычислительные машины смогут достичь интеллектуального уровня, сравнимого с человеческим... и даже намного превзойти его..., но, тем не менее, я должен открыть Вам одно обстоятельство: я никогда не верил в реальность всего этого – в этом отношении мое воображение отказало. Я никогда в действительности не предполагал, что искусственный интеллект возможен, и я совершенно определенно никак не предполагал, что застану время, когда люди будут серьезно работать в этой области” [1].

Научная теория по разработке искусственного интеллекта была провозглашена американским ученым Норбертом Винером в 1948 году. Тогда же он ввел понятие “кибернетики” как науки об общих закономерностях процессов управления и передачи информации в машинах, живых организмах и их объединениях. Вместе с этим словом более 50 лет назад в наш мир ворвался целый поток совершенно новых идей и представлений.

В России первые работы по кибернетике появились в 1956 году, причем сначала это были в основном переводы книг и статей американских ученых, а спустя некоторое время появились и собственные разработки. Одними из первых русских кибернетиков были советские ученые Аксель Иванович Берг, Виктор Михайлович Глушков и Александр Андреевич Ляпунов.

1.2. Предмет и методы исследования кибернетики

В качестве основных разделов кибернетики могут быть выделены:

1. Теория информации, изучающая способы восприятия, хранения, преобразования и передачи информации.

2. Теория программирования, изучающая и разрабатывающая методы переработки информации и использования ее для управления, причем программирование работы любой системы управления в общем случае включает в себя определение и разработку условной математической схемы (алгоритма) нахождения решений и составление программы в коде, воспринимаемом данной системой.

3. Теория систем управления, которая изучает:

структуру и принципы построения систем управления, к которым относятся любые физические объекты, осуществляющие целенаправленную переработку информации, такие как нервная система животного, система автоматического управления движением самолета, электронная программно-управляемая вычислительная машина, система подразделений банка и т. п.;

процесс управления – понятие, которое определяет, каким требованиям должна удовлетворять последовательность действий (или ее описание), осуществляемых системой управления.

Кибернетика изучает абстрактные системы управления, которые представлены в виде математических схем (моделей). Основным свойством этих моделей является то, что они сохраняют информационные свойства соответствующих классов реальных систем, т. е. тех систем, которые моделируются с помощью этих моделей. В рамках кибернетики под влиянием запросов техники ЦВМ и управляющих вычислительных машин возникла специальная математическая дисциплина – теория автоматов, которая изучает специальный класс дискретных (цифровых) систем переработки информации, включающих в себя большое число элементов. Дискретные автоматы, изучаемые в теории автоматов, являются абстрактными моделями реальных систем как технических, так и биологических, которые перерабатывают цифровую информацию дискретными временными тактами.

Для того чтобы в вычислительной машине реализовать некоторый процесс управления, иначе говоря, соответствующий процесс переработки информации, необходимо построить такой алгоритм и осуществить

такую же или примерно такую же переработку информации, как и исходный процесс, а затем оценить качество приближения.

Используемые в кибернетике методы исследования можно разделить на математические и экспериментальные, причем именно к последним относятся логический анализ и синтез систем управления. Поэтому дисциплины “Дискретная математика”, и “Теория автоматов” являются составными частями науки кибернетики.

2. ВВЕДЕНИЕ В ТЕОРИЮ АЛГОРИТМОВ

2.1. Определение алгоритма

Понятия теории алгоритмов всегда традиционно относились к “высокой” науке, считались слабо связанными с практикой и трудными для понимания. Однако жизнь опровергла эти представления, и это доказывается тем, что возникли прикладные ответвления этой теории, а именно, алгоритмические языки программирования и теория формальных исчислений, знание которых стало совершенно необходимым для любого исследователя, имеющего отношение к алгоритмизации процессов управления.

По существу знание этих вопросов дает понимание того, что можно и чего нельзя сделать с помощью вычислительной машины. Сейчас, когда вычислительных машин больше чем людей, умеющих правильно их использовать, такое понимание особенно важно.

Алгоритм может быть определен как эффективная процедура, однозначно приводящая к результату. При разработке алгоритма необходимо формализовать процесс решения задачи, сведя его к применению конечной последовательности достаточно простых правил. Так, мы регулярно пользуемся алгоритмами выполнения основных арифметических операций над многозначными числами, разработанными еще в IX в. древневосточным математиком Аль-Хорезми (термин “алгоритм” произошел от имени этого ученого). До середины XIX в. все алгоритмы были вычислительными и имели дело, в основном, с числами. Поэтому все многообразие вычислений комбинировалось из 10–15 четко определенных операций арифметики, тригонометрии и анализа. Все было очевидно и не требовало специальных исследований.

Появление во второй половине XIX в. математики, имеющей дело с нечисловыми объектами (неэвклидова геометрия, теория групп, теория множеств) показало, что все не так просто и очевидно. Оказалось, что привычные рассуждения, применяемые к этим теориям, основанные на содержательном описании алгоритма, приводят к неразрешимым противоречиям (пример – так называемые противоречия теории множеств). Совершенно особым образом и крайне осторожно следует обращаться с таким понятием, как “бесконечность”. Были созданы так называемые финитные методы исследований, сущность которых заключается в том, что они допускают только конечные комплексы действий над

конечным числом объектов. Все это потребовало уточнения понятия “алгоритм”.

2.2. Предмет теории алгоритмов

В технику термин “алгоритм” пришел вместе с кибернетикой. Если понятие “метода вычислений” не нуждалось в пояснениях, то понятие “процесса управления с помощью алгоритмов” пришлось вырабатывать практически заново. Неясность содержательного описания алгоритма послужила причиной уточнения понятия алгоритма, в результате чего был сделан вывод о существовании так называемой *алгоритмической неразрешимости*, т. е. о несуществовании единого алгоритма решения задач некоторого класса при возможности решения отдельных задач этого класса.

Поэтому предметом теории алгоритмов, решающей задачу уточнения понятия “алгоритм”, является в первую очередь уточнение таких понятий, как “объект”, который подвергается преобразованию и “действие”, совершаемое над этим объектом. Поэтому, разрабатывая алгоритм, необходимо предварительно выяснить:

какие объекты следует считать точно определенными;

какие элементарные действия над ними следует считать точно определенными;

какими свойствами и возможностями обладают комбинации элементарных действий, т. е. что можно и чего нельзя сделать с помощью этих действий;

каким требованиям должна удовлетворять последовательность действий, чтобы считаться конструктивно заданной, т. е. иметь право называться алгоритмом.

В этом осознании огромную роль сыграла практика использования вычислительных машин, сделавшая понятие алгоритма ощутимой реальностью. Поэтому применительно к вычислительной технике можно сформулировать более корректное определение алгоритма: *алгоритм – это точное предписание о выполнении в определенном порядке некоторой системы операций для решения всех задач некоторого заданного типа*.

Тем не менее, нельзя считать алгоритмом любую инструкцию, разбитую на шаги. Поэтому необходимо определить основные требования, предъявляемые к алгоритмам.

1. Алгоритм применяется к исходным данным и выдает результаты. В привычных технических терминах это означает, что алгоритм имеет входы и выходы. Кроме того, в ходе работы алгоритма проявляются промежуточные результаты, которые используются в дальнейшем. Таким образом, каждый алгоритм имеет дело с данными – входными, промежуточными и выходными.

2. Необходимо уточнить понятие “данные”, т. е. указать, каким требованиям должны удовлетворять объекты, чтобы алгоритм мог с ними работать. Для этого объекты должны быть четко определены и отличимы как друг от друга, так и от “необъектов”.

Поскольку точное и полное словесное определение объекта дать достаточно сложно, в теории алгоритмов фиксируют конкретные конечные наборы исходных объектов, называемых элементарными и конкретный набор средств построения других объектов из элементарных объектов. Набор элементарных объектов образует конечный алфавит исходных символов (цифр, букв и др.), из которых строятся другие объекты.

Наиболее распространенный тип алгоритмических данных – слова конечной длины в конечных алфавитах (например, числа), причем число символов в словах (длина слова) – естественная единица измерения обрабатываемой информации. Более сложный случай алгоритмических объектов – формулы. Они также являются словами конечной длины в данном конечном алфавите, однако не каждое слово есть формула.

3. Данные для своего размещения требуют памяти, которая состоит из ячеек и каждая ячейка может содержать один символ алфавита данных. Таким образом, единицы измерения объема данных и памяти согласованы.

4. Алгоритм должен быть дискретным, т. е. должен состоять из множества элементарных шагов или действий, причем множество различных шагов конечно. Примером множества элементарных действий является система команд ЭВМ.

Последовательность шагов алгоритма должна быть детерминирована, т. е. после каждого шага указывается, какой шаг делать дальше либо дается команда остановки.

5. Алгоритм должен быть результативным, т. е. должен остановиться после конечного числа шагов с одновременным указанием того, что считать результатом.

В частности, всякий, кто предъявляет алгоритм решения некоторой задачи, например вычисления некоторой функции $f(x)$, обязан показать, что алгоритм останавливается после конечного числа шагов (сходится) для любого x из области задания функции f.

6. Алгоритм должен удовлетворять требованию массовости, т. е. алгоритм должен быть таким, чтобы его можно было применить для класса задач, различающихся лишь исходными данными.

7. В процессе создания алгоритма следует различать понятия:

описание алгоритма (например, инструкцию или программу);

механизм реализации алгоритма (например, ЭВМ), который включает в себя средства управления ходом вычислений, а именно: средства пуска, средства остановки, средства реализации элементарных шагов, средства выдачи результатов, обеспечения детерминированности;

процесс реализации алгоритма, представляющий собой последовательность шагов, которая будет порождаться при применении алгоритма к конкретным данным.

2.3. Блок-схемы алгоритмов, композиция алгоритмов

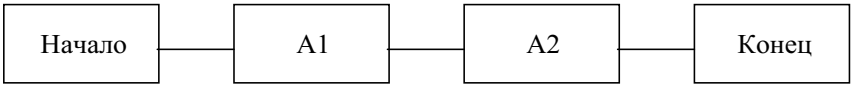
Для представления алгоритма и организации связи между его шагами используются блок-схемы. Блок схема алгоритма изображается в виде графа, в котором вершинам соответствуют шаги алгоритма, а ребрам – переходы между шагами. Вершины могут быть операторными или операторами (выходит одно ребро), условными или предикатам и (выходит два ребра), переключательными (выходит несколько ребер), единственный оператор начала (выходит одно ребро), единственный оператор конца (не выходит ни одного ребра).

Важной особенностью алгоритма является то, что связи, описываемые блок-схемой, не зависят от того, являются ли шаги алгоритма элементарными или представляют собой самостоятельные алгоритмы (блоки).

В программировании известно понятие “разблочивания” сложного алгоритма, когда отдельные его блоки программируются разными лицами. И наоборот, с помощью блок-схемы можно несколько отдельных алгоритмов рассматривать как блоки, которые могут быть связаны в один большой алгоритм.

Например, блок-схема, вычисляющая функцию $f = f_2(f_1(x))$, в которой в качестве аргумента используется другая функция и которая по-

этому называется композицией функций, будет иметь вид, представленный на рис. 2.1.



A1 – алгоритм вычисления функции $f_1(x)$,
$f_2(x)$.

Рис. 2.1

В такой блок-схеме входные данные (входы) алгоритма А2 есть результаты (выходы) алгоритма А1.Такое соединение алгоритмов называется композицией алгоритмов.

На блок-схеме алгоритма хорошо видна разница между описанием алгоритма и процессом его реализации. Описание – это граф, а процесс реализации – это пути в графе, которые определяются логическими условиями. Если в процессе реализации вычислений не появляется условий, ведущих к концу, и процесс зацикливается, это означает отсутствие сходимости алгоритма.

2.4. Алгоритмические модели

В классическом варианте блок-схемы алгоритмов, при всей своей наглядности, отражают связи лишь по управлению (что делать в следующий момент, какому блоку передать управление), а не по информации (не содержат сведений ни о данных, ни о памяти, ни об используемом наборе элементарных шагов). Таким образом, представление алгоритма в виде блок-схемы не удовлетворяет перечисленным выше требованиям.

Поэтому в теории алгоритмов принимается и другой подход к представлению алгоритма, позволяющий построить алгоритмическую модель процесса, когда удается удовлетворить всем требованиям, предъявляемым к алгоритмам. При построении алгоритмической модели производится уточнение детерминизма алгоритма:

выбирается конечный набор исходных объектов, которые называются элементарными;

выбирается конечный набор способов построения из них новых объектов;

фиксируется набор элементарных шагов;

разрабатывается способ создания памяти и выборки информации из памяти.

В результате этих действий создается конкретная алгоритмическая модель.

Можно выделить три основных типа алгоритмических моделей, которые различаются эвристическими соображениями относительно того, что такое алгоритм.

1. В первой модели понятие алгоритма связывается с наиболее традиционными понятиями математики – вычислениями и числовыми функциями. Наиболее широко используемая модель этого типа – *рекурсивные функции*. Рекурсивная функция описывается посредством так называемых индуктивных (или рекуррентных) определений, основанных на переходе от n к n+1. Слово "рекуррентный" означает "возвратный" от латинского слова *recurso* – "возвращаюсь, бегу назад". И если алгоритм носит возвратный характер, то он и называется рекурсивным. Примером может служить алгоритм получения натурального ряда чисел: чтобы определить число 4, нужно сначала определить число 3, а чтобы определить число 3, нужно сначала определить число 2 и т.д. вплоть до 1.

Рекурсивные функции могут быть определены как некоторые новые функции, получаемые из уже имеющихся функций с помощью операции суперпозиции.

Оператором суперпозиции S_m^n называется подстановка в функцию от m переменных m функций от n одних и тех же переменных. Например, если имеем функции $h(x_1, x_2, ...,x_m)$, $g_1(x_1, x_2,...x_n)$, $g_2(x_1, x_2, ...x_n)$, ..., $g_m(x_1, x_2, ..., x_n)$, то S_m^n $(h, g_1, g_2, ..., g_m) = h(g_1(x_1, x_2, ..., x_n), g_2(x_1, x_2, ..., x_n), ..., g_m(x_1, x_2, ...,x_n)) = f(x_1, x_2, ..., x_n)$.

2. Второй тип алгоритмической модели основан на представлении об алгоритме как о некотором детерминированном устройстве, способном выполнять в каждый отдельный момент времени лишь весьма примитивные операции.

Эвристика этих моделей близка к ЭВМ и, следовательно, близка к инженерной интуиции. Основной теоретической моделью этого типа, созданной в 30-х годах, является *машина Тьюринга*.

3. Третий тип алгоритмических моделей – это преобразования слов в произвольных алфавитах, в которых элементарными операциями являются операции подстановки, т. е. замены части слова (подслова) другим словом. Примерами таких моделей являются *конечные автоматы*, созданные по алфавитному отображению.

3. МАШИНА ТЬЮРИНГА

В 1937 году английский математик А.М. Тьюринг, который был последователем теории о том, что машина способна мыслить и с ее помощью можно смоделировать психическую деятельность человека, предложил общую и вместе с тем очень простую концепцию вычислительной машины. В своем труде Тьюринг исходил из того, что предлагаемая им вычислительная машина уподобляется вычислителю, который выполняет операции в точном соответствии с некоторым строгим описанием. Работа машины Тьюринга напоминает действия вычислителя, который, будучи не в состоянии сразу обозреть всю, часто очень громоздкую систему данных и предписаний, производит каждый раз лишь какое-либо "элементарное" действие, причем только над некоторой "воспринимаемой" им частью данных (или промежуточных результатов). На следующем этапе он либо продолжает воспринимать ту же часть данных, либо переходит к другой части, находящейся рядом с ней.

3.1. Структура машины

Машину Тьюринга можно представить состоящей из нескольких основных блоков.

1. *Управляющее устройство* (УУ), которое может быть настроено на выполнение одной из множества возможных операций или, как принято говорить, может находиться в одном из состояний, образующих конечное множество $Q = \{q_1, q_2, ..., q_n\}$.

Среди состояний УУ могут быть выделены начальное состояние q_1 и конечное (пассивное) состояние $q_z \subset \{q_1, ..., q_n\}$. В q_1 машина Тьюринга находится перед началом работы, а, попав в q_z, она останавливается. Очевидно, все состояния из множества Q, отличные от q_z, являются активными.

Работа машины Тьюринга происходит в дискретном времени, когда состояния машины рассматриваются на конечном множестве временных отсчетов, называемых тактами машинного времени и обозначаемых $t_1, t_2, ..., t_p, ...$.

2. *Лента*, разбитая на ячейки, в каждой из которых может быть записан один из символов конечного внешнего алфавита $S = \{s_1, s_2, ..., s_m\}$. Символами этого алфавита кодируются как сведения, подаваемые в машину, так и те сведения, которые в ней вырабатываются. Среди знаков внешнего алфавита имеется пустой символ (пробел) λ. Посылка

(вписывание) этого символа в какую-либо ячейку ленты гасит (стирает) тот символ, который в ней раньше хранился, и оставляет ее пустой. Следует заметить, что λ может принимать значение из алфавита *S*.

Лента бесконечна в обе стороны, однако в начальный момент времени только конечное число ячеек заполнено непустыми символами, остальные ячейки ленты пусты, т. е. содержат символ λ (пробел). И в любой последующий момент времени лишь конечный отрезок ленты заполнен символами, поэтому важна не фактическая бесконечность ленты, а ее неограниченность, т. е. возможность писать на ней сколь угодно длинные, но конечные слова.

3. *Устройство обращения к ленте*, представляющее собой считывающую и записывающую головку, которая в каждый момент времени t_i обозревает какую-либо ячейку ленты и в зависимости от символа в этой ячейке и состояния УУ выполняет следующие действия (рис. 3.1):

а) производит или запись в ячейку нового символа (может быть совпадающего со старым), или стирание символа (запись в ячейку пустого символа λ),

б) сдвигает головку на ячейку вправо или влево, при этом УУ переходит в новое состояние.

Перечисленные действия в комплексе представляют собой шаг (элементарное действие) машины Тьюринга, который определяется парой (q_i, s_j).

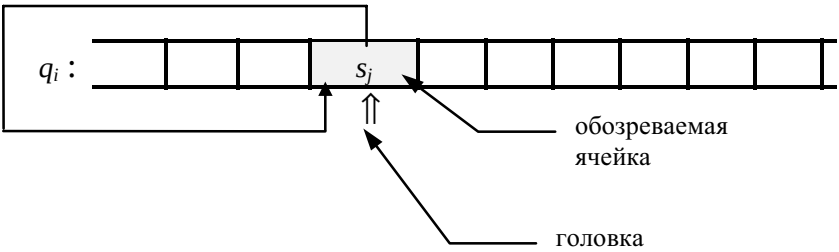


Рис. 3.1

Таким образом, сообразуясь с современными представлениями вычислительной техники, можно считать, что управляющее устройство и устройство обращения к ленте представляют собой Логический блок машины. Лента интерпретируется как внешняя память, в которой записываются исходные данные и окончательные результаты (данные – это

слова, представленные в алфавите S). Внутренняя память машины Тьюринга может быть представлена в виде двух ячеек: ячейки сдвига D, в которую заносится знак (направление) сдвига, и ячейки состояния Q, отображающей текущее состояние машины. Элементарными шагами машины Тьюринга являются: считывание и запись символов, сдвиг головки на ячейку вправо или влево, или, иначе говоря, изменение адреса обозреваемой ячейки ленты на 1, переход УУ в новое состояние.

3.2. Детерминированность машины Тьюринга

Машина Тьюринга обладает свойством детерминированности, т. е. последовательность ее шагов определяется следующим образом.

Для любого внутреннего состояния q_i и символа алфавита s_j однозначно задана *логическая функция*, которая определяет следующее состояние q_i'; символ s_i', который должен быть записан; направление сдвига головки d_k M $\{L, R, E\}$, где L – сдвиг влево, R – сдвиг вправо, E – отсутствие сдвига; причем множество $A = \{L, R, E, q_1, ..., q_m\}$ называется внутренним алфавитом машины Тьюринга.

Таким образом логическая функция машины Тьюринга сопоставляет каждой паре знаков (q_i, s_j) тройку знаков (s_i', q_i', d_k) и может быть записана разными способами.

Первым способом записи логической функции является ф у н к ц и о н а л ь н а я с х е м а машины Тьюринга, представляющей собой таблицу, строкам которой соответствуют входные символы из множества S, столбцам – состояния из множества Q, а на пересечении строк и столбцов записана тройка символов q_i' s_i' d_k. (рис. 3.2).

	q_1	q_2	q_3	…	q_r
s_1					
s_2		$q_8 s_7 L$			
s_3					
…					
s_k					

Рис. 3.2

Логическая функция может быть задана также с помощью системы Тьюринговых команд (Σ_T), которые имеют вид

$$q_i\, s_j \rightarrow q_i'\;\; s_i'\;\; d_k, \qquad (1)$$

где знак "→" читается "влечет за собой" или "приводит к ...". Команда, соответствующая фрагменту функциональной схемы, представленной на рис. 3.2, имеет вид $q_2\, s_2 \rightarrow q_8\, s_7\, L$.

Третьим способом задания логической функции является блок-схема, называемая диаграммой (графом) переходов и изображаемая в виде графа, в котором состояниям машины Тьюринга соответствуют вершины (узлы), а командам вида (1) – ребра, ведущие из q_i в q_i', на которых записано $s_j \rightarrow s_i'\;\; d_k$.

На рис. 3.3 приведен фрагмент диаграммы переходов машины Тьюринга, соответствующий фрагменту функциональной схемы, представленной на рис. 3.2.

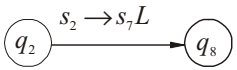


Рис. 3.3

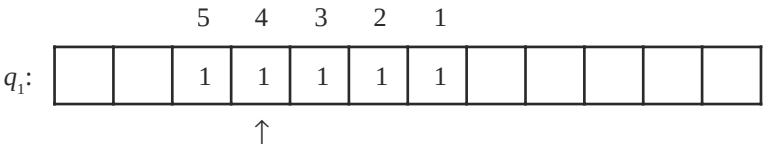
Таким образом, машина Тьюринга представляет собой максимально упрощенный вариант вычислительной машины, имеющей одноадресную структуру, с возможностью изменения адреса обозреваемой ячейки только на 1. Поэтому необходимое для процесса вычислений содержание какой-либо ячейки отыскивается путем постепенной проверки всех ячеек подряд до тех пор, пока не будет обнаружена нужная ячейка.

3.3. Работа машины Тьюринга

Рассмотрим работу машины Тьюринга на следующем примере.

Пусть задана машина Тьюринга с алфавитом $S = \{1, \alpha, \beta, \lambda\}$ и состояниями $Q = \{q_1, q_2, q_3, q_4, q_5\}$.

Перед началом работы машины Тьюринга на ленту заносится начальная информация (например, пять единиц) и фиксируется начальная обозреваемая ячейка (например, 4-я), сдвиг отсутствует. Информация в этой ячейке отражает начальное состояние q_1 машины Тьюринга.



Логическая функция рассматриваемой машины описывается функциональной схемой (рис. 3.4).

	q_1	q_2	q_3	q_4	q_5
λ	$q_4\ \lambda\ R$	$q_3\ \lambda\ L$	$q_1\ \lambda\ R$	$q_5\ \lambda\ L$	$q_5\ \lambda\ E$
1	$q_2\ \alpha\ E$	$q_1\ \beta\ E$	$q_1\ 1\ R$	$q_5\ \lambda\ R$	$q_5\ 1\ E$
α	$q_4\ \alpha\ L$	$q_2\ \alpha\ R$	$q_3\ 1\ L$	$q_4\ \lambda\ R$	$q_5\ \alpha\ E$
β	$q_1\ \beta\ L$	$q_2\ \beta\ R$	$q_3\ \lambda\ L$	$q_4\ 1\ R$	$q_5\ \beta\ E$

Рис. 3.4

Требуется определить, как изменяется информация на ленте при заданных начальных условиях.

Рассмотрим первый способ описания работы машины Тьюринга, в котором в каждом состоянии машины указывается последовательность символов в ячейках ленты.

Такт $1 - t_1$

В ячейки состояния Q и сдвига D заносятся значения начального состояния q_1 и начального сдвига E и обозревается содержание начальной 4-й ячейки (символ "1") при состоянии q_1. В соответствии с функциональной схемой результатом данного шага будет $q_2\ \alpha\ E$, т. е. выполняется Тьюрингова команда

$$q_1\ 1 \rightarrow q_2\ \alpha\ E\ ,$$

где q_2 указывает, на какую операцию перешли; α – что записали в обозреваемую ячейку; E – направление сдвига головки.

Следовательно, сдвиг головки устройства обращения к ленте отсутствует, а символ "1" заменяется в 4-й ячейке на "α".

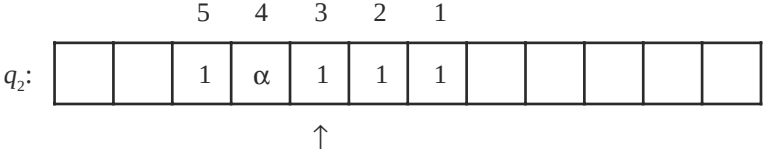
Получим

	5	4	3	2	1					
q_2:			1	α	1	1	1					
				↑								

Такт $2 - t_2$

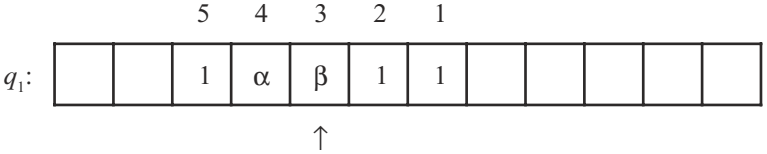
Так как сдвига не было, то вновь обозревается 4-я ячейка, но уже в состоянии q_2.

Результат: $q_2\,\alpha\,R$, т. е. выполняется команда $q_2\,\alpha \to q_2\,\alpha\,R$. Следовательно, головка передвинулась в 3-ю ячейку (вправо), в обозреваемой 4-й ячейке ленты остался символ α, а машина Тьюринга осталась в состоянии q_2. Получим



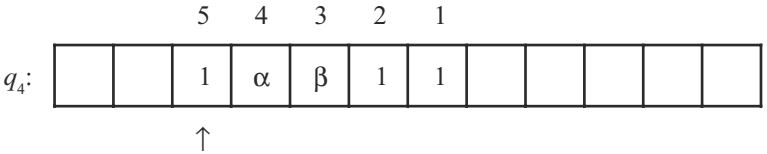
Такт $3 - t_3$

Обозревается "1" из 3-й ячейки. Результат: $q_1\,\beta\,E$, т. е. выполняется команда $q_2\ 1 \to q_1\,\beta\,E$, сдвига нет. Получим



Такт $4 - t_4$

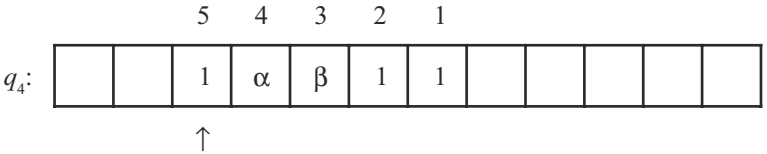
Вновь анализируется 3-я ячейка в состоянии q_1 и обозревается символ β. Результат: $q_1\,\beta\,L$, т. е. выполняется команда $q_1\ \beta \to q_1\,\beta\,L$, и осуществляется сдвиг влево. Получим



Такт $5 - t_5$

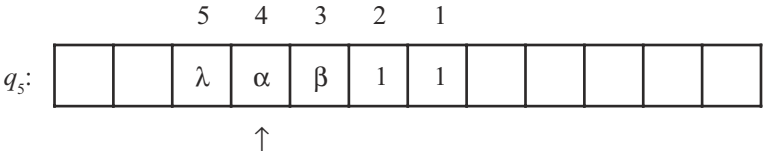
Анализируется 4-я ячейка в состоянии q_1, обозревается символ α.

Результат: $q_4\,\alpha\,L$, т. е. выполняется команда $q_1\ \alpha \to q_4\,\alpha\,L$, осуществляется сдвиг головки влево. Получим



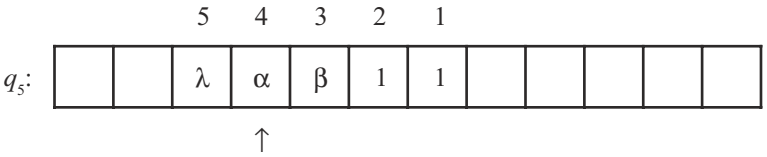
Такт $6 - t_6$

Обозревается 5-я ячейка в состоянии q_4. Там находится символ "1", поэтому результат: $q_5\,\lambda\,R$, т. е. выполняется команда $q_4\ 1 \to q_5\,\lambda\,R$, и осуществляется сдвиг вправо. Получим



Такт 7 – t_7

Обозревается 4-я ячейка в состоянии q_5. Там находится символ “α”, поэтому результат: $q_5\ \alpha\ E$, т. е. выполняется команда $q_5\ \alpha \rightarrow q_5\ \alpha\ E$, состояние и символ не меняются, сдвига нет. Получим



Состояние q_5 является конечным состоянием машины Тьюринга или *стоп–состоянием*, так как после анализа символа α в состоянии q_5, никаких изменений на ленте не происходит и в новое состояние машина не перейдет.

Этот вывод подтверждается анализом последнего столбца функциональной схемы, из которого видно, что при возникновении состояния q_5 произойдет остановка машины, так как любой обозреваемый символ не заменяется другим, а остается. Сдвига также не происходит, и машина снова и снова будет обозревать один и тот же символ. Это и есть стоп-состояние, сигнализирующее о результативном завершении процесса, о его сходимости. В этом случае говорят, что машина Тьюринга применима к информации, поданной на нее до запуска.

В результате работы машины Тьюринга была получена следующая схема изменения информации на ленте.

	5	4	3	2	1
t_0	1	1	1	1	1
t_1	1	α	1	1	1
t_2	1	α	1	1	1
t_3	1	α	β	1	1
t_4	1	α	β	1	1
t_5	1	α	β	1	1
t_6		α	β	1	1
t_7		α	β	1	1

Особенностью описания машины Тьюринга является то, что функциональная схема и структурная схема, ее реализующая, могут быть отождествлены, так как структурная схема всегда одинакова для любой машины. Таким образом, машины Тьюринга отличаются реализуемой логической функцией (Тьюринговой программой), которая существует или в виде функциональной схемы или в виде системы команд (Σ_T). Поэтому термины "Функциональная схема МТ" и "Тьюрингова программа" являются синонимами.

Удобнее пользоваться упрощенной записью Тьюринговых команд и функциональных схем, в которых не записываются выходные символы алфавита и новые состояния, если они не меняются, а также не фиксируется знак E, указывающий на отсутствие сдвига. Это позволяет в таблице опустить столбец, соответствующий стоп-состоянию, само стоп-состояние отметить знаком " ! ", а в системе команд нет необходимости фиксировать последнюю команду.

Например, вместо $q2\alpha \rightarrow q2\alpha R$ можно писать $q2\alpha \rightarrow R$. Тогда упрощенная функциональная схема, представленная на рис. 3.4, будет иметь вид, показанный на рис. 3.5.

	q_1	q_2	q_3	q_4
λ	$q_4 R$	$q_3 L$	$q_1 R$	$! L$
1	$q_2 \alpha$	$q_1 \beta$	$q_1 R$	$!\lambda R$
α	$q_4 L$	R	$1 L$	λR
β	L	R		$1 R$

Рис. 3.5

В таком представлении более нагляден факт того, что оказавшись в состоянии q_1 при обозреваемом знаке β, машина начинает серию сдвигов влево сквозь все рядом стоящие символы β, причем содержание обозреваемых ячеек не меняется (остается β), до тех пор пока в поле зрения головки не попадет какой-либо другой символ. Только при этих условиях машина Тьюринга выйдет из состояния q_1.

3.4. Конфигурация машины Тьюринга

Конфигурацией (полным состоянием или машинным словом) машины Тьюринга называется совокупность ее следующих характеристик:

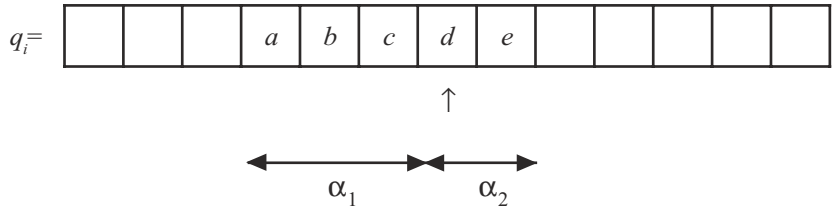
внутреннего состояния;

состояния ленты (т. е. слова, записанного на ленте);

положения головки на ленте.

Конфигурация обозначается тройкой символов $K = \alpha_1 q_i \alpha_2$, где q_i – текущее внутреннее состояние, α_1 – слово слева от головки; α_2 – слово, образованное символом, обозреваемым головкой, и символом справа от него, причем, слева от α_1 и справа от α_2 нет непустых символов (т. е. либо записано λ, либо ничего).

Например, если внутреннее состояние $q_i = abcde$, а головка обозревает символ d, тогда конфигурация машины Тьюринга $K = abcq_i de$, т. е.



Стандартная начальная конфигурация обозначается как $K_1 = q_1\alpha$, где q_1 – начальное состояние, а головка обозревает крайний левый символ. Стандартная конечная конфигурация имеет вид $Kz = q_z\alpha$, где q_z – конечное состояние и вокруг обозреваемой ячейки пустые символы.

Работа машины Тьюринга может быть описана с помощью последовательности конфигураций.

Определение. Если к некоторой конфигурации машины Тьюринга K применима ровно одна команда, приводящая к конфигурации K' , то говорят, что между конфигурациями K и K' существует отношение $K \underset{T}{\rightarrow} K'$, что означает: K переходит в K' по Тьюрингу.

Если же для K_1 и K_n существует последовательность различных конфигураций такая, что $K_1 \underset{T}{\rightarrow} K_2 \underset{T}{\rightarrow} K_3 \underset{T}{\rightarrow} \ldots \underset{T}{\rightarrow} K_n$, то такая последовательность обозначается $K_1 \underset{T}{\Rightarrow} K_n$. Последовательность конфигураций $K_1 \underset{T}{\rightarrow} K_2 \underset{T}{\rightarrow} \ldots \underset{T}{\rightarrow} K_n$ однозначно определяется исходной конфигурацией K_1 и полностью описывает работу машины Тьюринга, начиная с K_1.

В качестве примера рассмотрим систему команд Σ_T, составленную на основе функциональной схемы, приведенной на рис. 3.4 (выписываем только те команды, которые понадобятся для иллюстрации):

$q_1\, 1 \rightarrow q_2\, \alpha\, E$ (1) $\quad q_1\, \mathrm{b} \rightarrow q_1\, \beta\, L$ (4) $\quad q_4\, 1 \rightarrow q_5\, \lambda\, R$ (6)

$q_2\,\alpha \to q_2\,\alpha\,R$ (2) $\quad q_1\,\alpha \to q_4\,\alpha\,L$ (5) $\quad q_5\,\alpha \to q_5\,\alpha\,E$ (7)
$q_2\,1 \to q_1\,\beta\,E$ (3)

При заданной входной информации (11111) и начальной ячейке (4-й) получим следующую последовательность конфигураций машины Тьюринга:

$$1q_1\,1111 \to 1q_2\alpha111 \to 1\alpha\,q_2\,111 \to 1\alpha q_1\beta11 \to$$
$$\text{–К1–} \quad \text{–К2–} \quad \text{–К3–} \quad \text{–К4–}$$
$$\to 1q_1\alpha\beta11 \to q_4 1\alpha\beta11 \to \lambda q_5\,\alpha\beta11 \to \lambda q_5\alpha\beta11$$
$$\text{–К5–} \quad \text{–К6–} \quad \text{–К7–} \quad \text{— ! —}$$
стоп-состояние

Следовательно, $1q_1 1111 \underset{T}{\Rightarrow} \lambda q_5\alpha\beta11$.

3.5. Тьюрингово вычисление

Рассмотрим, как строится машина Тьюринга, реализующая некоторые простые алгоритмы.

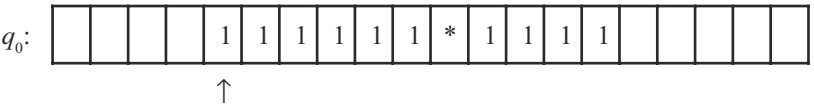
Алгоритм операции "Сложение".

Исходные данные и результаты операции сложения являются натуральными числами.

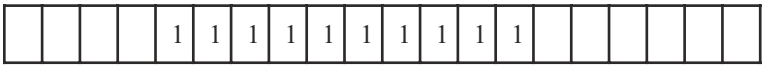
Считаем, что в машине Тьюринга каждое натуральное число задано в виде набора "1" и отделяется от другого числа символом "*". Таким образом, алфавит машины Тьюринга будет $S = \{1,*,\lambda\}$. Состояния заданы множеством $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$.

Рассмотрим пример.

Начальные условия: в начальном состоянии q_0 на ленту машины Тьюринга подается пара чисел 6 и 4 и в поле зрения машины находится левая единица.



Необходимо найти их сумму, т. е. записать подряд 10 единиц, получив



Просто убрать * нельзя, так как на ее месте будет пустая ячейка, а совокупность единиц с пробелом не является заданием числа натураль-

ного ряда. Для реализации сложения необходимо использовать функциональную схему машины Тьюринга, изображенную на рис. 3.6.

	q_0	q_1	q_2
1	$q_2\ \lambda\ R$	L	R
λ	R	$q_0 R$	$q_1 1$
*	!λ	L	R

Рис. 3.6

Или использовать следующую систему команд (выписаны только те, которые понадобятся при создании последовательности конфигураций):

$$q_0\ 1 \to q_2\ \lambda\ R \quad (1)$$
$$q_2\ 1 \to q_2\ 1\ R \quad (2)$$
$$q_2\ * \to q_2\ *\ R \quad (3)$$
$$q_2\ \lambda \to q_1\ 1\text{E} \quad (4)$$
$$q_1\ 1 \to q_1\ 1\ L \quad (5)$$
$$q_1\ \lambda \to q_0\ \lambda\ R \quad (6)$$
$$q_0\ * \to !\ \lambda\ \text{E} \quad (7)$$
$$q_1\ * \to q_1\ *\ L \quad (8)$$

Такт $1 - t_1$

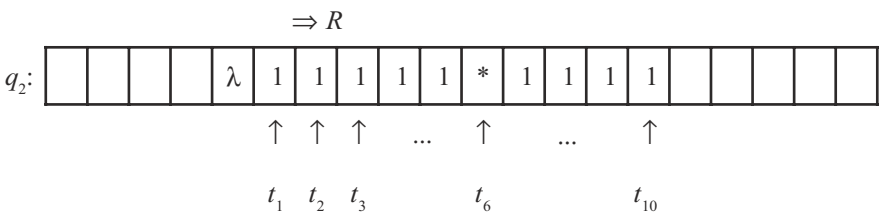
$q_0\ 1 \to q_2\ \lambda\ R$, т. е. вместо первой “1” устанавливается пробел и в состоянии q_2 обозревается вторая “1”, так как сдвиг должен быть вправо.

Такт $2 - t_2$

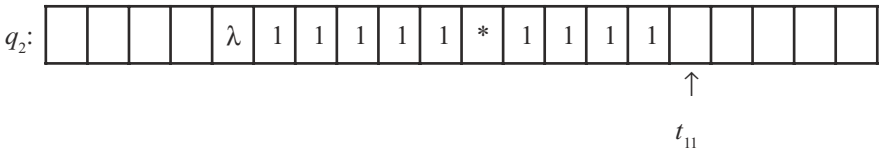
$q_2\ 1 \to R$ (записано в упрощенной форме), следовательно символ не меняется и состояние тоже, поэтому переходя направо от “1” к “1”, будем все время оставлять их в ячейках, оставаясь в состоянии q_2. Попав на знак “*”, его тоже оставим, так как $q_2\ * \to R$.

Этот сдвиг вправо будет продолжаться в течение 9 тактов до тех пор, пока в такте 11 не попадем в пустую ячейку.

Такты 1–10 ($t_1 - t_{10}$)

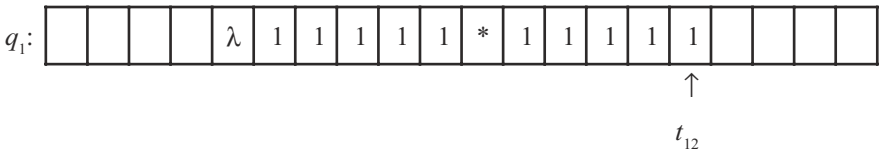


Такт 11 – t_{11}



Такт 12 – t_{12}

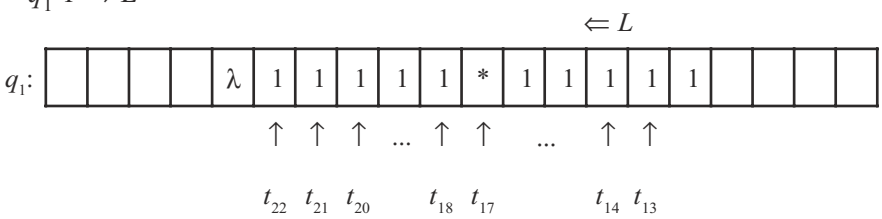
$q_2\ \lambda \rightarrow q_1\ 1$, т. е. в пустую ячейку вписываем “1” и переходим в состояние q_1



Теперь вновь попали в начальную ситуацию, но уже не в состоянии q_0, а в состоянии q_1.

Такт 13 – t_{13}

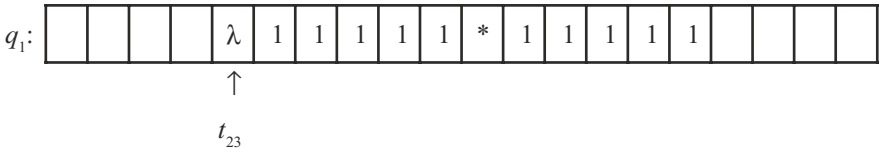
$q_1\ 1 \rightarrow L$



Далее в течение тактов $t_{14} - t_{23}$ будет происходить обратный сдвиг (влево) через все 1 и * до тех пор, пока головка не окажется в левой пустой ячейке.

Такт 23 – t_{23}

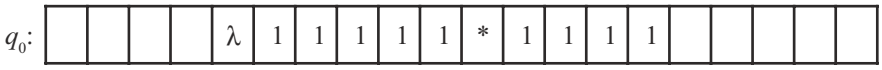
$q_1\ 1 \rightarrow L$, произвели последний сдвиг влево и оказались в пустой ячейке



Такт 24 – t_{24}

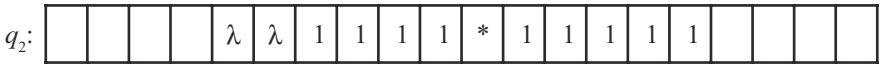
$q_1\ \lambda \rightarrow q_0 R$, переходим в состояние q_0 и сдвигаемся вправо к первой “1”

$\Rightarrow R$



t_{24} (аналог t_0)

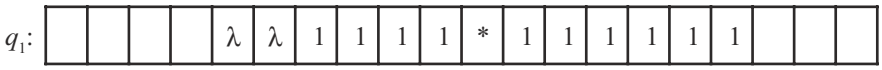
Такт 25 – t_{25}



t_{25} t_{26} (аналог t_2)

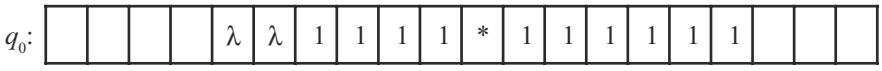
Т.е. цикл завершен, и с этого момента (t_{26}) начинается новый цикл аналогичных операций. По окончании этого цикла еще одна “1” будет перенесена слева направо в конец последовательности.

Такт 47 – t_{47}



t_{47} t_{37} (аналог t_{12})

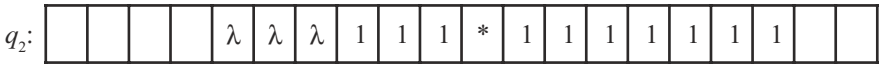
Такт 48 – t_{48}



t_{48} (аналог t_0) t_{35} (аналог t_{10})

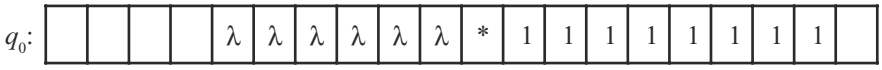
Такт 49 – t_{49}

$q_0 1 \rightarrow q_2 \lambda R$



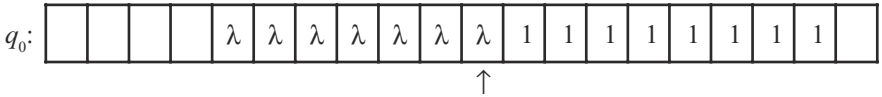
t_{49} t_{50} (аналог t_2) t_{60}

Предпоследний такт (t_{143})



Последний такт (t_{144})

q_0 * → ! λ, переходим в конечное состояние и помещаем пробел, вместо *



В результате получена нужная сумма.

Длительность выполнения операции “Сложение” зависит от величины слагаемых, так как временные затраты на один цикл сложения составляют $T_{цикл} = 2\ (m+n+1)$ тактов, где m – число символов перед “*”, n – число символов после “*”.

Полностью на выполнение сложения понадобится

$T_{слож} = 2\ (m+n+1)\ m + 2m = 2m\ (m+n+2)$ тактов

число циклов — время на подготовку к циклу и на завершение цикла.

Поэтому в нашем примере $T_{цикл} = 2 \cdot 11 = 22$ (такта) и $T_{слож} = 12 \cdot 12 = 144$ (такта).

Временные соотношения иллюстрируются табл. 3.1

Таблица 3.1

Подготовка	Цикл		Завершение (+11)
	Начало	Установка "1" (+10)	
t_0	$t_1 - t_2$	t_{12}	t_{23}
t_{24}	$t_{25} - t_{26}$	t_{36}	t_{47}
t_{48}	$t_{49} - t_{50}$	t_{60}	t_{71}
t_{72}	$t_{73} - t_{74}$	t_{84}	t_{95}
t_{96}	$t_{97} - t_{98}$	t_{108}	t_{119}
t_{120}	$t_{121} - t_{122}$	t_{132}	t_{143}–t_{144} (стоп-состояние)

I цикл – $t_2 - t_{23}$
II цикл – $t_{26} - t_{47}$
III цикл – $t_{50} - t_{71}$
IV цикл – $t_{74} - t_{95}$
V цикл – $t_{98} - t_{119}$
VI цикл – $t_{122} - t_{143}$, t_{144} – конец

3.6. Тезис Тьюринга

При реализации алгоритмов на машине Тьюринга может возникнуть вопрос для всех ли конструктивных процедур, т. е. для тех процедур, для которых можно построить алгоритм, можно разработать реализующие их машины Тьюринга. В тезисе Тьюринга, который является основной гипотезой теории алгоритмов, содержится утвердительный ответ на этот вопрос.

Тезис формулируется следующим образом: всякий алгоритм может быть задан посредством Тьюринговой функциональной схемы и реализован в соответствующей машине Тьюринга. Доказать тезис Тьюринга нельзя, так как само понятие алгоритма является неточным. В формулировке тезиса Тьюринга идет речь, с одной стороны, о всяком алгоритме, т. е. об общем понятии алгоритма, которое не является точным математическим понятием; а с другой стороны, в этой же формулировке речь идет о точном математическом понятии – о Тьюринговой функциональной схеме.

Значение гипотезы как раз и заключается в том, что она уточняет общее, но расплывчатое понятие “всякого алгоритма “ через более специальное, но уже совершенно точное математическое понятие “Тьюринговой функциональной схемы” (и ее реализации в машине Тьюринга), т. е. общее расплывчатое понятие алгоритма отождествляется с точным понятием функциональной схемы машины Тьюринга.

Уверенность в справедливости тезиса Тьюринга основана главным образом на опыте. Все известные алгоритмы, которые были придуманы в течение многих веков истории математики, могут быть заданы посредством Тьюринговых функциональных схем. По своему характеру тезис Тьюринга напоминает об адекватности математических моделей физическим явлениям и процессам.

Исходя из тезиса Тьюринга, невозможность построения машины Тьюринга означает отсутствие алгоритма решения данной проблемы. Расшифруем это положение. В числе общих требований, предъявляемых к алгоритмам, упоминается требование результативности. Наиболее радикальной формулировкой здесь было бы требование, чтобы по любому алгоритму A и данным α можно было бы определить, приведет ли работа A при исходных данных α к результату или нет. Иначе говоря нужно построить алгоритм B, такой что $B(A,\alpha)$ = *истина*, если $A(\alpha)$ дает результат, и $B(A,\alpha)$ = *ложь*, если $A(\alpha)$ не дает результата.

В силу тезиса Тьюринга задачу о том, приведет ли реализация алгоритма A при исходных данных α к результату или нет, можно сформулировать как задачу построения машины Тьюринга следующим образом [1].

Построить машину Тьюринга T_0 такую (с такой системой команд), что для любой машины Тьюринга T с системой команд $\sum_т$ и любых исходных данных a для машины T $T_0(\sum_т, \alpha) = истина$, если $T(\alpha)$ – останавливается и $T_0(\sum_т, \alpha) = ложь$, если $T(\alpha)$ не останавливается (происходит зацикливание).

Эта задача называется проблемой остановки, и если остановка невозможна, т. е. решаемая проблема является алгоритмически неразрешимой, никакое кодирование системы команд $\sum_т$ и a в алфавите машины T_0 не приводит к успеху.

Но, тем не менее, если удастся разбить проблему на отдельные частные случаи, ее оказывается возможным решить, пользуясь разными средствами. Поэтому неразрешимость общей проблемы остановки вовсе не снимает необходимость доказывать сходимость предлагаемых алгоритмов, а лишь доказывает, что поток таких доказательств нельзя полностью автоматизировать.

Машина Тьюринга является прообразом любой вычислительной машины, так как каждая физически осуществимая вычислительная машина может быть рассмотрена лишь как некоторая приближенная модель машины Тьюринга. Именно в реальных машинах объем внешней памяти ограничен, в то время как в машине Тьюринга фигурирует бесконечная лента. Разумеется, техническое осуществление неограниченной памяти невозможно, но очевидна и современная, и прошлая тенденции к постоянному увеличению объема памяти и скорости вычислений по сравнению с уже достигнутым уровнем.

4. ВВЕДЕНИЕ В ТЕОРИЮ АВТОМАТОВ

Теория автоматов – это теория, на которой основаны экспериментальные методы исследования в кибернетике. При подходе к теории автоматов, как к части теории алгоритмов, центральной проблемой является изучение возможностей автоматов в терминах множеств слов, с которыми работают автоматы.

Можно выделить два основных аспекта работы автоматов.

1. Автоматы-распознаватели, которые распознают входные слова, т. е. отвечают на вопрос, принадлежит ли поданное на вход слово данному множеству.

2. Автоматы-преобразователи, которые преобразуют входные слова в выходные, т. е. реализуют автоматные отображения.

Одной из задач теории автоматов является задача описания автомата и его реализации, т. е. представления автомата как структуры, состоящей из объектов фиксированной сложности (элементов). В этом отношении теория автоматов оказалась наиболее развитой ветвью теории алгоритмов.

Общая теория автоматов подразделяется на абстрактную теорию и структурную теорию автоматов. Абстрактная теория автоматов занимает промежуточное положение между алгеброй и логикой. С точки зрения приложений значение абстрактной теории автоматов отнюдь не сводится к удовлетворению запросов одной лишь вычислительной техники. Современная теория автоматов представляет собой математический аппарат для решения широкого класса комбинаторных проблем.

В частности, с помощью теории автоматов могут быть решены многие лингвистические задачи.

Например, пусть дано некоторое число фраз на незнакомом языке и их перевод на другой незнакомый язык. Требуется осуществить перевод некоторого числа новых фраз с первого языка на второй при условии, что в них используются лишь те слова и грамматические правила, которые встречаются в уже переведенных фразах. Решение этой задачи может состоять из следующих этапов.

1. В исходном множестве фраз первого языка выделяются различные входящие в них элементы языка (корни слов, окончания, суффиксы, префиксы и т.п.). Эти элементы объединяются в алфавит, называемый входным.

2. Аналогичным образом из исходных фраз второго языка формируется выходной алфавит.

Можно осуществлять и более мелкое дробление, используя в качестве входного и выходного алфавитов обычные алфавиты первого и второго языков, но тогда решение задачи становится более громоздким.

3. Перевод теперь представляется как установление соответствия между словами во входном и выходном алфавитах, что и делается на третьем этапе.

4. Используя алгоритм синтеза, по установленному соответствию строится осуществляющий его автомат *A*.

5. Используя алгоритм минимизации, производится оптимизация автомата А по числу состояний.

Полученный автомат будет осуществлять преобразование входных слов в выходные на более широкой области, включающей (при соблюдении оговоренных выше условий и соответствующих ограничений грамматики) все новые фразы, подлежащие переводу. Примером такого перевода было решение задачи перевода с венгерского языка на баскский, выполненное в Институте кибернетики в г. Киеве под руководством В.М. Глушкова. При ее решении по 10 парам исходных фраз был синтезирован автомат с 75 состояниями, который путем минимизации был приведен к 46 состояниям.

Структурная теория автоматов позволяет реализовать абстрактный автомат на элементах, принадлежащих к заранее заданному классу.

4.1. Алфавитные операторы и автоматы

Под абстрактным алфавитом понимают любую конечную совокупность объектов, называемых буквами данного алфавита. Слова в этом алфавите определяют как любые конечные упорядоченные последовательности букв. Число букв в слове называют длиной слова, причем, наряду со словами положительной длины (состоящими не менее, чем из одной буквы), рассматривают также пустое слово, не содержащее ни одной буквы. Слова единичной длины отождествляются с буквами алфавита.

Алфавитным оператором, или алфавитным отображением, называют всякое соответствие (функцию), сопоставляющее словам в том или ином алфавите слова в том же самом или в некотором другом фиксированном алфавите. Первый алфавит называют при этом входным, а

второй – выходным алфавитом данного оператора. Алфавитный оператор, сопоставляющий каждому входному слову (слову во входном алфавите оператора) не более одного выходного слова (слова в выходном алфавите оператора), называют однозначным. Если алфавитный оператор не сопоставляет данному входному слову никакого выходного слова (в том числе и пустого), то говорят, что он не определен на этом слове. Совокупность всех слов, на которых алфавитный оператор определен, называется его областью определения. Алфавитный оператор называют частичным, если его область определения не совпадает с совокупностью всех входных слов.

Всякий дискретный преобразователь информации, выдающий некоторый выходной сигнал (букву выходного алфавита) в ответ на каждый входной сигнал (букву входного алфавита), реализует некоторый алфавитный оператор. Такие преобразователи воспринимают и выдают сигналы лишь в моменты времени, разделенные промежутками ненулевой длительности, и называются *автоматами*.

В общем случае выходной сигнал автомата в каждый момент времени зависит от значений входного сигнала не только в данный, но и в предыдущие моменты времени. Эта зависимость проявляется в изменении у автомата его внутренних состояний таким образом, что сигнал на его выходе в каждый момент времени определяется как входным сигналом в тот же момент времени, так и состоянием, в котором оказался автомат к данному моменту времени в результате воздействия на его вход сигналов, поступивших в предыдущие моменты времени.

Если выходной сигнал автомата в каждый момент времени целиком определяется входным сигналом в тот же момент времени, то автомат имеет единственное внутреннее состояние и его называют комбинационной схемой.

Примером автомата с неединственным состоянием может служить устройство управления лифтом. Входными сигналами для него являются номера требуемых этажей $Z_1 < Z_2 < ... < Z_F < \infty$. Реакция кабины лифта на каждый входной сигнал должна быть различной в зависимости от того, на какой этаж кабина приведена входными сигналами, поступившими к данному моменту времени. Если кабина находится ниже требуемого этажа она должна двигаться вверх до требуемого этажа, если выше – вниз до требуемого этажа, если на требуемом этаже – должна остаться на месте. Таким образом, выходные сигналы w_j уст-

ройства управления лифтом можно считать равными разностям $z_i - z_k$, где z_k – номер этажа, на котором находилась кабина лифта к моменту поступления очередного входного сигнала (номера требуемого этажа) z_i, при этом сигнал $w_j = 0$ не включается в выходной алфавит (пустой символ). Появление каждого выходного сигнала зависит от входного сигнала, действующего в данный момент времени, и от номера этажа, на котором к этому моменту времени оказалась кабина лифта. По этой причине можно считать, что автомат управления лифтом имеет число состояний, равное числу этажей здания. Поскольку процесс движения кабины между этажами не представляет интереса, считается, что состояние автомата изменяется мгновенно в момент достижения кабиной очередного этажа и с этим же моментом времени отождествляется момент поступления входного сигнала, вызвавшего движение кабины к этому этажу.

4.2. Абстрактные автоматы

Автоматы, рассматриваемые безотносительно к их внутренней структуре, принято называть *абстрактными автоматами*.

Абстрактный автомат работает в дискретном автоматном времени, последовательные моменты которого отождествляются с натуральными числами $t = 1, 2, \ldots$. В примере с лифтом моменты автоматного времени определяются кнопкой “пуск” либо моментами нажатия кнопок с номерами требуемых этажей.

Для задания абстрактного автомата нужно задать входной алфавит $Z = \{z_1, z_2, \ldots, z_F\}$, выходной алфавит $W = \{w_1, w_2, \ldots, w_G\}$ и множество $A = \{a_1, a_2, \ldots, a_M\}$ его внутренних состояний, называемое просто множеством состоянии автомата. В каждый момент времени $t = 0, 1, 2, \ldots$ абстрактный автомат A находится в некотором состоянии $a_i = a(t)$ из A. Состояние $a_0 = a(0)$ в начальный момент времени $t = 0$ называется начальным состоянием автомата A. Условно абстрактный автомат изображается в виде устройства (рис. 4.1) с одним входом и одним выходом.

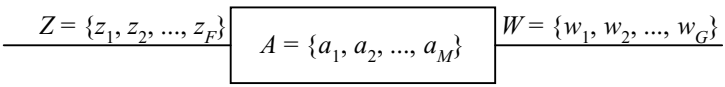


Рис. 4.1

В каждый момент t автоматного времени, начиная с $t = 1$, на вход автомата поступает в качестве входного сигнала одна из букв $z_i = z(t)$ алфавита Z. Конечные упорядоченные последовательности входных букв $z(1)z(2)...z(k)$ автомата будут входными словами. На вход автомата может подаваться любое входное слово из некоторого фиксированного множества допустимых входных слов. Каждое допустимое слово $p = z(1)z(2)...z(k)$, поданное на вход данного автомата A, вызывает появление на выходе автомата выходного слова $q = w(1)w(2)...w(k)$, представляющего собой некоторую упорядоченную конечную последовательность выходных сигналов автомата A (букв алфавита W), имеющего ту же самую длину, что и соответствующее ему входное слово p. Получаемое преобразование φ_A допустимых входных слов p в соответствующие им выходные слова q является алфавитным оператором, индуцированным автоматом A, или просто оператором автомата A.

В примере с лифтом оператор управляющего автомата можно задать явно. Из найденных выше соотношений между входными и выходными сигналами для этого автомата следует, что

$$w(t) = z(t) - z(t-1),$$

если принять $z(0) = z_1$. Это и есть искомое преобразование входных слов автомата в выходные. Поскольку длина входных слов для этого преобразования не ограничена, множество допустимых входных слов для него бесконечно и даже несчетно при всяком $F \geq 2$.

Оператор φ_A однозначно определяется заданием функции переходов δ и функции выходов λ рассматриваемого автомата. Функция переходов определяет состояние $a(t)$ автомата в некоторый момент t автоматного времени по входному сигналу $z(t)$ в тот же самый момент и состояний $a(t-1)$ в предыдущий момент автоматного времени

$$a(t) = \delta(a(t-1), z(t)). \qquad (4.1)$$

Функция выходов определяет зависимость выходного сигнала от тех же самых переменных

$$w(t) = \lambda(a(t-1), z(t)). \qquad (4.2)$$

Задавая любое выходное слово $p = z(1)z(2)...z(k)$ и начальное состояние $a(0)$ автомата, с помощью соотношений (4.1) и (4.2) можно последовательно определить все буквы соответствующего выходного слова

$$q = \varphi_A(p) = w(1)w(2)...w(k).$$

Таким образом, соотношения (4.1) и (4.2) действительно определяют оператор φ_A автомата A.

Функции переходов и выходов представляются обычно абстрактными частичными функциями $\delta(a, z)$ и $\lambda(a, z)$, задающими однозначное отображение некоторого множества пар $(a, z)(a \in A, z \in Z)$ в множества A и W соответственно. Допустимыми входными словами для автомата A называют те и только те входные слова p, на которых с помощью функций δ и λ указанным выше способом можно определить соответствующие им выходные слова $q = \varphi_A(p)$.

Автомат называют конечным, если конечны все три определяющие его множества Z, W, A.

Для примера с лифтом функции переходов и выходов определяются соответственно следующими равенствами:

$$a(t) = z(t),$$

$$w(t) = z(t)–a(t–1).$$

Автомат управления лифтом конечен, хотя область определения его оператора состоит из бесконечного множества допустимых слов.

Поскольку дальше будут рассматриваться только конечные автоматы, слово “конечный” будет опускаться.

Автомат называют вполне определенным, если его функции переходов и выходов заданы на всех парах (a, z), и частичным – в противном случае.

Два абстрактных автомата считаются одинаковыми, если они отличаются друг от друга лишь обозначениями входных и выходных сигналов, состояний.

4.3. Способы задания абстрактных автоматов

Если заданы входной и выходной алфавиты автомата, а также множество его состояний, среди которых фиксировано начальное состояние a_0, для задания абстрактного автомата остается задать функцию переходов δ функцию выходов A .

Автоматы, функции переходов и выходов которых удовлетворяют условиям (4.1), (4.2), называют автоматами Мили. Автоматы, у которых функции переходов и выходов удовлетворяют условиям

$$a(t) = \delta(a(t–1), z(t)), \qquad (4.3)$$

$$w(t) = \lambda(a(t)), \qquad (4.4)$$

называют автоматами Мура [2] .

Различие между автоматами Мили и Мура состоит в том, что выходной сигнал в автомате Мили зависит как от состояния в предыдущий момент времени, так и от входного сигнала в рассматриваемый момент времени, а в автомате Мура – только от состояния в рассматриваемый момент времени. Если подставить правую часть (4.3) в (4.4), то получится равенство типа (4.2). Таким образом, автомат Мура всегда можно свести к автомату Мили с тем же самым числом состояний и теми же самыми входным и выходным алфавитами. Для конечного автомата функции δ и λ определены на конечном множестве значений аргументов и принимают значения из конечных множеств. По этой причине для конечных автоматов функции δ и λ можно задать в виде таблиц.

Таблица 4.1

$a(t-1)$	z_0	z_1	z_2
a_0	a_2	a_3	a_1
a_1	a_3	a_0	a_2
a_2	a_0	a_1	a_1
a_3	a_2	a_2	a_3

Таблица 4.2

$a(t-1)$	z_0	z_1	z_2
a_0	w_3	w_2	w_1
a_1	w_0	w_3	w_0
a_2	w_1	w_0	w_2
a_3	w_2	w_1	w_3

В табл. 4.1 и 4.2 приведены примеры соответственно функций переходов и выходов автомата Мили.

Поскольку области определения функций δ и λ совпадают, таблицы переходов и выходов могут быть совмещены в одну таблицу переходов-выходов (табл. 4.3), в которой на пересечении i-й строки и j-го столбца ($i \in \{0, 1, 2, ..., n\}$, $j \in \{0, 1, 2, ..., r\}$) записывается в числителе новое состояние автомата, а в знаменателе – выходной сигнал, выработанный при переходе в это состояние.

В табл. 4.4, 4.5 приведены примеры функций переходов δ и выходов λ автомата Мура. Функция выходов λ автомата Мура зависит только от одного параметра $a(t)$, поэтому при совмещении таблиц переходов и выходов достаточно столбцу состояний автомата поставить в соответствие столбец выходных сигналов (табл. 4.6).

Таблица 4.3

$a(t-1)$	z_0	z_1	z_2
a_0	a_2 / w_3	a_3 / w_2	a_1 / w_1
a_1	a_3 / w_0	a_0 / w_3	a_2 / w_0
a_2	a_0 / w_1	a_1 / w_0	a_1 / w_2
a_3	a_2 / w_2	a_2 / w_1	a_3 / w_3

Таблица 4.4

$a(t-1)$	z_0	z_1	z_2
a_0	a_3	a_2	a_0
a_1	a_0	a_3	a_2
a_2	a_2	a_0	a_1
a_3	a_0	a_2	a_2

Таблица 4.5

$a(t)$	$w(t)$
a_0	w_2
a_1	w_0
a_2	w_3
a_3	w_1

Таблица 4.6

$w(t-1)$	$a(t-1)$	z_0	z_1	z_2
w_2	a_0	a_3	a_2	a_0
w_0	a_1	a_0	a_3	a_2
w_3	a_2	a_2	a_0	a_1
w_1	a_3	a_0	a_2	a_2

Возможен графический способ задания функций переходов и выходов автомата. На рис. 4.2 и 4.3 приведены графы переходов автоматов Мили и Мура, таблицы переходов-выходов которых только что рассматривались.

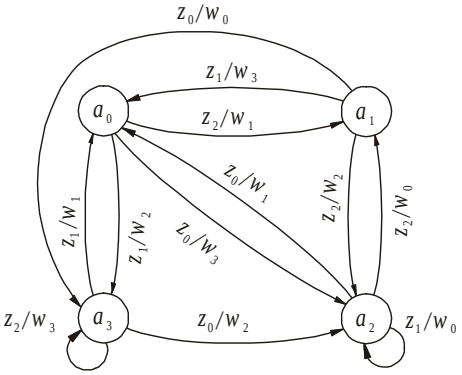


Рис. 4.2

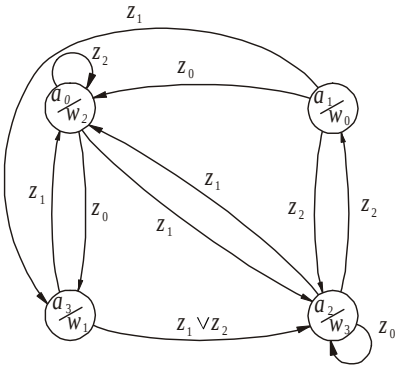


Рис. 4.3

Для автомата Мили вершины графа отмечаются состояниями. Если из состояния a_i имеется переход в состояние a_j, то из вершины a_i в

вершину a_j проводится дуга, около которой в числителе указывается входной сигнал, вызывающий этот переход, а в знаменателе – возникающий при этом выходной сигнал.

Для автомата Мура вершины графа отмечаются состояниями и связанными с ними входными сигналами. Дуги графа отмечаются входными сигналами, под действием которых возникают рассматриваемые переходы.

В приведенных примерах функции δ и λ определены при всех значениях переменных $a(t–1)$ и $z(t)$, поэтому им соответствуют вполне определенные автоматы. Для частичных автоматов функции δ и λ определены не на всех парах $a(t–1)$ и $z(t)$. В этом случае на месте неопределенных переходов или выходов в таблицах ставятся прочерки. В графах же отсутствуют дуги, соответствующие неопределенным переходам, а вместо неопределенных выходных сигналов ставятся прочерки.

Если заданы функции переходов и выходов автомата, по ним можно построить оператор автомата. Однако не по всякому заданному алфавитному оператору можно построить функции переходов и выходов некоторого автомата.

4.4. Автоматные операторы

Пусть оператор j_A автомата A удовлетворяет следующим четырем условиям.

1. φ_A осуществляет однозначное отображение (вообще говоря, частичное) множества входных слов в множество выходных слов.

2. Область определения оператора φ_A удовлетворяет условиям полноты, т. е. вместе с любым содержащимся в ней словом содержит и все начальные отрезки этого слова (так, если области определения оператора φ_A принадлежит слово $z_1z_2z_3z_4$, то ей принадлежат и слова z_1, z_1z_2, $z_1z_2z_3$). Пустое слово всегда входит в область определения оператора φ_A.

3. φ_A сохраняет длину слова: любое входное слово p, на котором оператор φ_A определен, имеет ту же длину, что и его образ $\varphi_A(p)$. В частности, пустое слово переводится оператором φ_A в пустое слово.

4. φ_A переводит любой начальный отрезок слова p, на котором он определен, в имеющий ту же длину начальный отрезок слова $\varphi_A(p)$.

Эти четыре условия называют условиями автоматности оператора, а удовлетворяющий им оператор – *автоматным оператором*.

Если алфавитный оператор φ удовлетворяет условиям автоматности, можно построить автоматы Мили и Мура (вообще говоря, бесконечные), индуцирующие оператор φ. В случае, когда область определения оператора φ конечна (конечно множество слов, на которых определен оператор φ), эти автоматы также могут быть выбраны конечными [2] .

Не всякий алфавитный оператор удовлетворяет условиям автоматности. Существует, однако, прием, позволяющий превратить любой алфавитный оператор в автоматный оператор.

Пусть φ – произвольный алфавитный оператор с конечными входным Z и выходным W алфавитами, P – область определения этого оператора. Будем применять к оператору φ две операции. Первая из них называется операцией выравнивания длин слов. Она заключается в том, что во входной и выходной алфавиты добавляется по одной пустой букве a и β соответственно, а затем к любому слову p из P приписываются справа m_p экземпляров буквы a, а к его образу $q = \varphi(p)$ приписываются слева n_q экземпляров буквы β так, чтобы длины полученных в результате приписывания этих букв слов p_1 и q_1 совпали. Если, в частности, числа m_p выбираются равными длине слова q, а числа n_q равными длине слова p, то операция выравнивания длины слов называется стандартной.

После выполнения операции выравнивания длин слов строится новый оператор φ_1, отображающий некоторое множество P_1 слов в алфавите $Z \cup \{a\}$ в множество слов в алфавите $W \cup \{\beta\}$. Он действует по правилу $q_1 = \varphi_1(p_1)$, где слова p_1 и q_1 получены выравниванием длин слов p и q, причем p пробегает все P. Оператор φ_1 называется при этом выравненным оператором.

Вторая операция применяется только к выравненным алфавитным операторам, т. е. к таким операторам, у которых длины входных и соответствующих им выходных слов равны между собой. Ее называют операцией пополнения. Она заключается в распространении отображения φ на начальные отрезки слов. Это осуществляется следующим образом. Если S – произвольный начальный отрезок любого слова p из области определения оператора φ, то $\varphi(S)$ полагается равным начальному отрезку слова $\varphi(p)$, имеющим ту же, что и отрезок S, длину.

В результате применения операции пополнения к произвольному выравненному алфавитному оператору φ_1 возникает оператор φ_1', называемый пополнением оператора φ_1. Если пополнение φ_1' оператора φ_1 яв-

ляется однозначным, то оно удовлетворяет, очевидно, всем четырем условиям автоматности. К сожалению, однако, в большинстве случаев пополнение оператора φ_1 неоднозначно. Вместе с тем справедливо следующее утверждение [2]: в случае, когда оператор φ_1 получен из некоторого однозначного алфавитного оператора в результате стандартной операции выравнивания длин слов, пополнение φ_1' этого оператора однозначно и является автоматным оператором.

В ряде случаев при превращении заданного оператора в автоматный оператор можно применять не стандартную операцию выравнивания, а какой-нибудь более экономный (с точки зрения числа дописываемых букв) вариант операции выравнивания. В частности, если сам исходный оператор был автоматным, можно считать, что применяется нулевая операция выравнивания, при которой никакого дописывания пустых букв вообще не происходит.

Обычно на практике поступают следующим образом. Сначала операцию выравнивания проводят наиболее экономным образом, и, применяя затем операцию пополнения, проверяют (по признаку однозначности пополнения), получится ли в результате автоматный оператор. Если нет, то производят новое дописывание пустых букв, новую проверку пополнения и т.д. В результате продолжения подобного процесса обязательно будет получен автоматный оператор. Этот метод приведения алфавитного оператора к автоматному оператору называют методом последовательного приведения.

Для построенного таким методом оператора можно построить функции переходов и выходов.

5. АБСТРАКТНЫЙ СИНТЕЗ АВТОМАТОВ

5.1. Построение функций переходов и выходов по алфавитному оператору

Если область определения алфавитного оператора φ конечна, его чаще всего задают с помощью таблицы соответствия. В левой половине этой таблицы выписывают в том или ином порядке входные слова, на которых задан оператор φ, а в правой части таблицы – соответствующие им выходные слова.

Рассмотрим процесс построения функций переходов и выходов по алфавитному оператору на конкретном примере. Пусть входной и выходной алфавиты состоят только из двух букв $\{z_0, z_1\}$ и $\{w_0, w_1\}$ соответственно, а оператор (табл. 5.1) задан таблицей соответствия.

Таблица 5.1

$z(1)$	$z(2)$	$z(3)$	$w(1)$	$w(2)$	$w(3)$
z_0	z_0	z_0	w_1	w_0	w_0
z_0	z_0	z_1	w_0	w_0	w_0
z_0	z_1	z_0	w_1	w_0	w_0
z_0	z_1	z_1	w_0	w_1	w_1
z_1	z_0	z_0	w_0	w_0	w_1
z_1	z_0	z_1	w_0	w_1	w_1
z_1	z_1	z_0	w_0	w_1	w_0
z_1	z_1	z_1	w_1	w_0	w_1

Данный оператор уже выравнен, так как длина каждого из входных слов равна длине соответствующего выходного слова. Каждому входному слову здесь сопоставляется не более одного выходного слова, поэтому оператор φ однозначен. Однако он не удовлетворяет второму условию автоматности. Действительно, если нумеровать слова в табл. 5.1 сверху вниз, то $z(1) = z_0$ для первого и третьего слов преобразуется в $w(1) = w_1$, а для второго и четвертого слов $w_1(1) = w_0$. По этой причине оператор φ неоднозначен для начальных отрезков слов. Устранить это можно было бы, приписав справа к каждому входному слову $m_p = 3$ экземпляра буквы α и слева к каждому выходному слову $m_q = 3$ экземпляра буквы β, т. е. применив стандартную процедуру выравнивания длин слов. Воспользуемся более экономной процедурой.

Поскольку в табл. 5.1 при $z(1) = z_0$ для второго и четвертого слов $w(1) = w_0$, а для первого и третьего слов $w(1) = w_1$, допишем справа к первым четырем входным словам α и слева к первым четырем, выходным словам β (табл. 5.2). Точно так же в табл. 5.1 при $z(1) = z_1$ для пятого, шестого и седьмого слов $w(1) = w_0$, а для восьмого слова $w(1) = w_1$, поэтому справа к пятому, шестому, седьмому и восьмому входным словам припишем букву α и слева к соответствующим выходным словам припишем букву β. Если при этом считать, что $z(1)$ преобразуется в β, то для однобуквенных начальных отрезков слов получилось однозначное преобразование (см. табл. 5.2).

Таблица 5.2

$z(1)$	$z(2)$	$z(3)$	$z(4)$	$w(1)$	$w(2)$	$w(3)$	$w(4)$
z_0	z_0	z_0	α	β	w_1	w_0	w_0
z_0	z_0	z_1	α	β	w_0	w_0	w_0
z_0	z_1	z_0	α	β	w_1	w_0	w_0
z_0	z_1	z_1	α	β	w_0	w_1	w_1
z_1	z_0	z_0	α	β	w_0	w_0	w_1
z_1	z_0	z_1	α	β	w_0	w_1	w_1
z_1	z_1	z_0	α	β	w_0	w_1	w_0
z_1	z_1	z_1	α	β	w_1	w_0	w_1

Таблица 5.3

$z(1)$	$z(2)$	$z(3)$	$z(4)$	$z(5)$	$w(1)$	$w(2)$	$w(3)$	$w(4)$	$w(5)$
z_0	z_0	z_0	α	α	β	β	w_1	w_0	w_0
z_0	z_0	z_1	α	α	β	β	w_0	w_0	w_0
z_0	z_1	z_0	α	α	β	β	w_1	w_0	w_0
z_0	z_1	z_1	α	α	β	β	w_0	w_1	w_1
z_1	z_0	z_0	α		β	w_0	w_0	w_1	
z_1	z_0	z_1	α		β	w_0	w_1	w_1	
z_1	z_1	z_0	α	α	β	β	w_0	w_1	w_0
z_1	z_1	z_1	α	α	β	β	w_1	w_0	w_1

Рассмотрим двухбуквенные начальные отрезки входных слов. Они совпадают в парах входных слов (см. табл. 5.2). Поскольку комбинация

$z(1)z(2) = z_0z_0$ преобразуется неоднозначно, справа от первых двух входных слов дописываем букву α, а слева от первых двух выходных слов дописываем букву β (табл. 5.3). Точно так же поступаем со второй и четвертой парами входных и выходных слов. В третьей паре слов комбинация $z(1)z(2) = z_1z_0$ преобразуется однозначно (см. табл. 5.2) в $w(1)w(2) = \beta w_0$. По этой причине третью пару входных и выходных слов не изменяем (см. табл. 5.3). Трехбуквенные начальные отрезки входных слов в табл. 5.2 все различны, поэтому они преобразуются однозначно. То же самое относится к четырехбуквенным начальным отрезкам входных слов и к полным входным словам в табл. 5.3. Таким образом, задаваемый табл. 5.3 алфавитный оператор φ', будет автоматным.

Если бы исходный оператор содержал не трехбуквенные, а четырехбуквенные входные и выходные слова, для приведения его к автоматному виду только что рассмотренным способом пришлось бы проверять однозначность преобразования и трехбуквенных начальных отрезков входных слов.

Построим теперь по табл. 5.3 графы автоматов Мили и Мура. Будем при этом предполагать, что последний символ каждого входного слова должен переводить автомат в начальное состояние. Начнем с графа автомата Мили (рис. 5.1).

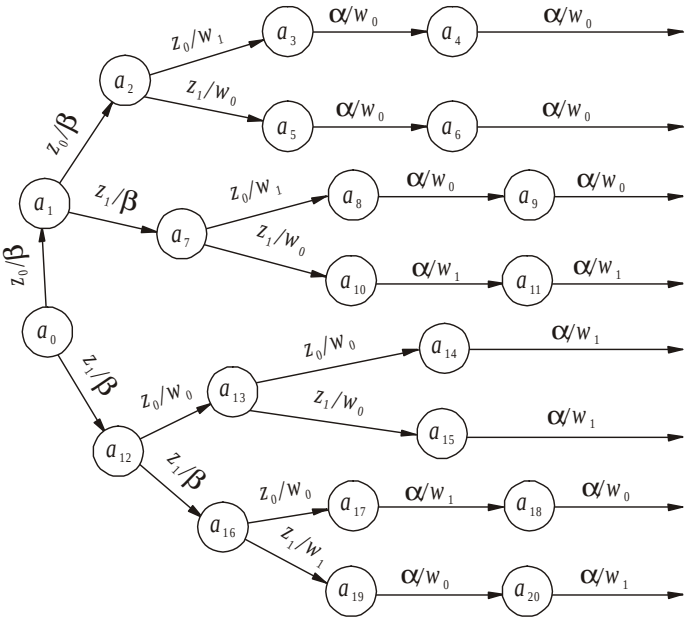


Рис. 5.1

В момент $t = 0$ автомат находится в состоянии a_0. При подаче в последующие моменты времени каждого входного сигнала $z(t)$ автомат переходит в новое состояние и вырабатывает выходной сигнал $w(t)$.

Поскольку для абстрактного автомата порядок нумерации состояний, отличных от a_0, безразличен, можно считать, что буква $z(1) = z_0$ первого входного слова из табл. 5.3 переводит автомат в состояние a_1. При этом вырабатывается выходной сигнал $w(1) = \beta$. Буква $z(2) = z_0$ переводит автомат из состояния a_1 в состояние a_2 и обеспечивает выработку выходного сигнала $w(2) = \beta$. Входная буква $z(3) = z_0$ переводит автомат из a_2 в a_3. Этому переходу соответствует выходной сигнал $w(3) = w_1$. Буквой $z(4) = a$ автомат переводится в состояние a_4 с выдачей выходного сигнала $w(4) = w_0$. Последней во входном слове буквой $z(5) = a$ автомат, согласно условию, должен переводиться в состояние a_0. Этому переходу соответствует выходной сигнал $w(5) = w_0$. Начальные отрезки $z(1)z(2)$, $w(1)w(2)$ второго входного и выходного слов совпадают с соответствующими начальными отрезками первого входного и выходного слов, поэтому первые два перехода для второго входного слова совпадают с уже построенными. Последующие переходы для этого слова строятся точно так же, как и для первого слова. Затем аналогичным образом строятся переходы для остальных входных и выходных слов (см. табл. 5.3).

Граф автомата Мура приведен на рис. 5.2. Он строится почти так же, как и для автомата Мили. Первое отличие состоит в том, что выходные сигналы записываются в вершинах, так как выходной сигнал автомата Мура в каждый момент времени определяется только состоянием и не зависит от входного сигнала.

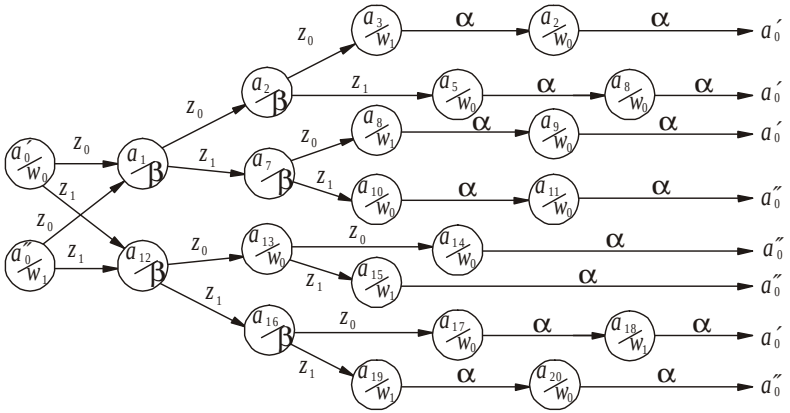


Рис. 5.2

Второе отличие – в числе начальных состояний. Так как последняя буква каждого входного слова должна переводить автомат в начальное состояние, последняя буква каждого выходного слова соответствует начальному состоянию. Поскольку выходные слова в табл. 5.3 содержат две различные последние буквы w_0 и w_1, автомат должен иметь два равноправных начальных состояния a_0' и a_1'', отмеченных выходными сигналами w_0 и w_1 соответственно. Эти состояния должны быть равноправными в том смысле, что при $z(1) = z_0$ автомат из каждого из них переходит в состояние a_1, а при $z(1) = z_1$ автомат из каждого из них переходит в состояние a_{12}.

Автоматы, соответствующие рис. 5.1 и 5.2, имеют входной алфавит $\{z_0, z_1, a\}$. Так как из каждой вершины этих графов выходят дуги, отмеченные не более чем двумя из этих символов, рассматриваемые автоматы будут частичными.

По графам, изображенным на рис. 5.1 и 5.2, легко можно построить таблицы переходов и выходов автоматов.

5.2. Постановка задачи о синтезе автоматов

Синтез автомата заключается в реализации автомата в виде устройства, соответствующего заданию. Для конечных автоматов заданием обычно служит представленный в виде таблицы соответствия алфавитный оператор. Процесс синтеза состоит из двух этапов: этапа абстрактного синтеза и этапа структурного синтеза.

На этапе *абстрактного синтеза* входные и выходные сигналы рассматриваются просто как буквы входного и выходного алфавитов. При рассмотрении состояний интересуются их числом, считая, что при меньшем числе состояний реализация автомата проще. Этот этап синтеза заключается в построении функций переходов и выходов автомата по заданному алфавитному оператору и в нахождении абстрактного автомата с наименьшим числом состояний в некотором смысле эквивалентного исходному.

На этапе *структурного синтеза* выбирается способ представления входных и выходных сигналов через сигналы, принятые за элементарные, а также способ представления состояний автомата через состояния автоматов, принятых за элементарные. Конечной целью структурного синтеза является получение схемы,

состоящей из минимального числа элементов заданного логического базиса.

Возможный путь решения первой части задачи абстрактного синтеза указан в подразд. 5.1. Решению второй части задачи необходимо предпослать несколько определений.

Два вполне определенных абстрактных автомата с общим входным и общим выходным алфавитами называют эквивалентными, если они имеют один и тот же алфавитный оператор.

Два частичных автомата с общим входным и общим выходным алфавитами называют эквивалентными, если их частичные алфавитные операторы имеют одну и ту же область определения и совпадают на этой области. Для частичных автоматов, однако, большее значение имеет не отношение эквивалентности, а отношение эквивалентного продолжения автоматов.

Говорят, что частичный оператор φ продолжает частичный оператор ψ, если область определения оператора φ включает в себя область определения D оператора ψ, и на области D оба оператора совпадают. Частичный автомат B называют эквивалентным продолжением частичного автомата A, если оператор автомата B продолжает оператор автомата A.

Абстрактный синтез автомата завершается нахождением автомата с минимальным числом состояний, эквивалентного заданному вполне определенному автомату или эквивалентно продолжающего заданный частичный автомат. Процесс нахождения такого автомата называется минимизацией абстрактного автомата, а полученный в результате его автомат называют минимальным автоматом.

Первым (предварительным) этапом всякой минимизации является выделение неопределенных выходных сигналов и состояний и внесение соответствующей неопределенности в таблицы переходов и выходов автомата таким образом, чтобы не изменить оператор автомата. С этой целью при задании частичного автомата Мили таблицами переходов и выходов нужно прочеркнуть все места в таблице переходов автомата, которым соответствуют прочеркнутые места в таблице выходов. Если для частичных автоматов Мура считать запрещенными состояния, для которых не определена функция выходов, то в таблице переходов автомата Мура нужно заменить черточками символы запрещенных состояний.

Пусть табл. 5.4 и 5.5 задают функции переходов и выходов частичного автомата Мили.

Таблица 5.4

$a(t-1)$	z_0	z_1
a_0	a_0	a_1
a_1	a_2	a_0
a_2	a_0	a_2

Таблица 5.5

$a(t-1)$	z_0	z_1
a_0	w_0	w_0
a_1	–	w_1
a_2	w_1	–

Таблица 5.6

$a(t-1)$	z_0	z_1
a_0	a_0	a_1
a_1	–	a_0
a_2	a_0	–

Если при каком-либо переходе выходной сигнал не определен, то и состояние, в которое перейдет автомат, не имеет значения. Поэтому после внесения неопределенности из табл.5.5 в табл.5.4 получим табл.5.6. Соответствующий табл.5.4 и табл.5.5 граф автомата Мили приведен на рис. 5.3. На рис. 5.4 приведен граф автомата, соответствующий табл. 5.5 и 5.6.

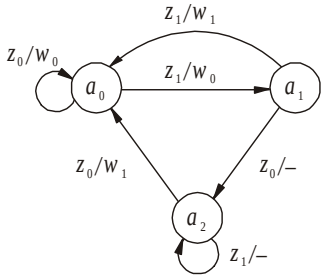


Рис. 5.3

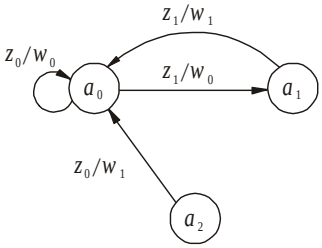


Рис. 5.4

Если исходным является автомат Мура, заданный в табл. 5.7, после внесения неопределенности в таблицу переходов, получим табл. 5.8.

Таблица 5.7

$w(t-1)$	$a(t-1)$	z_0	z_1
w_0	a_0	a_0	a_1
w_1	a_1	a_2	a_0
–	a_2	a_0	a_2

Таблица 5.8

$w(t-1)$	$a(t-1)$	z_0	z_1
w_0	a_0	a_0	a_1
w_1	a_1	–	a_0
–	a_2	a_0	–

Табл.5.7 соответствует граф автомата, приведенный на рис.5.5, а табл. 5.8 – граф автомата, приведенный на рис. 5.6.

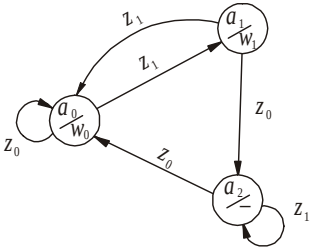


Рис. 5.5

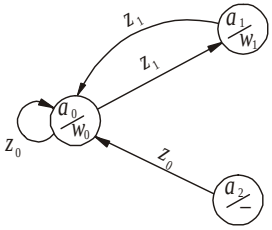


Рис. 5.6

Вторым этапом минимизации является исключение недостижимых состоянии. Состояние a абстрактного (частичного или вполне определенного) автомата называется достижимым, если оно совпадает с начальным состоянием или если существует такое достижимое состояние b и такой входной сигнал z, что $a = \delta(b, z)$. В противном случае состояние a называется недостижимым. Автомат, все состояния которого достижимы, называется связным. Для получения связного автомата нужно вычеркнуть все строки таблиц переходов и выходов автомата, которые обозначены недостижимыми состояниями. Возможность уменьшения числа состояний на этом этапе увеличивается при увеличении степени неопределенности, внесенной в автомат на первом этапе.

Для автомата Мили, заданного табл. 5.4, 5.5 или рис. 5.3, все состояния достижимы, поэтому автомат, заданный табл. 5.4, 5.5 или рис. 5.3, будет связным. Для автомата, заданного табл. 5.5, 5.6 или рис. 5.4, состояние a_2 будет недостижимым, в силу чего автомат не будет связным.

Точно так же автомат Мура, заданный табл. 5.7 или рис. 5.5, является связным, а автомат, заданный табл. 5.8 или рис. 5.6, не будет связным, поскольку у него состояние a_2 недостижимо.

После вычеркивания в табл. 5.5, 5.6 и 5.8 строки, соответствующей состоянию a_2, получим связный автомат Мили, заданный табл. 5.9, 5.10 или рис. 5.7, и связный автомат Мура, заданный табл. 5.11 или рис. 5.8.

Таблица 5.9

$a(t-1)$	z_0	z_1
a_0	a_0	a_1
a_1	–	a_0

Таблица 5.10

$a(t-1)$	z_0	z_1
a_0	w_0	w_0
a_1	–	w_1

Таблица 5.11

$w(t-1)$	$a(t-1)$	z_0	z_1
w_0	a_0	a_0	a_1
w_1	a_1	–	a_0

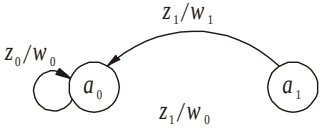


Рис. 5.7

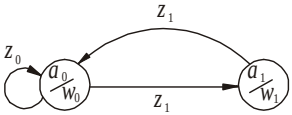


Рис. 5.8

Третьим этапом минимизации автомата является объединение в одно состояние множества совместимых состояний.

5.3. Классы совместимости автомата

Пусть A – произвольный абстрактный автомат, a – любое его состояние. Говорят, что слово p во входном алфавите автомата A применимо к состоянию a, если, подавая это слово на вход автомата A, установленного предварительно в состояние a, мы получим на выходе определенное слово q в выходном алфавите, имеющее ту же самую длину, что слово p. Слово q называют результатом применения слова p к состоянию a.

Если слово p неприменимо к состоянию a, то результат применения слова p к состоянию a считается неопределенным (при этом безразлично, дают ли некоторые непустые начальные отрезки слова p в применении к состоянию a определенные результаты или нет).

Состояния a_{i1}, ..., a_{in}, входящие в автомат Мили, называются *совместимыми*, если все определенные (не считая неопределенных результатов) результаты применения слова p к состояниям a_{i1}, ..., a_{in} будут одними и теми же (зависящими только от слова p, но не от выбора состояния a_{ik} из данного множества состояний). Для автомата Мура, кроме этого условия, для совместимости данных состояний требуется, чтобы (не считая неопределенных отметок) все совместимые состояния имели бы одинаковые отметки. Совместимые состояния во вполне определенных автоматах называются также эквивалентными.

Для конечных автоматов существует конструктивный прием нахождения совместимых состоянии автомата Миля и Мура. Этот прием основан на использовании понятия i-совместимости состояний.

Состояния a_{i1}, ..., a_{in} автомата Мили называются i-совместимыми для любого данного $i = 1, 2, ...$, если (с точностью до неопределенных результатов) результат применения любого слова длины i к состояниям a_{i1}, ..., a_{in} будет одним и тем же, находясь в зависимости лишь от выбо-

ра слова, но не от выбора состояния. Состояния автомата Мура называются 0-совместимыми, если (не считая неопределенных отметок) они одинаково отмечены; они называются i-совместимыми для любого i = 1, 2, ..., если они 0-совместимы и (с точностью до неопределенных результатов) результат применения любого данного слова длины i ко всем рассматриваемым состояниям одинаков.

Очевидно, i-совместимые состояния будут также и φ-совместимыми для любого $j < i$. Состояния тогда и только тогда совместимы, когда они i-совместимы для всех i =1, 2, ... i-классом данного автомата Мили и Мура называется всякое максимальное множество i-совместимых между собой состояний автомата, т. е. такое множество, к которому нельзя добавить ни одного нового состояния без нарушения свойства i-совместимости. Всякое максимальное множество совместимых между собой состояний автомата называют *финальным классом* или *классом совместимости* автомата.

Непосредственно по таблицам выходов могут быть найдены 1-классы для автоматов Мили и 0-классы для автоматов Мура. В случае автомата Мили в один и тот же 1-класс зачисляются все состояния, одинаково обозначающие (с точностью до неопределенных выходных сигналов) строки таблиц выходов. В случае автомата Мура в один и тот же 0-класс зачисляются все одинаково отмеченные состояния и все состояния, отметки которых не определены (последние попадают, таким образом, во все 0-классы).

На этом примере видно, что для частичных автоматов i-классы, вообще говоря, пересекаются между собой. То же самое имеет место и для финальных классов. Для вполне определенных автоматов i-классы не могут пересекаться между собой.

Пусть $K_1(i)$, ..., $K_p(i)$ – совокупности всех i-классов автомата (Мили или Мура). Говорят, что тождество N состояний a_{j1}, ..., a_{jk}, целиком содержащееся в одном из i-классов $K_r(i)$ выдерживает умножение на входную букву z_m, если все состояния $\delta(a_{j1}, z_m)$, $\delta(a_{j2}, z_m)$, ..., $\delta(a_{jk}, z_m)$ (не считая тех, которые не определены) содержатся в одном и том же i-классе $K_t(i)$, зависящем от выбора N и $z_m \in N$.

Нахождение максимальных подмножеств состояний каждого i-класса, выдерживающих умножение на все буквы $z_1, z_2, ..., z_F$ входного алфавита автомата называется операцией расщепления (разбиения) i-классов. Операция расщепления i-классов выполняются очевидным образом с помощью таблицы переходов рассматриваемого автомата.

В результате применения операции расщепления i-классов автомата Мили или Мура возникают все (i+1)-классы этого автомата (i = 1, 2, ...). Ими является все максимальные множества, возникающие в результате расщепления.

Если применять последовательно операцию расщепления i-классов к конечному автомату Мили или Мура, отправляясь от 1-классов (для автомата Мили) или от 0-классов (для автомата Мура), то через конечное число шагов для некоторого $k \geq 0$ процесс расщепления k-классов даст в результате те же самые k-классы. Удовлетворяющие этому условию (нерасщепляемые далее) k-классы будут совпадать с финальными классами исходного автомата. Этот полученный В. М. Глушковым [2] результат является основой минимизации числа состояний конечных автоматов.

5.4. Автомат с минимальным числом состояний

Обозначим c_1, c_2, ..., c_S финальные классы какого-либо автомата A с входным алфавитом $Z = \{z_1, z_2, ..., z_F\}$. Так как финальные классы являются вместе с тем и j-классами для всех j = 0, 1, 2, ..., то для каждой буквы z_k все состояния, входящие в любой финальный класс C_m, порождают один и тот же выходной сигнал (для автомата Мили) или отмечены одним и тем же выходным сигналом (для автомата Мура), либо соответствующие выходные сигналы не определены. Построим таблицу выходов некоторого автомата C, состояниями которого служат финальные классы c_1, c_2, ..., c_S, а входными сигналами – буквы алфавита Z. Для автомата Мили относим каждой паре (c_m, z_k) выходной сигнал, соответствующий паре (a_v, z_k) для любого an из класса c_m, для которого этот сигнал определен. Если же для всех пар (a_v, z_k) соответствующие им выходные сигналы не определены, то считаем, что выходной сигнал пары (c_m, z_k) также не определен. Для автомата Мура отмечаем каждый класс c_m выходным сигналом, которым отмечен произвольный элемент $a_v \in c_m$. Если же все элементы, входящие в c_m не отмечены, то будем считать отметку класса c_m неопределенной.

Таблицу δ_1 переходов автомата C построим по следующему правилу: переход из c_m в $\delta_1(c_m, z_k)$ будет считаться неопределенным, если для состояний a_v, составляющих класс c_m, переход из a_v в $\delta(a_v, z_k)$ не определен. Если же хотя бы для одного состояния $a_v \in c_m$ переход из a_v в $\delta(a_v, z_k)$ определен, то переход из c_m в $\delta_1(c_m, z_k)$ также будет считаться опре-

деленным, а в качестве состояния $\delta_1(c_m, z_k)$ будет приниматься любой из финальных классов c_i (их может быть несколько), содержащий все определенные состояния вида $\delta(a_v, z_k)(a_v \in c_m)$. Очевидно, финальные классы c_i с требуемыми свойствами всегда существуют.

За начальное состояние автомата C можно принять любой финальный класс, содержащий начальное состояние автомата A, либо принять за начальные состояния автомата C некоторые или все финальные классы, содержащие начальное состояние автомата A.

Совокупность E всех финальных классов автомата A удовлетворяет условию полноты и условию замкнутости. Первое условие означает, что каждое состояние автомата A принадлежит какому-либо из финальных классов. Второе условие означает, что для каждой буквы входного алфавита все состояния каждого финального класса (с точностью до неопределенных переходов) переходят в состояния, принадлежащие одному финальному классу (тому же самому или другому). Пусть E_0 – наименьшее подмножество E, удовлетворяющее условиям полноты и замкнутости. Условие замкнутости здесь означает, что для каждой буквы входного алфавита все состояния каждого финального класса из E_0 (с точностью до неопределенных переходов) переходят в состояния, принадлежащие одному (тому же самому или другому) финальному классу из E_0. Назовем минимальной нормальной формой автомата A нормальную форму, построенную по множеству финальных классов E_0.

Если в минимальной нормальной форме C связного автомата A в качестве начального состояния выбран какой-либо финальный класс, содержащий начальное состояние автомата A, то автомат C будет иметь наименьшее число состояний среди всех автоматов, эквивалентно продолжающих автомат A.

Процесс минимизации автомата A можно разбить на два этапа. На первом этапе с помощью какого-либо эвристического приема строится эквивалентный автомату A автомат B с меньшим, чем у A количеством состояний, а на втором этапе стандартным методом осуществляется минимизация автомата A. При этом чем меньше количество состояний имеет автомат B, тем проще будет его последующая минимизация.

Наибольший объем работы по минимизации автомата связан с нахождением финальных классов. Для конечных автоматов эта работа может быть существенно упрощена с помощью треугольных таблиц. Правила составления и использования этих таблиц удобнее всего рассмотреть на конкретных примерах абстрактного синтеза автоматов.

5.5. Пример минимизации автомата Мили

Пусть частичный автомат Мили A задан графом, приведенным на рис. 5.1. Это связный автомат без неопределенных выходных сигналов. Его минимизацию будем осуществлять в два этапа. Сначала построим эквивалентный автомату A автомат B с меньшим чем у A числом состояний.

Согласно рис. 5.1, единственным применимым к состояниям a_4, a_6, a_9 и a_{18} входным словом будет однобуквенное слово $p = \alpha$ с результатом применения $q = w_0$. По этой причине состояния a_4, a_6, a_9 и a_{18} совместимы, и их можно заменить одним состоянием b_4. Точно так же убедимся в совместимости состояний a_{11}, a_{14}, a_{15} и a_{20}, в силу чего их можно заменить одним состоянием b_7. Для состояний a_3, a_5, a_8 применимыми входными словами будут лишь двухбуквенное слово $p_1 = \alpha\alpha$, которому соответствует результат применения $q_1 = w_0w_0$, и однобуквенное слово $p = \alpha$ с результатом применения $q = w_0$. Следовательно, состояния a_3, a_5, a_8 совместимы, и их можно заменить одним состоянием b_3. Переобозначив теперь на рис. 5.1 a_0 на b_0, a_1 на b_1, a_2 на b_2, a_7 на b_5, a_{10} на b_6, a_{12} на b_8, a_{13} на b_9, a_{16} на b_{10}, a_{17} на b_{11} и a_{19} на b_{12}, получим граф автомата Мили B, изображенный на рис. 5.9.

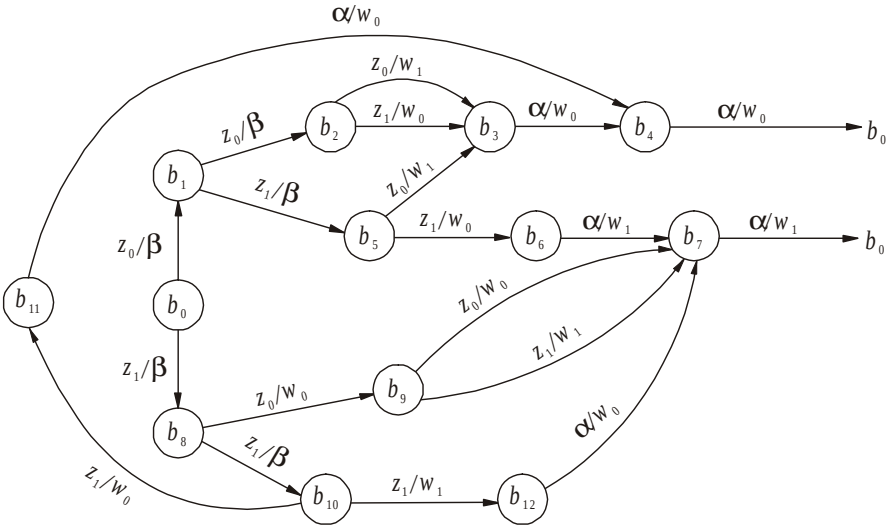


Рис. 5.9

Автомат B имеет только 13 состояний вместо 21 состояния у автомата A. Эти автоматы эквивалентны. Действительно, пройдя все пути из состояния b_0 через другие состояния опять в b_0, убедимся в том, что оператор автомата B определен на всех входных словах из табл. 5.3 и на их начальных отрезках.

Других слов область определения оператора B не содержит. При этом каждому входному слову оператор автомата B сопоставляет то же самое единственное выходное слово, что и оператор автомата A.

Таблица 5.12

$b(t-1)$	α	z_0	z_1
b_0	–	b_1/β	b_8/β
b_1	–	b_2/β	b_5/β
b_2	–	b_3/w_1	b_3/w_0
b_3	b_4/w_0	–	–
b_4	b_0/w_0	–	–
b_5	–	b_3/w_1	b_6/w_0
b_6	b_7/w_1	–	–
b_7	b_0/w_1	–	–
b_8	–	b_9/w_0	b_{10}/β
b_9	–	b_7/w_0	b_7/w_1
b_{10}	–	b_{11}/w_0	b_{12}/w_1
b_{11}	b_4/w_1	–	–
b_{12}	b_7/w_0	–	–

Графу рис. 5.9 соответствует таблица переходов-выходов табл. 5.12 автомата B. Финальные классы этого автомата будем искать с помощью треугольной таблицы (табл. 5.13), которая строится следующим образом.

Строки обозначаются состояниями b_1, b_2, ..., b_{h-1}, а столбцы состояниями b_0, b_1, ..., b_{h-2}, где h – число состояний автомата. На пересечении i-й строки и j-го столбца записываются условия, при которых возможно совмещение состояний b_i и b_j. Если состояния нельзя совместить ни при каких условиях, ставится знак $\times$, если совмещаются безусловно, то знак $\vee$. Клетки, соответствующие пересечениям строк и столбцов с одинаковыми индексами, не заполняются. Окончательное совмещение со-

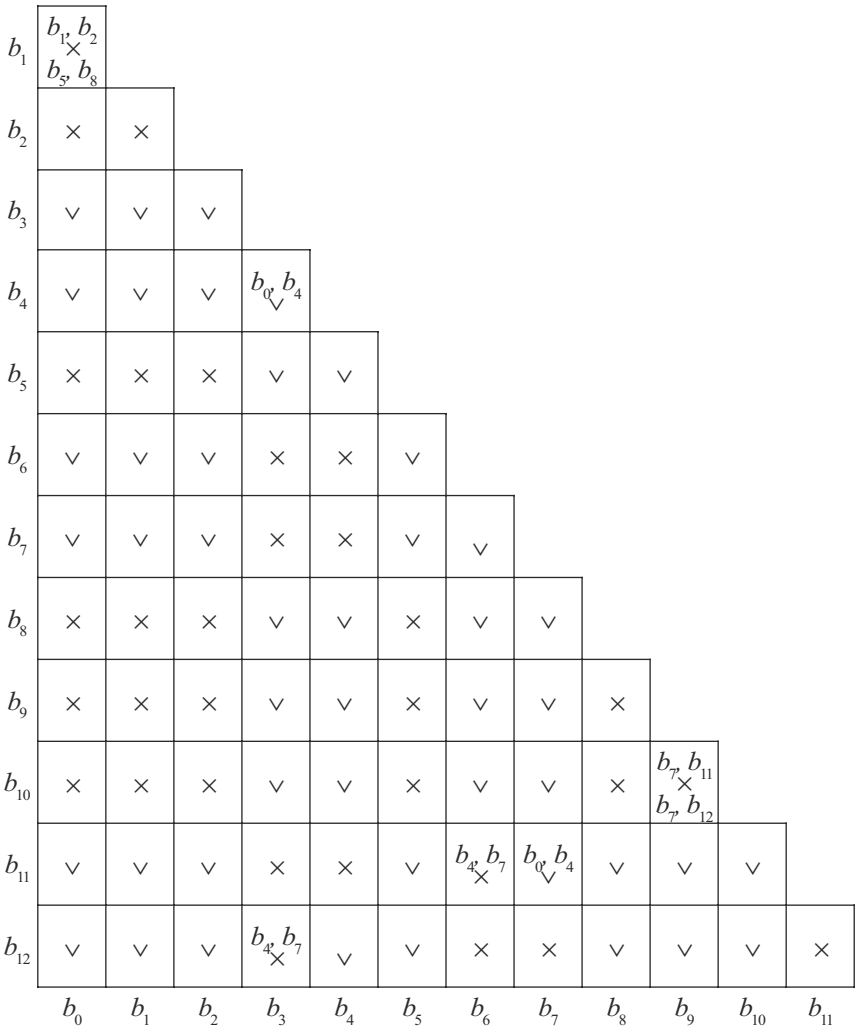
стояний определяется на основании анализа непротиворечивости условий, записанных в клетках. Для рассматриваемого примера треугольная таблица (табл. 5.13) строится следующим образом.

Согласно табл. 5.12, состояниям b_0 и b_1 для каждого входного сигнала соответствуют одинаковые выходные сигналы, поэтому состояния b_0 и b_1 можно совместить, если при каждом входном сигнале они переходят в совместимые состояния, т. е. если можно совместить состояние b_1 с b_2 и состояние b_8 с b_5. Эти условия и записываются в верхнюю клетку, соответствующую столбцу для b_0 (табл. 5.13). Поскольку для состояний b_0 и b_2 найдется входной сигнал, для которого (см. табл. 5.12) выдаются различные входные сигналы, эти состояния совместить невозможно, и соответствующая клетка табл. 5.13 отмечается символом $\times$. Состояния b_0 и b_3 совмещаются безусловно, поскольку при каждом входном сигнале переход определен только для одного из них. В соответствующей клетке табл. 5.13 нужно поставить поэтому символ $\vee$. Затем анализируются остальные пары состояний столбца табл. 5.13, отмеченного состоянием b_0, после чего точно так же анализируются пары состояний, соответствующие остальным столбцам, начиная с верхней строки в каждом столбце. В результате этого анализа все клетки табл. 5.13 окажутся заполненными символами $\times$, $\vee$ или условиями, при которых возможно совмещение состояний, соответствующих клеткам. После этого начинается анализ условий, записанных в клетках табл. 5.13.

Состояния b_0 и b_1 можно совместить в том случае, если совместимы состояния b_1, b_2 и b_5, b_8. Но в клетке, соответствующей состояниям b_1, b_2, стоит знак$\times$, поэтому совместить состояния b_0 и b_1 невозможно и в соответствующей им клетке ставится символ $\times$. Аналогичным образом находим, что и в клетке, отмеченной состояниями b_2, b_4, нужно поставить знак $\times$. В клетке, отмеченной состояниями b_0 и b_4, стоит символ $\vee$, поэтому в клетке, отмеченной состояниями b_3 и b_4, так же нужно поставить $\vee$. В результате после такого анализа клеток, в которых записаны условия совместимости, в каждой клетке табл. 5.13 будет стоять символ $\times$ или символ $\vee$. По такой треугольной таблице можно найти финальные классы.

Процедура нахождения минимального множества финальных классов, предполагающая полный перебор совместимых состояний, приведена в литературе [5]. Поскольку часто эта процедура оказывается гро-

Таблица 5.13



моздкой, рассмотрим упрощенный способ формирования множества финальных классов. Этот способ, к сожалению, не гарантирует получение оптимального результата в случае частичного автомата с большим числом состояний, так как содержит элементы альтернативного неформального выбора при доопределении автомата и в процессе включения

конкретных состояний в финальные классы. Предлагаемый способ может быть описан с помощью следующей процедуры.

1. Формируется множество пар совместимых состояний на основании разметки каждого столбца треугольной таблицы.

Для рассматриваемого автомата Мили (табл. 5.12) по табл. 5.13 находим следующие пары совместимых состояний:

(b_0, b_3), (b_1, b_3), (b_2, b_3), (b_3, b_4), (b_4, b_5), (b_5, b_6),
(b_0, b_4), (b_1, b_4), (b_2, b_4), (b_3, b_5), (b_4, b_8), (b_5, b_7),
(b_0, b_6), (b_1, b_6), (b_2, b_6), (b_3, b_8), (b_4, b_9), (b_5, b_{11}),
(b_0, b_7), (b_1, b_7), (b_2, b_7), (b_3, b_9), (b_4, b_{10}), (b_5, b_{12}),
(b_0, b_{11}), (b_1, b_{11}), (b_2, b_{11}), (b_3, b_{10}), (b_4, b_{12}),
(b_0, b_{12}), (b_1, b_{12}), (b_2, b_{12}),

(b_6, b_7), (b_7, b_8), (b_8, b_{11}), (b_9, b_{11}), (b_{10}, b_{11}),
(b_6, b_8), (b_7, b_9), (b_8, b_{12}), (b_9, b_{12}), (b_{10}, b_{12}).
(b_6, b_9), (b_7, b_{10}),
(b_6, b_{10}), (b_7, b_{11}),

2. На основе полученного множества производится укрупнение групп совместимых состояний, в результате чего должна быть получена полная совокупность финальных классов, в каждом из которых все состояния совместимы между собой. Для уменьшения сложности этой процедуры возможно при формировании каждого очередного класса совместимых состояний не рассматривать состояния, уже вошедшие в какой-либо из сформированных классов. В частности, в рассматриваемом случае состояние b_0 совместимо с состояниями b_3 и b_4, причем состояние b_3 так же совместимо с состоянием b_4, следовательно, их можно включить в один финальный класс K_1. Состояние b_0 совместимо и с состоянием b_6, однако b_6 не совместимо с состоянием b_3 и не может входить в этот же финальный класс. Аналогичная ситуация обнаруживается при анализе совместимых с b_0 состояний b_7, b_{11}, b_{12}. Подобным же образом формируются и все остальные финальные классы. В результате будут получены следующие классы совместимых состояний:

$K_1 = \{b_0, b_3, b_4\}$, $K_5 = \{b_8\}$,
$K_2 = \{b_1, b_6, b_7\}$, $K_6 = \{b_9\}$,
$K_3 = \{b_2, b_{11}\}$, $K_7 = \{b_{10}\}$.
$K_4 = \{b_5, b_{12}\}$,

3. Производится анализ полученных финальных классов на удовлетворение условиям полноты и замкнутости. Проверка условия полноты очевидна и не вызывает затруднений, а проверка условия замкнутости выполняется по таблице переходов-выходов для всех состояний автомата. Если какие-либо два состояния, входящие в один из выбранных финальных классов, переводятся некоторым входным сигналом в состояния, принадлежащие разным классам, то нужно выбрать новое решение и проверить его на замкнутость.

Для рассматриваемого автомата при проверке финальных классов на замкнутость оказывается, что принадлежащие классу K_2 состояния b_6 и b_7 сигналом a переводятся в состояния, принадлежащие разным классам ($b_6 \text{–}a\rightarrow b_7 \in K_1$, а $b_7 \text{–}a\rightarrow b_0 \in K_2$), т. е. условие замкнутости не выполняется. Следовательно, необходимо одно из этих состояний (например, b_7) перенести в другой класс, не нарушая при этом условие замкнутости. Таким классом может быть класс K_5. В результате будет получено окончательное множество финальных классов, совпадающих с состояниями минимизированного автомата C

$$c_0 = K_1 = \{b_0, b_3, b_4\}, \quad c_3 = K_4 = \{b_5, b_{12}\}, \quad c_6 = K_7 = \{b_{10}\}.$$
$$c_1 = K_2 = \{b_1, b_6\}, \quad c_4 = K_5 = \{b_7, b_8\},$$
$$c_0 = K_3 = \{b_2, b_{11}\}, \quad c_5 = K_6 = \{b_9\}.$$

4. Строится таблица переходов-выходов минимизированной нормальной формы заданного автомата. Далее по этой таблице можно построить граф переходов минимизированного автомата. Состояния переходов и выходные сигналы в этом автомате определяются по переходам и выходным сигналам того состояния, которое наиболее полно определено.

Таблица 5.14

$c(t\text{–}1)$	α	z_0	z_1
c_0	c_0 / w_0	c_1 / β	c_4 / β
c_1	c_4 / w_1	c_2 / β	c_3 / β
c_2	c_0 / w_1	c_0 / w_1	c_0 / w_0
c_3	c_4 / w_0	c_0 / w_1	c_1 / w_0
c_4	c_0 / w_1	c_5 / w_0	c_6 / β
c_5	–	c_4 / w_0	c_4 / w_1
c_6	–	c_2 / w_0	c_3 / w_1

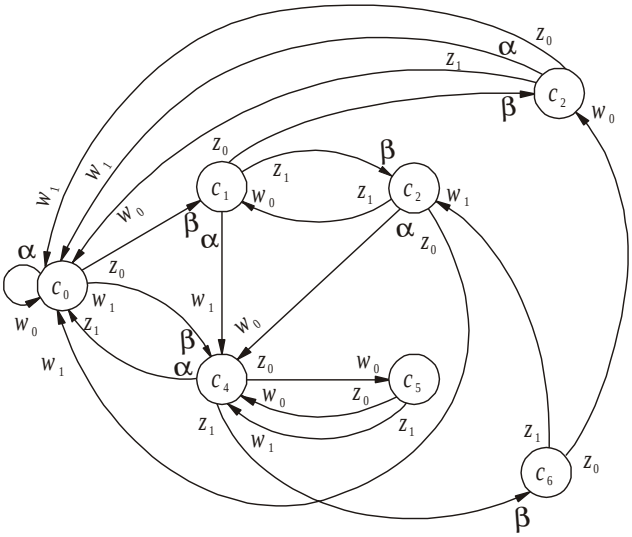


Рис. 5.10

В рассматриваемом примере получены таблица переходов-выходов минимизированного автомата (табл. 5.14) и граф переходов (рис. 5.10).

5.6. Пример минимизации автомата Мура

Пусть частичный автомат Мура A задан графом, представленным на рис. 5.2. Поскольку все состояния этого автомата достижимы, а выходные сигналы определены для всех состояний, отпадает необходимость исключать недостижимые состояния. Будем опять минимизировать автомат в два этапа. Сначала построим эквивалентный автомату A автомат B с меньшим числом состояний.

Согласно рис. 5.2, если автомат A установлен в любое из состояний a_4, a_6, a_9, то единственным допустимым входным словом будет однобуквенное слово $p = \alpha$, которому соответствует единственное однобуквенное слово $p = 0$. По этой причине состояния a_4, a_6, a_9 совместимы и их можно заменить одним состоянием b_4. Аналогичным образом находим, что совместимы состояния a_{11} и a_{15}, a_{14} и a_{20}.

Первую пару заменим состоянием b_8, вторую – состоянием b_{13}. Если автомат установлен в состояния a_3 и a_8, то допустимыми вход-

ными словами будут лишь двухбуквенное слово $p_1 = \alpha\alpha$, которому соответствует выходное слово $q_1 = w_0 w_0$ и однобуквенное слово $p = \alpha$, которому соответствует выходное слово $q = w_0$. Следовательно, состояния a_3 и a_8 совместимы, и их можно заменить одним состоянием. Переобозначив теперь на рис. 5.2 a_0'' на b_0'', a_0' на b_0', a_1 на b_1, a_2 на b_2, a_5 на b_5, a_7 на b_6, a_{10} на b_7, a_{11} на b_8, a_{12} на b_9, a_{13} на b_{10}, a_{16} на b_{11}, a_{19} на b_{12}, a_{20} на b_{13}, a_{17} на b_{14}, a_{18} на b_{15}, получим граф автомата Мура B, изображенный на рис. 5.11.

Автомат B имеет 17 состояний вместо 22 у автомата A и эти автоматы эквивалентны.

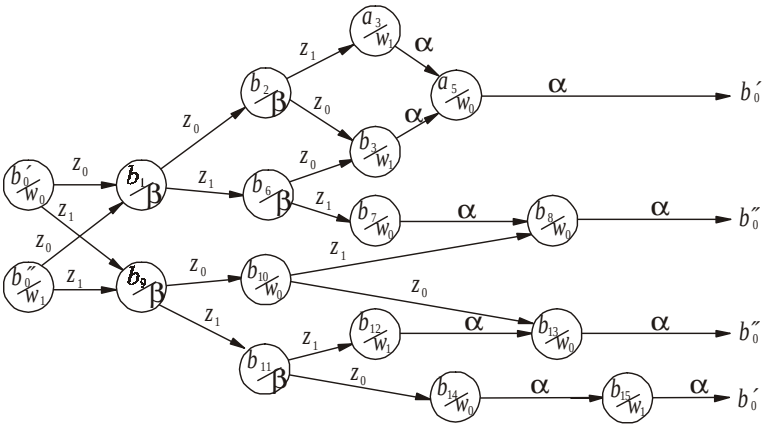


Рис. 5.11

Действительно, пройдя все пути из начальных состояний через другие состояния опять в начальные, убеждаемся в том, что оператор автомата определен на всех входных словах из табл. 5.3 и на начальных отрезках и только на них. При этом каждому входному слову он сопоставляет то же самое единственное выходное слово, что и оператор автомата A.

Графу рис. 5.11 соответствует отмеченная таблица переходов (табл. 5.15) автомата B. Финальные классы автомата B будем находить с помощью треугольной таблицы (табл. 5.16), которая заполняется так же, как и для автомата Мили. Финальные классы находятся из нее для автомата Мура по тем же правилам, что и автомата Мили.

Таблица 5.15

$w(t-1)$	$b(t-1)$	α	z_0	z_1
w_0	b_0'	–	b_1	b_9
w_1	b_0''	–	b_1	b_9
β	b_1	–	b_2	b_6
β	b_2	–	b_3	b_5
w_1	b_3	b_4	–	–
w_0	b_4	b_0'	–	–
w_0	b_5	b_4	–	–
β	b_6	–	b_3	b_7
w_0	b_7	b_8	–	–
w_1	b_8	b_0''	–	–
β	b_9	–	b_{10}	b_{11}
w_0	b_{10}	–	b_{13}	b_8
β	b_{11}	–	b_{14}	b_{12}
w_1	b_{12}	b_{13}	–	–
w_0	b_{13}	b_0''	–	–
w_0	b_{14}	b_{15}	–	–
w_1	b_{15}	b_0'	–	–

Из табл. 5.16 имеем следующие пары совместимых состояний:

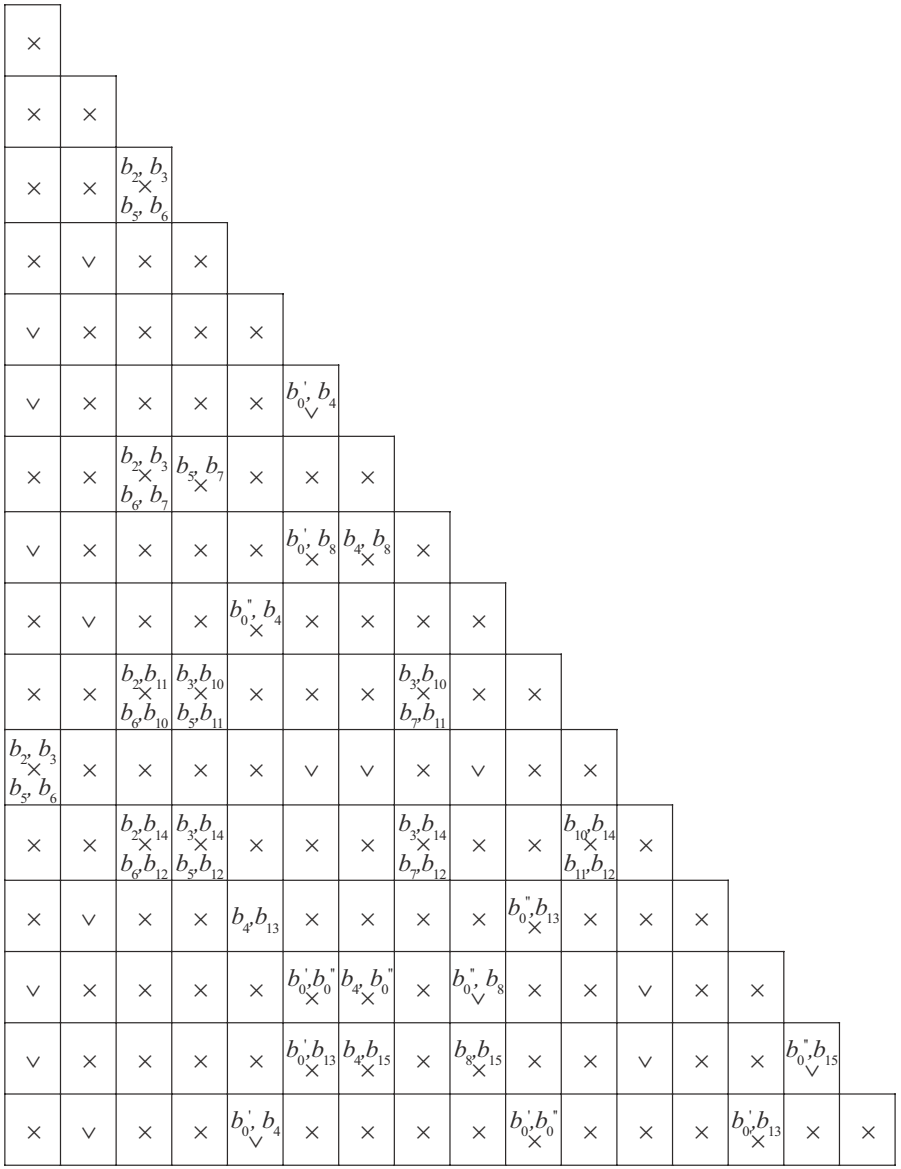
(b_0', b_4), (b_0'', b_3), (b_3, b_{15}), (b_4, b_5), (b_5, b_{10}),
(b_0', b_5), (b_0'', b_8), (b_4, b_{10}),
(b_0', b_7), (b_0'', b_{12}),
(b_0', b_{13}), (b_0'', b_{15}),
(b_0', b_{14}),

(b_7, b_{10}), (b_{10}, b_{13}), (b_{13}, b_{14}).
(b_7, b_{13}), (b_{10}, b_{14}),

После проведения укрупнения групп совместимых состояний получим финальные классы и соответствующие им состояния минимального автомата C.

$C_0' = K_1 = \{b_0', b_4, b_5\}$, $C_5 = K_7 = \{b_8\}$,
$C_0'' = K_2 = \{b_0'', b_3, b_{15}\}$, $C_6 = K_8 = \{b_9\}$,

Таблица 5.16


×
× ×
× × b_2, b_3 × b_5, b_6
× ∨ × ×
∨ × × × ×
∨ × × × × b_0', b_4 ∨
× × b_2, b_3 × b_6, b_7 b_5, b_7 × × × ×
∨ × × × × b_0', b_8 × b_4, b_8 × ×
× ∨ × × b_0'', b_4 × × × × ×
× × b_2, b_{11} × b_6, b_{10} b_3, b_{10} × b_5, b_{11} × × × b_3, b_{10} × b_7, b_{11} × ×
b_2, b_3 × b_5, b_6 × × × × × ∨ ∨ × ∨ × ×
× × b_2, b_{14} × b_6, b_{12} b_3, b_{14} × b_5, b_{12} × × × b_3, b_{14} × b_7, b_{12} × × b_{10}, b_{14} × b_{11}, b_{12} ×
× ∨ × × b_4, b_{13} × × × × b_0'', b_{13} × × × × ×
∨ × × × × b_0', b_0'' × b_4, b_0'' × × b_0'', b_8 ∨ × × ∨ × × × ×
∨ × × × × b_0', b_{13} × b_4, b_{15} × × b_8, b_{15} × × ∨ × × b_0'', b_{15} ∨
× ∨ × × b_0', b_4 ∨ × × × × b_0', b_0'' × × × × b_0', b_{13} × × × ×

$C_1 = K_3 = \{b_1\}$, $C_7 = K_9 = \{b_{11}\}$,
$C_2 = K_4 = \{b_2\}$, $C_8 = K_{10} = \{b_{12}\}$,
$C_3 = K_5 = \{b_6\}$, $C_9 = K_{11} = \{b_{13}, b_{14}\}$.
$C_4 = K_6 = \{b_1, b_{10}\}$,

Полученное множество финальных классов удовлетворяет условиям полноты и замкнутости, в чем легко убедиться с помощью табл. 5.15. Для этого множества строим нормальную форму автомата, которая и будет минимальным автоматом эквивалентно продолжающим автомат *A*. Отмеченная таблица переходов минимального автомата приведена в табл. 5.17. По ней на рис. 5.12 построен граф минимального автомата Мура.

Таблица 5.17

$w(t-1)$	$c(t-1)$	α	z_0	z_1
w_0	c_0'	c_0'	c_1	c_6
w_1	c_0''	c_0'	c_1	c_6
β	c_1	–	c_2	c_3
β	c_2	–	c_0''	c_0'
β	c_3	–	c_0''	c_4
w_0	c_4	c_5	c_9	c_5
w_1	c_5	c_0''	–	–
β	c_6	–	c_4	c_7
β	c_7	–	c_9	c_8
w_1	c_8	c_9	–	–
w_0	c_9	c_0''	–	–

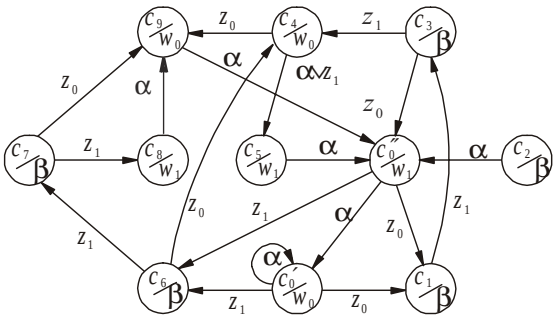


Рис. 5.12

С помощью табл. 5.3 легко убеждаемся в том, что все допустимые входные слова оператора автомата A и их начальные отрезки принадлежат области определения оператора минимального автомата. По рис. 5.12 находим, что входное слово $p = z_0 z_0 z_1 \alpha\alpha\alpha$ принадлежит области определения оператора минимального автомата. Согласно табл. 5.3, это слово не принадлежит области определения оператора исходного автомата A. Таким образом, минимальный автомат эквивалентно продолжает заданный.

6. СТРУКТУРНЫЙ СИНТЕЗ АВТОМАТОВ

6.1. Композиция автоматов

Структурный синтез автоматов осуществляется на базе структурной теории автоматов, в которой в отличие от абстрактной теории производится учет большого числа свойств реально существующих цифровых автоматов. Абстрактный автомат представляет собой математическую модель проектируемого устройства. В структурном же автомате учитывается структура входных и выходах сигналов, а также внутренняя структура автомата на уровне так называемых структурных схем. В структурной теории автоматов принят отсчет автоматного времени, начиная с 0 такта, т. е. $t = 0, 1, 2, ...$

Главной задачей структурной теории автоматов является нахождение общих приемов построения структурных схем автомата на основе композиции элементарных автоматов, принадлежащих к заранее заданному конечному числу типов. Рассмотрим, как представляется автомат в структурной теории автоматов. У абстрактного автомата один входной и один выходной каналы. В структурной же теории как входные, так и выходные каналы автомата считаются состоящими из нескольких элементарных входных и соответственно элементарных выходных каналов. По этим каналам передаются элементарные сигналы. Набор всех возможных для данного автомата элементарных сигналов называется структурным алфавитом данного автомата. Каждый элементарный входной канал подсоединен к входному узлу автомата, а каждый элементарный выходной канал к выходному узлу автомата.

Сформулируем определение общего способа *композиции автоматов*. Пусть $\{A_1, A_2, ..., A_n\}(n \geq 0)$ – конечное множество автоматов. Необходимо произвести объединение этих автоматов в систему совместно работающих автоматов. Введем некоторое конечное множество узлов, которые назовем внешними входными узлами, и некоторое конечное множество других узлов, которые назовем внешними выходными узлами. Входные и выходные узлы автоматов $A_1, A_2, ..., A_n$ будем называть внутренними входными и выходными узлами.

Композиция автоматов состоит в том, что в полученной системе, состоящей из заданных автоматов $A_1, A_2, ..., A_n$ и внешних узлов, производится отождествление некоторых узлов (как внешних, так и внутрен-

них). У цифровых автоматов операции отождествления узлов соответствует соединение этих узлов проводниками.

После проведения отождествления узлов произвольная система автоматов превращается в так называемую схему или сеть автоматов. Будем считать, что автоматы, входящие в схему, работают совместно, если в каждый момент t автоматного времени (t = 0, 1, 2, ...) на все внешние входные узлы схемы подается какой-либо набор элементарных сигналов (структурный входной сигнал), а со всех внешних выходных узлов схемы снимается полученный на них набор элементарных выходных сигналах (структурный выходной сигнал). Если в каждый момент дискретного времени t = 0, 1, 2, ... структурный выходной сигнал схемы однозначно определяется поступившей к этому времени конечной последовательностью структурных входных сигналов, начальными состояниями, входящих в схему автоматов, и сделанными при построении схемы отождествлениями (соединениями) узлов, то построенная схема может рассматриваться как некоторый автомат A и называется структурной схемой этого автомата. Автомат A, полученный описанным способом, есть результат композиции автоматов A_1, A_2, ..., A_n.

Следует заметить, что операция отождествления узлов может быть выполнена неоднозначно. При отождествлении узлов необходимо соблюдать два условия, называемые условиями корректности построения структурных схем:

в любой момент автоматного времени на каждый узел схемы (как внешний, так и внутренний) поступает какой-либо элементарный сигнал;

неоднозначность элементарных сигналов в каком-нибудь узле схемы хотя бы один момент автоматного времени является недопустимой.

Первое условие корректности удовлетворяется в том случае, если любой узел схемы будет подключен через конечное число комбинационных схем или автоматов Мили к внешнему узлу, поскольку у автоматов Мили и комбинационных схем выходные сигналы возникают одновременно с поступлением сигналов на входы.

Неоднозначность элементарного сигнала в узле может появиться по двум причинам: к некоторому входному узлу подсоединено одновременно несколько выходных узлов, являющихся источниками элементарных сигналов, а также источником неоднозначности могут быть так назы-

ваемые циклические цепи или петли структурных схем. Цепью называется последовательность автоматов, у каждого из которых один из выходных узлов соединен с входом последующего. Если выходной узел последнего автомата соединен со входным узлом первого, то такая цепь называется циклической или петлей (рис. 6.1).

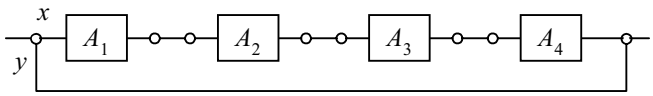


Рис. 6.1

Если A_1, A_2, A_3, A_4 – автоматы Мили, у которых выходной сигнал появляется одновременно со входным, то на входе автомата возникает неоднозначность элементарного сигнала, поскольку неизвестно, какой сигнал считать истинным – входной или поступивший по линии обратной связи. Автомат, построенный таким образом, называется некорректным. Некорректности можно избежать, если хотя бы один из входящих в петлю автоматов будет автоматом Мура, который реагирует на тот или иной входной сигнал выходным сигналом, возникающим на один такт автоматного времени позже, чем вызывавший его появление входной сигнал. Таким образом, для соблюдения второго условия корректности необходимо и достаточно, чтобы любой входной узел схемы соединялся не более, чем с одним внешним входным или внутренним выходным узлом схемы, и любая нетривиальная (содержащая не менее одного автомата) петля в схеме содержала бы в своем составе хотя бы один автомат Мура.

Схема, в которой выполняются все условия корректности, называется правильной.

6.2. Канонический метод структурного синтеза автоматов

Задачей структурного синтеза конечных автоматов является создание такой композиции некоторого множества автоматов, называемых элементарными, чтобы полученный автомат был эквивалентен заданному. Эта задача имеет решение, если система элементарных автоматов структурно полна.

Одним из широко используемых методов структурного синтеза является так называемый *канонический метод*, теоретические основы

которого были разработаны З.М. Глушковым, сформулировавшим и доказавшим *теорему о структурной полноте* [2].

Теорема: всякая система элементарных автоматов, которая содержит автомат Мура, обладающий полной системой переходов и полной системой выходов, и какую-нибудь функционально полную систему логических элементов (элементарных автоматов без памяти), является структурно полной системой. Существует общий конструктивный прием (канонический метод структурного синтеза), позволяющий в рассмотренном случае свести задачу синтеза произвольных конечных автоматов к задаче структурного синтеза комбинационных схем.

Полнота системы переходов в автомате означает, что для любой пары состояний (a_i, a_j) этого автомата найдется входной сигнал, переводящий один элемент этой пары в другой. Это положение справедливо, если $i = j$. Чтобы детерминированный автомат мог удовлетворять условию полноты переходов, число его входных сигналов должно быть не меньше числа состояний.

Полнота системы выходов в автомате Мура означает, что каждому состоянию автомата соответствует свой собственным выходной сигнал, отличный от выходного сигнала, соответствующего любому другому состоянию. Поэтому в автомате Мура с полной системой выходов можно отождествить внутренние состояния с выходными сигналами автомата. Для того, чтобы обеспечить в этом автомате полноту системы выходов, необходимо в выходном алфавите иметь число выходных сигналов не меньше числа состояний автомата.

На основании теоремы о структурной полноте структурная схема всякого автомата A, синтезированного каноническим методом, будет состоять из двух частей: запоминающей части и комбинационной схемы (рис. 6.2). Запоминающая часть представляет собой совокупность элементарных автоматов Мура с полной системой переходов и выходов, а комбинационная часть представляет собой схему, построенную из логических элементов, составляющих функционально полный базис. Рассмотрим этапы структурного синтеза автомата каноническим методом.

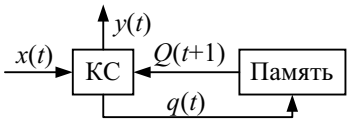


Рис. 6.2

1. Кодирование состояний абстрактного автомата. В процессе структурного синтеза различным состояниям заданного абстрактного автомата a_i ставятся в соответствие различные упорядоченные последовательности состояний элементарных автоматов Q_1, Q_2, ..., Q_p. Этот процесс называется кодированием состояний автомата.

Результатом кодирования состояний является возникновение структурных состояний автомата $Q^l = Q_1^{\,l}\, Q_2^{\,l} \ldots Q_R^{\,l}$, где $l = 0, 1, 2, \ldots, M$ (M+1) – число состояний абстрактного автомата. $R =]\log_2(M+1)[$ и $]b[$ означает ближайшее целое число, большее b или равное ему, если b – целое. Эти состояния можно отождествить со структурными выходными сигналами запоминающей части автомата.

В связи с тем, что каждый структурный выходной сигнал памяти автомата представляет собой совокупность элементарных сигналов, поступающих по отдельным каналам, будем считать структурный выходной сигнал памяти векторным сигналом $\mathbf{Q}^l$, в котором каждая компонента отождествляется с буквой структурного алфавита состояния $Q = \{Q_1, Q_2, \ldots, Q_\mathrm{P}\}$, т. е. $Q_i^l \in Q$.

2. Кодирование абстрактных входных и выходных сигналов. Абстрактным входным и выходным сигналами $z_i \in Z$ и $w_i \in W$, где Z и W – входной и выходной абстрактные алфавиты, ставятся в соответствие внешние структурные входные и выходные сигналы автомата, обозначаемые соответственно $x^j = x_1^{\,j}\, x_2^{\,j} \ldots x_\mathrm{L}^{\,j}$, и $y^k = y_1^{\,k}\, y_2^{\,k} \ldots y_\mathrm{N}^{\,k}$, где $j = 1, 2, \ldots, F$, F – число символов входного абстрактного алфавита, $k = 1, 2, \ldots, G$, G – число символов выходного абстрактного алфавита, $L =]\log_2 F[$ и $N =]\log_2 G[$. Сигналы $\boldsymbol{x}^j$ и $\boldsymbol{y}^k$ являются векторными сигналами, компоненты которых x_i^j и $\mathrm{y}_i^{\,k}$ – соответственно элементарные входные и выходные сигналы на каждом элементарном входном или выходном канале, т. е. $x_i^j \in X = \{x_1, x_2, \ldots, x_n\}$, а $y_i^k \in Y = \{y_1, y_2, \ldots, y_r\}$, – где X и Y – соответственно структурные входной и выходной алфавиты.

3. Составление кодированных таблиц переходов-выходов структурного автомата. В процессе синтеза необходимо обеспечить, чтобы переходы из одной последовательности состояний элементарных автоматов в другую происходили в полном соответствии с функцией переходов заданного абстрактного автомата, а значения структурных выходных сигналов вырабатывались в соответствии с заданной последовательностью абстрактных входных сигналов. Таким образом, должны быть обеспечены следующие законы функционирования для структурного автомата Мили:

$$\mathbf{Q}(t+1) = \delta(\mathbf{Q}(t), \mathbf{x}(t)),$$
$$\mathbf{y}(t) = \lambda(\mathbf{Q}(t), \mathbf{x}(t)) \quad (6.1)$$

и для структурного автомата Мура

$$\mathbf{Q}(t+1) = \delta(\mathbf{Q}(t), \mathbf{x}(t)),$$
$$\mathbf{y}(t) = \lambda(\mathbf{Q}(t)). \quad (6.2)$$

Закон функционирования структурного автомата может быть описан с помощью кодированных таблиц переходов-выходов, которые формируются на основании таблиц переходов-выходов абстрактного автомата и полученных на первых двух этапах структурных значений состояний и сигналов автомата. В клетках этих таблиц вместо символов, обозначающих абстрактные состояния и сигналы, записываются коды соответствующих им структурных состоянии и сигналов.

4. Формирование таблицы функций возбуждения структурного автомата. Из рис. 6.2 следует, что

$$\mathbf{q}(t) = \gamma(\mathbf{Q}(t), \mathbf{x}(t)). \quad (6.3)$$

Сравнивая выражение (6.3) с функцией переходов структурного автомата (6.1) или (6.2) можно сделать вывод, что именно сигналом $\mathbf{q}(t)$ можно осуществить требуемые переходы при условии формирования некоторой функции $\gamma(\mathbf{Q}, \mathbf{x})$, называемой функцией возбуждения автомата A , в соответствии с функцией переходов $\delta(\mathbf{Q}, \mathbf{x})$. В дальнейшем для упрощения формулировок сигнал $\mathbf{q}(t)$ будем также называть функцией возбуждения автомата A. Отдельные компоненты вектора $\mathbf{q}(t)$ являются входными сигналами используемых элементарных автоматов.

Функции возбуждения задаются о помощью таблицы, сформированной на базе структурной таблицы переходов проектируемого автомата, и таблицы переходов заданного элементарного автомата. В клетках таблицы функций возбуждения записываются значения структурных входных сигналов выбранных элементарных автоматов, обеспечивающие переходы их состояний в соответствии с кодированной таблицей переходов.

5. Получение логических выражений функций возбуждения и выходных сигналов автоматов. Сигнал $\mathbf{q}(t)$ является структурным выходным сигналом комбинационной схемы автомата A, отличающейся тем, что ее структурный выходной сигнал получается путем преобразования структурного входного сигнала $\mathbf{x}(t)$ автомата A и структурного выходного сигнала $\mathbf{Q}(t)$ его памяти. Реализуя схему преобразования в виде композиции заданных логических элементов, можно осуществить все

те переходы, которые предусматриваются функцией переходов автомата A. Аналогично формируется структурный выходной сигнал $\mathbf{y}(t)$ на выходе комбинационной схемы автомата A. Требуемые значения выходного сигнала обеспечиваются соответствующим синтезом этой схемы.

Для получения логических выражений функций возбуждения и выходных сигналов необходимо воспользоваться соответственно таблицей функций возбуждения и кодированной таблицей выходов в качестве таблиц истинности.

Как известно, по таблице истинности можно получить только совершенную дизъюнктивную и нормальную форму (СДНФ) заданной функции. Для того чтобы минимизировать функцию, целесообразно воспользоваться диаграммой Вейча.

6. Построение структурной схемы. На основании полученных логических выражений строится схема структурного автомата из заданных элементарных автоматов и логических элементов функционально полного базиса.

Рассмотрим в качестве примера синтез структурной схемы автомата Мили, спроектированного в разделе 5, функции переходов которого описываются в табл. 5.14.

Обычно в качестве структурного алфавита используется двоичный структурный алфавит {0, 1}, а в качестве элементарного автомата используется автомат Мура с двумя состояниями.

Пусть имеем абстрактный элементарный автомат с входным алфавитом $V=\{v_1,v_2\}$, выходным алфавитом и алфавитом состояний $C=\{c_1,c_2\}$ и таблицей переходов-выходов (табл. 6.1).

Таблица 6.1

	c_1	c_2
v_1	c_1	c_2
v_2	c_2	c_1

Произведем операцию кодирования состояний, входных и выходных сигналов у элементарного и исходного автоматов. Поскольку структурный алфавит двоичный, число элементарных входных каналов автомата определяется как $L \geq]\log_2 F[$, число элементарных выходных каналов определяется как $N \geq]\log_2 G[$, а число элементов памяти автомата – как $R \geq]\log_2(M+1)[$.

Перейдем от абстрактного элементарного автомата к структурному. Закодируем абстрактные входные сигналы (табл. 6.2), выходные сигналы и состояния (табл. 6.3). Получим кодированную таблицу переходов-выходов элементарного автомата (табл. 6.4).

Таблица 6.2

v	q
v_1	0
v_2	1

Таблица 6.3

c	Q
c_1	0
c_2	1

Таблица 6.4

	0	1
0	0	1
1	1	0

Произведем также кодирование абстрактных входных сигналов (табл. 6.5), выходных сигналов (табл. 6.6) и состояний (табл. 6.7) заданного автомата, определив при этом число элементов памяти. Число элементарных входных каналов проектируемого структурного автомата $L \geq]\log_2 3[= 2$. Число его элементарных выходных каналов $N \geq]\log_2 3[= 2$ и число элементов памяти $R \geq]\log_2 7[= 3$. Следовательно, схема проектируемого структурного автомата должна иметь вид, представленный на рис. 6.3. Определим структурный входной алфавит автомата (табл. 6.5), структурный выходной алфавит (табл. 6.6) и алфавит состояний (табл. 6.7). Разобьем таблицу переходов-выходов для данного абстрактного автомата (табл. 5.14) на две таблицы: таблицу переходов (табл. 6.8) и таблицу выходов (табл. 6.9).

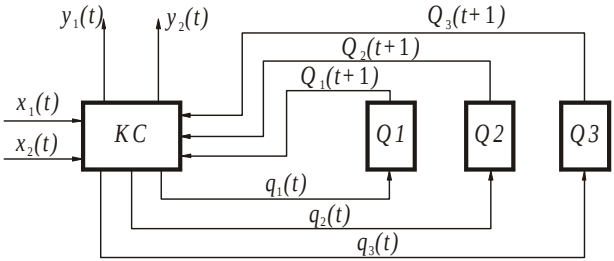


Рис. 6.3

Таблица 6.5

z	x_1	x_2
z_0	0	0
z_1	0	1
α	1	0

Таблица 6.6

w	y_1	y_2
w_0	0	0
w_1	0	1
β	1	0

Таблица 6.7

c	Q_1	Q_2	Q_3
c_0	0	0	0
c_1	0	0	1
c_2	0	1	0
c_3	0	1	1
c_4	1	0	0
c_5	1	0	1
c_6	1	1	0

На основании табл. 6.8 и 6.7 построим кодированную таблицу переходов автомата

(табл. 6.10), а на основании табл. 6.6 и 6.9 построим кодированную таблицу выходов (табл. 6.11).

Таблица 6.8

$c(t-1)$	α	z_1	z_2
c_0	c_0	c_1	c_4
c_1	c_4	c_2	c_3
c_2	c_0	c_0	c_0
c_3	c_4	c_0	c_1
c_4	c_0	c_5	c_6
c_5	–	c_4	c_4
c_6	–	c_2	c_3

Таблица 6.9

$c(t-1)$	α	z_1	z_2
c_0	w_0	β	β
c_1	w_1	β	β
c_2	w_1	w_1	w_0
c_3	w_0	w_1	w_0
c_4	w_1	w_0	β
c_5	–	w_0	w_1
c_6	–	w_0	w_1

Таблица 6.10

Q_1	Q_2	Q_3	X_1X_2 10	X_1X_2 00	X_1X_2 01
0	0	0	000	001	100
0	0	1	100	010	011
0	1	0	000	000	000
0	1	1	100	000	001
1	0	0	000	101	110
1	0	1	–	100	100
1	1	0	–	010	011

Таблица 6.11

Q_1	Q_2	Q_3	X_1X_2 10	X_1X_2 00	X_1X_2 01
0	0	0	00	10	10
0	0	1	01	10	10
0	1	0	01	01	00
0	1	1	00	01	00
1	0	0	01	00	10
1	0	1	–	00	01
1	1	0	–	00	01

В каждой клетке табл. 6.10 переходов записывается двоичный код соответствующего состояния автомата, а в каждой клетке табл. 6.11 выходов – двоичный код соответствующего выходного сигнала. Далее, пользуясь табл. 6.10 и 6.4, можно построить таблицу функций возбуждения структурного автомата (табл. 6.12), для чего в каждой клетке этой таблицы необходимо записать значения требуемых входных сигналов q_1, q_2 и q_3 для каждого из трех элементов памяти Q_1, Q_2 и Q_3 в зависимости от фиксируемого в соответствующей клетке табл. 6.10 перехода из одного состояния в другое. Далее по табл. 6.11 и 6.12 сформируем логические выражения для функций возбуждения и выходного сигнала.

Чтобы получить минимальные ДНФ функций возбуждения q_1, q_2, q_3 и выходных сигналов Q_1, Q_2, Q_3, построим диаграммы Вейча для каж-

дой из искомых функций. Аргументами всех функций возбуждения в соответствии с выражением (6.3) и табл. 6.12 являются состояния элементарных автоматов Q_1, Q_2, Q_3 и входные сигналы x_1 и x_2. Т. е.

$$q_1 = \gamma_1(Q_1(t), Q_2(t), Q_3(t), x_1(t), x_2(t)),$$
$$q_2 = \gamma_2(Q_1(t), Q_2(t), Q_3(t), x_1(t), x_2(t)),$$
$$q_3 = \gamma_3(Q_1(t), Q_2(t), Q_3(t), x_1(t), x_2(t)).$$

Аргументами для y_1 и y_2 в соответствии с выражением (6.1) и табл. 6.11 также являются Q_1, Q_2, Q_3, x_1 и x_2. Таким образом,

Таблица 6.12

Q_1	Q_2	Q_3	X_1X_2 10	X_1X_2 00	X_1X_2 01
0	0	0	000	001	100
0	0	1	101	011	010
0	1	0	010	010	010
0	1	1	111	011	010
1	0	0	100	001	010
1	0	1	–	001	001
1	1	0	–	100	101

$$y_1(t) = \lambda_1(Q_1(t), Q_2(t), Q_3(t), x_1(t), x_2(t)),$$
$$y_2(t) = \lambda_2(Q_1(t), Q_2(t), Q_3(t), x_1(t), x_2(t)).$$

В результате проведенной минимизации получим следующие логические выражения для функций возбуждения:

$$q_1 = x_1\overline{Q}_2\overline{Q}_3 \vee x_1Q_2Q_3 \vee x_1\overline{Q}_1\overline{Q}_2\overline{Q}_3 \vee Q_1Q_2,$$

$$q_2 = \overline{x}_1\overline{Q}_1Q_3 \vee x_2Q_1\overline{Q}_2\overline{Q}_3 \vee \overline{Q}_1Q_2,$$

$$q_3 = x_2Q_1Q_3 \vee \overline{x}_2\overline{Q}_1Q_3 \vee x_2Q_1Q_2 \vee \overline{x}_1\overline{x}_2Q_1\overline{Q}_2.$$

Поскольку на комбинационной схеме выходные сигналы вырабатываются практически одновременно с моментом поступления входных сигналов, временные соотношения могут быть в выражениях опущены.

Логическое выражение для выходного сигнала y_2 целесообразно получать, минимизируя y_2 по “0”, а не по “1”, поскольку в этом случае получается выражение с меньший рангом. В результате имеем

$$y_1 = \overline{x}_1\overline{Q}_1\overline{Q}_2 \vee x_2\overline{Q}_2\overline{Q}_3,$$

$$y_2 = x_2\overline{Q}_1 \vee x_1Q_2Q_3 \vee x_1\overline{x}_2Q_1 \vee \overline{x}_2Q_1Q_2 \vee \overline{x}_1\overline{x}_2\overline{Q}_2 \vee \overline{Q}_1\overline{Q}_2\overline{Q}_3.$$

На базе полученных выражений можно построить структурную схему автомата в любом функционально полном базисе.

7. ЭЛЕМЕНТАРНЫЕ АВТОМАТЫ

Из теоремы о структурной полноте следует, что для структурной полноты системы элементарных автоматов необходимо включение в нее элементарных автоматов Мура с полной системой переходов и полной системой выходов. Теоретически такие автоматы, представляющие собой элементы памяти, могут обладать любым числом внутренних состояний, однако исходя из реальных возможностей современной технологии, оптимальным числом состояний элементарного автомата является два, а структурный алфавитом состояний автомата является двоичный алфавит.

Рассмотрим некоторую обобщенную модель элементарного автомата, представляющую собой автомат Мура, заданный следующим множеством элементов:

$$A = \{C, V, \delta, \lambda, c_1\},$$

где $C = \{c1, c2\}$ – алфавит состояний; $V = \{v_1, v_2, v_3, v_4\}$ – входной алфавит, причем v_1 – сигнал, не меняющий исходное состояние автомата, такой, что $c_1 = \delta\{c_1, v_1\}$, $c_2 = \delta\{c_2, v_1\}$; v_2 – сигнал, переводящий автомат в состояние, противоположное исходному, такой, что $c_1 = \delta\{c_2, v_2\}$, $c_2 = \delta\{c_1, v_2\}$; v_3 – сигнал, всегда переводящий автомат в состояние c_1, такой, что $c_1 = \delta\{c_1, v_3\}$, $c_1 = \delta\{c_2, v_3\}$; v_4 – сигнал, всегда переводящий автомат в состояние v_4, такой, что $c_2 = \delta\{c_1, v_4\}$, $c_2 = \delta\{c_2, v_4\}$; c_1 – начальное состояние; δ и λ – функция переходов и функция выходов, определяемые с помощью графа переходов (рис. 7.1).

Поскольку в автомате Мура с полной системой выходов внутренние состояния отождествляются с выходными сигналами, для их обозначения использован один и тот же алфавит (в данном случае алфавит C).

На базе рассмотренной модели можно построить 16 элементов памяти с различными комбинациями абстрактных входных сигналов, но только семь из них будут обладать полной системой переходов и полной системой выходов. Автоматы с одним входным сигналом не могут обладать полнотой, поскольку для этого необходимо, чтобы число абстрактных входных сигналов автомата было, по крайней мере, не меньше числа его состо-

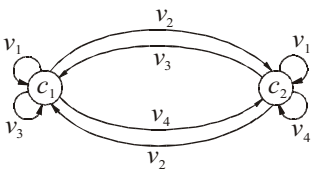


Рис. 7.1

яний. Из автоматов с двумя входными сигналами только два из шести удовлетворяют приведенному требованию. Это автоматы, в которых в качестве входного алфавита используется алфавит $V_1 = \{v_3, v_4\}$ (автомат A_1) и алфавит $V_2 = \{v_1, v_2\}$ (автомат A_2). Полным также является автомат A_3 с тремя входными сигналами с алфавитом $V_3 = \{v_1, v_3, v_4\}$, а также автомат A_4 с четырьмя входными сигналами.

Перечисленные элементарные автоматы и их модификации являются наиболее широко используемыми элементами памяти в современных цифровых устройствах.

Рассмотрим структурные особенности этих автоматов, считая, что любой структурный элементарный автомат с двумя состояниями имеет вид, представленный на рис. 7.2. Поскольку этот автомат является автоматом Мура, выходной сигнал задерживается относительно входного на один такт автоматного времени. В соответствии с теоремой о структурной полноте схему любого структурного элементарного автомата можно представить состоящей из двух частей: запоминающей части, в которой не производится логическое преобразование информации (элемент задержки сигнала t) и комбинационной схемы (рис. 7.3). Приведенная на рис. 7.3 функция возбуждения $q_{эл}(t)$ представляет собой собственную функцию возбуждения элемента памяти. Следует заметить, что для задания структурного элементарного автомата целесообразно использовать матрицу переходов, элементами которой являются значения структурных входных сигналов автомата, заданные на упорядоченных парах состояний структурного автомата и переводящие первый элемент соответствующей пары во второй.

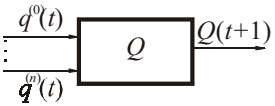


Рис. 7.2

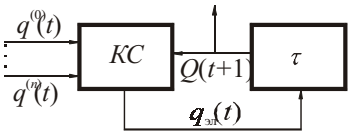


Рис. 7.3

Для рассматриваемой модели существует четыре возможных перехода структурных состояний: 0→0, 0→1, 1→0, 1→1. Для каждого из этих переходов найдется значение входного сигнала, вызывающего заданный переход. Тогда закон функционирования элементарного автома-

та, имеющего m элементарных входных каналов, можно описать следующей матрицей переходов:

$Q(t)$	$Q(t+1)$
0	0
0	1
1	0
1	1

$$\mathbf{M} = \begin{vmatrix} q_1 & q_2 & & q_k & & q_m \\ b_{00}^1 & b_{00}^2 & \cdots & b_{00}^k & \cdots & b_{00}^m \\ b_{01}^1 & b_{01}^2 & \cdots & b_{01}^k & \cdots & b_{01}^m \\ b_{10}^1 & b_{10}^2 & \cdots & b_{10}^k & \cdots & b_{10}^m \\ b_{11}^1 & b_{11}^2 & \cdots & b_{11}^k & \cdots & b_{11}^m \end{vmatrix}$$

где $Q(t)$ и $Q(t+1)$ – состояния структурного элементарного автомата в последовательные моменты автоматного времени. Количество строк матрицы **M** для любого элементарного автомата с полной системой переходов равно четырем, а количество столбцов равно числу входных каналов.

Элемент матрицы b_{ij}^k представляет собой значение входного сигнала q_k, под действием которого автомат переходит из состояния i в состояние j. При этом каждый элемент матрицы может быть равен 1, 0 или неопределенному коэффициенту b. Неопределенные коэффициенты записываются в том случае, когда значения сигналов, поступающих на данный вход, не влияют на рассматриваемый переход.

7.1. Элементарные автоматы с двумя входными сигналами

Для проведения операции структурного синтеза с целью получения структурной схемы элементарного автомата A_1, граф переходов которого приведен на рис. 7.4, необходимо провести операцию кодирования состояний автомата и его входных сигналов. Поскольку абстрактному автомату с двумя входными и двумя выходными сигналами соответствует структурный автомат с одним входным и одним выходным каналом, результат операции кодирования входных сигналов и состояний будет иметь вид, представленный соответственно в табл. 7.1 и 7.2.

Таблица 7.1

v	q
v_3	0
v_4	1

Таблица 7.2

c	Q
c_1	0
c_2	1

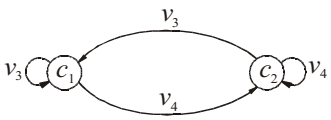


Рис.7.4

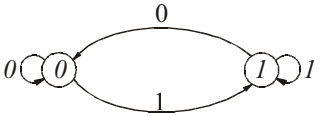


Рис.7.5

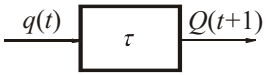


Рис. 7.6

На основании рис.7.4 и табл.7.1 и 7.2 можно составить кодированную таблицу переходов структурного элементарного автомата A_1 (табл.7.3), кодированный граф переходов (рис. 7.5), а также его матрицу переходов

$Q(t)$	$Q(t+1)$		q
0	0		0
0	1	**M** =	1
1	0		0
1	1		1

(7.1)

Рассматривая табл. 7.3 и матрицу (7.1), можно заключить, что значение состояния автомата $Q(t+1)$ и соответствующего ему выходного сигнала определяются значением входного сигнала $q(t)$ и не зависят от состояния автомата $Q(t)$, т. е. $Q(t+1) = q(t)$. В то же время на основании анализа схемы, приведенной на рис. 7.3, можно сделать вывод, что

$$Q(t+1) = q_{эл}(t).$$

Следовательно,

$$q(t) = q_{эл}(t)$$

и комбинационная схема в таком случае становится для автомата A_1 излишней.

Таким образом, схема автомата приобретает вид, представленный на рис. 7.6, а сам автомат A_1 представляет собой элемент задержки и называется триггером типа D (*delay* – задержка).

Таблица 7.3

q	0	1
0	0	0
1	1	1

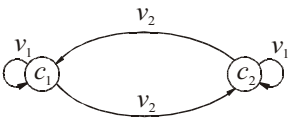


Рис.7.7

Перейдем к рассмотрению схемы структурного элементарного автомата A_2, граф переходов которого представлен на рис. 7.7. В процессе кодирования входных сигналов и состояний автомата можно сформировать соответственно табл. 7.4 и 7.2, из которых следует, что структурный автомат A_2 имеет так же, как и автомат A_1, только один входной канал. На базе графа переходов автомата A_2 и табл. 7.4 и 7.2 получаем кодированный граф переходов (рис. 7.8), кодированную таблицу переходов (табл. 7.5) и матрицу переходов (7.2) автомата A_2.

Таблица 7.4

v	q
v_1	0
v_2	1

Таблица 7.5

q	0	1
0	0	1
1	1	0

Проведем операцию синтеза структурной схемы автомата A_2. Из табл. 7.5 можно получить логическое выражение для состояния и соответствующего ему выходного сигнала автомата

$$Q(t+1) = q(t)\overline{Q}\,(t) \vee \overline{q}\,(t)Q(t) = q(t) \oplus Q(t).$$

Поскольку в любом элементарном автомате

$$Q(t+1) = q_{эл}(t),$$

можно записать

$$q_{эл}(t) = q(t) \oplus Q(t).$$

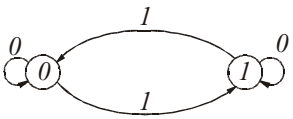


Рис. 7.8

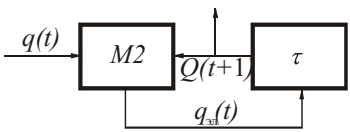


Рис. 7.9

Следовательно, автомат A_2 может быть реализован на базе элемента задержки с добавлением комбинационной схемы, выполняющей операцию сложения по модулю 2, которая может быть построена в любом логическом базисе (рис. 7.9). Этот автомат называется триггером со счетным входом или триггером типа *T*.

7.2. Элементарные автоматы с тремя входными сигналами

Так же, как и для автоматов с двумя входными сигналами, в этом случае в зависимости от комбинаций и кодирования трех используемых входных сигналов можно спроектировать различные структурные схемы элементарных автоматов.

Рассмотрим автомат A_3, использующий входной алфавит $V_3 = \{v_1, v_3, v_4\}$, граф переходов которого представлен на рис. 7.10.

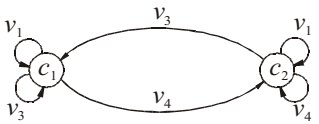


Рис. 7.10

Таблица 7.6

v	$q^{(0)}$	$q^{(1)}$
v_1	0	0
v_2	1	0
v_3	0	1

Закодируем состояния автомата A_3 (табл. 7.2) и его входные сигналы. Поскольку автомат A_3 имеет три абстрактных входных сигнала, число его структурных входных каналов должно равняться двум и может быть закодировано способом, представленным в табл. 7.6. Комбинация структурных входных сигналов {1, 1} является запрещенной комбинацией для данного типа элементарного автомата. На основании табл. 7.2, 7.6 и рис. 7.10 построим кодированный граф переходов автомата A_3 (рис. 7.11), кодированную таблицу переходов (табл. 7.7) и матрицу переходов (7.3). Символ “*b*” в матрице переходов означает безразличное значение данного входного сигнала для указанного перехода. Прослеживая по табл. 7.7 переходы автомата A_3, можно убедиться, что единичное значение сигнала $q^{(0)}$ переводит автомат в нулевое состояние независимо от значения его исходного состояния, и, наоборот, единичное значение сигнала $q^{(1)}$ переводит автомат всегда в единичное состояние. Благодаря этой особенности автомата A_3 его входные каналы называются соответственно нулевым ($q^{(0)}$) и единичным ($q^{(1)}$).

$Q(t)$	$Q(t+1)$		$q^{(0)}$	$q^{(0)}$
0	0		b	0
0	1	**M** =	0	1
1	0		1	0
1	1		0	b

(7.3)

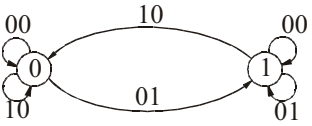


Рис. 7.11

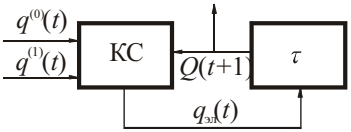


Рис. 7.12

Логическое выражение для фикции $q_{эл}$ автомата A_3 можно получить из табл. 7.7, доопределив неопределенные переходы единицами. В результате получим

$$q_{эл}(t) = Q(t+1) = \overline{q}^{(0)}(t)Q(t) \vee q^{(1)}(t).$$

Таблица 7.7

$q^{(0)}$	$q^{(1)}$	0	1
0	0	0	1
0	1	1	1
1	0	0	0
1	1	–	–

Переведя полученное выражение в базис Шеффера, получим

$$q_{эл}(t) = S(S(S(q^{(0)}(t)),\ Q(t)),\ S(q^{(1)}(t)).$$

Можно получить другое выражение для $q_{эл}$ автомата A_3, доопределив запрещенные наборы не единицами, а нулями. В результате имеем

$$\overline{q}_{эл}(t) = q^{(0)}(t) \vee \overline{q}^{(1)}(t)\ \overline{Q}(t). \qquad (7.4)$$

Взяв отрицание от обеих частей выражения (7.4) и переведя его в базис Пирса, получим

$$q_{эл}(t) = P(q^{(0)}(t),\ P(q^{(1)}(t), Q(t))). \qquad (7.5)$$

Таким образом, комбинационная схема элементарного автомата представляет собой последовательное соединение либо элементов Пирса (для прямого значения входных сигналов), либо двух элементов Шеффера (для инверсного значения входных сигналов) и имеет два входных канала (рис. 7.12). Этот автомат получил название триггера с раздельными входами или триггера типа *RS* (*set* – установить; *reset* –сбросить), где с помощью *R* и *S* обозначены соответственно входы $q^{(0)}$ и $q^{(1)}$.

Наличие неопределенных состояний, отмеченных в табл. 7.7 прочерками, ограничивает функциональные возможности *RS*-триггера. В ряде случаев требуется, чтобы состояния триггера были определены при всех комбинациях входных сигналов, включая и те, которые запрещены для *RS*-триггера.

Для построения различных модификаций *RS*-триггера в этом случае делается дублирование одного из трех абстрактных входных сигналов: v_1, v_3 или v_4 и кодирование дополнительного сигнала структурными символами 11. Каждая полученная при этом разновидность триггера считается самостоятельным типом и имеет свое наименование.

Рассмотрим получаемые таким образом автоматы. Переходы, осуществляемые дополнительными сигналами v'_1, v'_3 или v'_4, показаны на графах переходов абстрактных автоматов A_4, A_5, A_6 соответственно на рис. 7.13, 7.14 и 7.15.

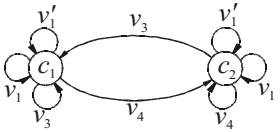


Рис. 7.13

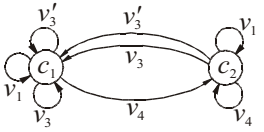


Рис. 7.14

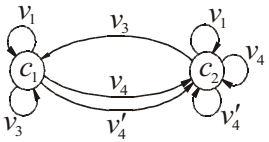


Рис. 7.15

Кодированные графы переходов, соответствующие приведенным абстрактным автоматам, имеют вид, представленный на рис. 7.16, 7.17 и 7.18.

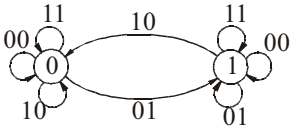


Рис. 7.16

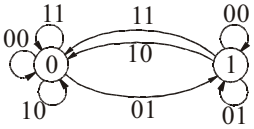


Рис. 7.17

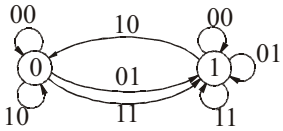


Рис. 7.18

Анализируя переходы автоматов на рис. 7.16, 7.17 и 7.18, можно заметить, что каждый дополнительный сигнал обеспечивает один из четырех возможных переходов. Автомат на рис. 7.16 под действием сигнала 11 остается в прежнем состоянии и называется триггером типа *E* (*exclusive* – исключительный, особенный), автомат на рис. 7.17 под действием сигнала 11 переходит в состояние 0 и называется триггером типа *R*, а автомат на рис. 7.18 переходит в состояние 1 и называется триггером типа *S*. Табл. 7.8 представляет собой общую кодированную таблицу переходов этих трех триггеров.

Таблица 7.8

R	*S*	*E*-триггер		*R*-триггер		*S*-триггер	
		0	1	0	1	0	1
0	0	0	1	0	1	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	1	0	1	0	0	1	1

Матрицы (7.4), (7.5) и (7.6) являются соответственно матрицами переходов триггеров *E*, *R* и *S*.

$$\begin{array}{cc} Q(t) & Q(t+1) \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{array} \qquad \mathbf{M} = \left| \begin{array}{cc} R & S \\ b_1 & b_2 \\ 0 & 1 \\ 1 & 0 \\ b_2 & b_1 \end{array} \right| \qquad (7.4)$$

$Q(t) \quad Q(t+1)$

$0 \text{ ——— } 0$

$0 \text{ ——— } 1$

$1 \text{ ——— } 0$

$1 \text{ ——— } 1$

$$\mathbf{M} = \begin{vmatrix} R & S \\ b_1 & b_2 \\ 0 & 1 \\ 1 & b \\ 0 & b \end{vmatrix} \qquad (7.5) \qquad \mathbf{M} = \begin{vmatrix} R & S \\ b & 0 \\ b & 1 \\ 1 & 0 \\ b_2 & b_1 \end{vmatrix} \qquad (7.6)$$

В матрицах (7.4), (7.5) и (7.6) элементы могут принимать любые значения, кроме комбинации $b_1 = 0$ и $b_2 = 1$ одновременно, т. е. $\bar{b}_1 b_2 = 0$.

Автомат A_4, представленный на рис. 7.13, может быть закодирован иначе, способом, предложенным в табл. 7.9. Тогда его кодированный граф переходов будет иметь вид, представленный на рис. 7.19, а таблица переходов – вид, представленный в табл. 7.10.

Таблица 7.9

v	$q^{(1)}$	$q^{(2)}$
v_1	0	0
v_2	1	1
v_3	1	0
v_1'	0	1

Таблица 7.10

$q^{(1)}$	$q^{(2)}$	0	1
0	0	0	1
1	0	0	1
0	1	0	0
1	1	1	1

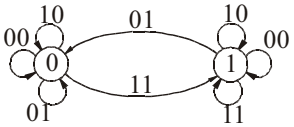


Рис.7.19

Если сравнить таблицу переходов этого автомата с таблицей переходов триггера типа D (табл. 7.3), можно заметить, что при $q^{(2)}(t) = 1$ состояния автомата $Q(t+1)$ в табл. 7.10 так же, как и в табл. 7.3, определяются только значениями входного сигнала $q^{(1)}(t)$. В то же время при $q^{(2)}(t) = 0$ автомат переходит в режим хранения информации (его состояния не меняются) независимо от смены сигналов на входе $q^{(1)}(t)$.

В связи с этим рассматриваемый автомат представляет собой модификацию D-триггера и называется триггером типа DV (*valve* – вентиль, клапан), где буквами D и V обозначены соответственно входы $q^{(1)}$ и $q^{(2)}$. Вход V является разрешающим входом по отношению ко входу D. Матрица переходов триггера типа DV имеет вид

$Q(t)$ $Q(t+1)$

0 ——— 0

0 ——— 1

1 ——— 0

1 ——— 1

$$\mathbf{M} = \begin{vmatrix} D & V \\ b_2 & b_1 \\ b & 1 \\ 1 & 0 \\ b_3 & b_4 \end{vmatrix} \qquad (7.9)$$

где элементы b_1 и b_2 могут принимать любые значения, кроме комбинации $b_1 = b_2 = 1$, т. е. $b_1 b_2 = 0$, а элементы b_3 и b_4 должны удовлетворять условию $\overline{b}_3 b_4 = 0$, т. е. недопустимо, чтобы $b_3 = 0$, а $b_4 = 1$ одновременно.

Функция возбуждения DV-триггера имеет вид

$$q_{эл}(t) = q^{(1)}(t)\, q^{(2)}(t) \vee \overline{q}^{(2)}(t)\, Q(t) = Q(t+1)$$

или иначе

$$Q(t+1) = D(t)V(t) \vee \overline{V}(t)Q(t).$$

7.3. Элементарный автомат с четырьмя входными сигналами

Элементарный автомат A_7, граф переходов которого представлен на рис. 7.20, использует в качестве входного алфавита алфавит $V_4 = \{v_1, v_2, v_3, v_4\}$, содержащий все четыре абстрактных входных сигнала описываемой модели. Закодировав символы входного алфавита в соответствии с табл. 7.11, построим на основании рис. 7.20, табл. 7.2 и 7.11 кодированный граф переходов автомата (табл. 7.12) и матрицу переходов (7.10) этого автомата.

$Q(t)$ $Q(t+1)$

0 ——— 0

0 ——— 1

1 ——— 0

1 ——— 1

$$\mathbf{M} = \begin{vmatrix} q^{(0)} & q^{(1)} \\ b & 0 \\ b & 1 \\ 1 & b \\ 0 & b \end{vmatrix} \qquad (7.10)$$

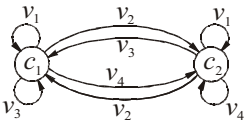


Рис. 7.20

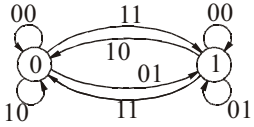


Рис. 7.21

Таблица 7.11

v	$q^{(1)}$	$q^{(2)}$
v_1	0	0
v_2	1	1
v_3	1	0
v_1'	0	1

Таблица 7.12

$q^{(1)}$	$q^{(2)}$	0	1
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

Анализируя переходы автомата A_7 по табл. 7.12, приходим к заключению, что он функционирует одновременно как триггер *RS* (при нулевых и противоположных значениях входных сигналов) и как триггер *T* (при $q^{(0)} = q^{(1)} = 1$). Поэтому автомат A_7 считается универсальным триггером типа *JK*, где входом *J* считается вход $q^{(1)}$, а входом *K* – вход $q^{(0)}$.

Функция возбуждения триггера *JK* имеет вид

$$q_{\text{эл}}(t) = q^{(1)}(t)\ \overline{Q}\ (t) \vee\ \overline{q}^{\,(0)}(t)\ Q(t) = Q(t+1)$$

или иначе

$$Q(t+1) = J(t)\ \overline{Q}\ (t) \vee\ \overline{K}\ (t)Q(t).$$

Структурная схема триггера реализуется обычно на базе триггера типа *RS*. В этом случае входы $q^{(0)}(t)$ и $q^{(1)}(t)$ *RS*-триггера будут иметь значения, определяемые следующими логическими выражениями:

$$q^{(0)}{}_{RS}\,(t) = R(t) = K(t)\ Q(t),$$
$$q^{(1)}{}_{RS}\,(t) = S(t) = J(t)\ \overline{Q}\ (t).$$

Достаточно широкое применение в цифровых схемах находит элементарный автомат, полученный путем избыточного кодирования входных абстрактных сигналов автомата A_7, представленного в табл. 7.13.

Любая комбинация, содержащая два единичных значения элементарных структурных сигналов, является запрещенной.

Таблица 7.13

v	$q^{(0)}$	$q^{(1)}$	$q^{(2)}$
v_1	0	0	0
v_2	0	0	1
v_3	1	0	0
v_4	0	1	0

Таблица 7.14

$q^{(0)}$	$q^{(1)}$	$q^{(2)}$	0	1
0	0	0	0	1
0	0	1	1	1
1	0	0	0	0
0	1	0	1	0

На основании табл. 7.13 и 7.2 и рис. 7.20 получим кодированный граф переходов автомата (рис. 7.22), кодированную таблицу переходов (табл. 7.14) и матрицу переходов (7.11) этого автомата.

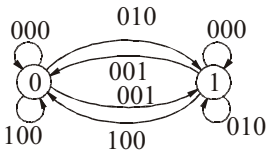


Рис. 7.22

$$
\begin{array}{ll}
Q(t) & Q(t+1) \\
0 & \longrightarrow 0 \\
0 & \longrightarrow 1 \\
1 & \longrightarrow 0 \\
1 & \longrightarrow 1
\end{array}
\qquad
\mathbf{M} = \left|\begin{array}{ccc}
q^{(0)} & q^{(1)} & q^{(2)} \\
b & 0 & 0 \\
0 & b_1 & \overline{b}_1 \\
b_2 & 0 & \overline{b}_2 \\
0 & b & 0
\end{array}\right| \qquad (7.11)
$$

Анализируя переходы автомата по табл. 7.14, можно заметить, что сигналы $q^{(0)}$ и $q^{(1)}$ работают аналогично сигналам R и S триггера типа RS, а сигнал $q^{(2)}$ изменяет состояние автомата на противоположное аналогично входному сигналу триггера T. В связи с этим данный элемен-

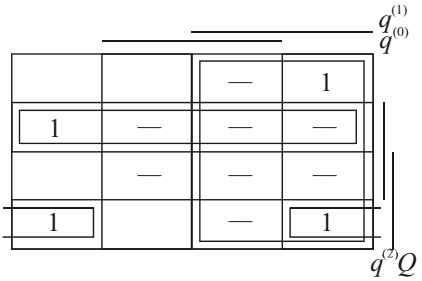


Рис. 7.23

тарный автомат получил название счетного триггера с раздельной установкой или триггера типа RST.

Логическое выражение для функции возбуждения этого триггера может быть получено на основании табл. 7.14 с помощью диаграммы Вейча (рис. 7.23) и будет иметь вид

$$Q(t+1) = q_{эл}(t) = q^{(1)}(t) \vee \overline{q}^{(0)}(t)\ \overline{q}^{(2)}(t)Q(t) \vee q^{(2)}(t)\ \overline{Q}\,(t). \qquad (7.12)$$

Или иначе $Q(t+1) = S(t) \vee T(t)\ \overline{Q}\,(t) \vee \overline{R}\,(t)\ \overline{T}\,(t)Q(t)$.

Структурная схема *RST*-триггера реализуется обычно на базе *RS*-триггера.

8. ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ ЭЛЕМЕНТАРНЫХ АВТОМАТОВ

Для устойчивой работы автомата необходима его синхронизация и введение дополнительных схем (конъюнкторов) [5]. Обычно дополнительные схемы и входы синхронизирующих сигналов предусматриваются в самом элементе памяти при его технической реализации. На основе характеристического уравнения *RS*-триггера

$$Q(t+1) = S(S(S(\overline{R}_{\text{вх}}(t)),\ Q(t)),\ S(\overline{S}_{\text{вх}}(t)))$$

можно построить асинхронную схему этого триггера на элементах Шеффера с управлением по входам, представленную на рис. 8.1.

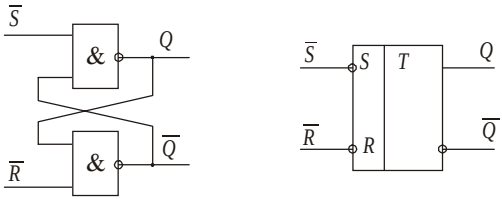


Рис. 8.1

Во избежание гонок [5] асинхронные триггеры обычно не используются в качестве элементов памяти в структурных автоматах. Подключая к входам асинхронного триггера схему управления, состоящую из логических элементов используемого базиса, можно получить синхронный (тактируемый) триггер. На рис. 8.2, *а*, *б* приведена схема синхронного *RS*-триггера и его логическая структура, в которых пунктиром обозначены побочные асинхронные входы. На рис. 8.2, *в* приведено условное изображение такого триггера. Входы *S* и *R* – информационные (входы функций возбуждения), а вход *C* – синхронизирующий (тактовый).

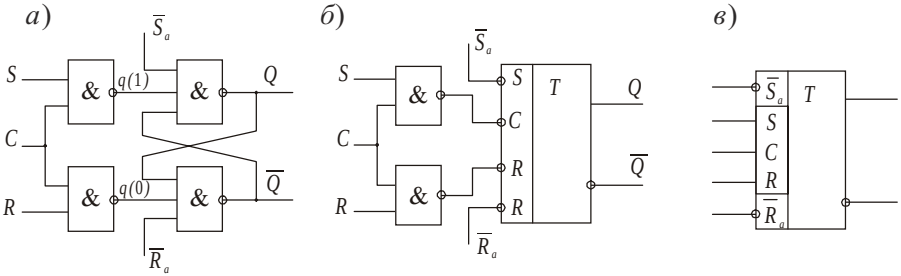


Рис. 8.2

Изменение состояния такого триггера может произойти только при $C = 1$. Побочные входы $\overline{S}_a$ и $\overline{R}_a$ триггера предназначены для асинхронной установки триггера в состояние 0 и 1, минуя информационные и тактирующий входы. Функционирование в этом случае соответствует асинхронному *RS*-триггеру с инверсным управлением. При синхронной работе на побочных входах должна поддерживаться нейтральная комбинация сигналов ($\overline{S}_a = \overline{R}_a = 1$).

Помимо синхронизации для устранения гонок в автоматах используется двойная память. В этом случае все элементы памяти структурного автомата должны быть организованы в виде двухтактных (двухступенчатых) схем. Они строятся на основе двух синхронных триггеров, соединенных последовательно. Синхронизирующие сигналы поступают на эти триггеры в соответствии с условиями организации таких сигналов в схемах с двойной памятью: на вход *C* первого триггера поступает прямой сигнал, а на вход *C* второго триггера – инверсный. Двухтактная схема *RS*-триггера, ее логическая структура и условное изображение приведены соответственно на рис. 8.3, *а*, *б* и *в*.

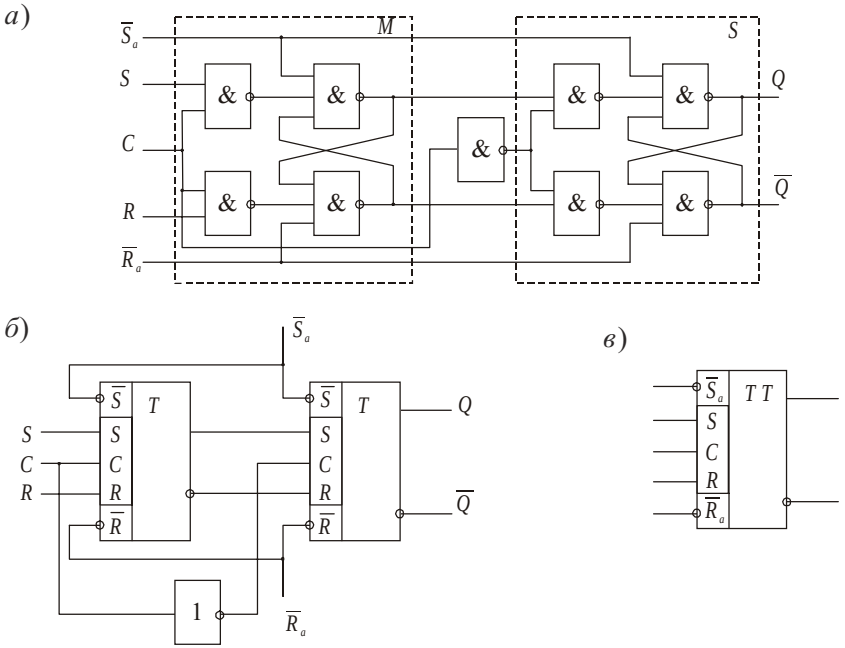


Рис. 8.3

Первый триггер в двухтактной схеме называется ведущим *M*, а второй – ведомым *S* (*M* – *master* (хозяин), *S* – *slave* (невольник)).

Поскольку *RS*-триггер устойчив по отношение к длительности входного сигнала, на его основе целесообразно строить различные варианты схем элементарных автоматов. Рассмотрим реализацию триггеров различных типов на основе синхронного *RS*-триггера.

На рис. 8.4 приведены схемы *S*- и *R*-триггеров, функционирующих в соответствии с табл. 7.8.

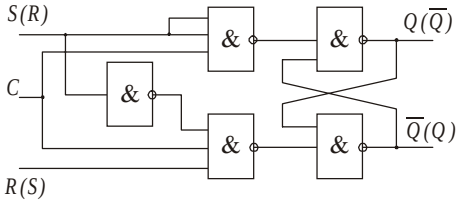


Рис. 8.4

Функциональные схемы *S*-триггера и *R*-триггера одинаковы и конкретный тип триггера определяется наименованием входных и выходных каналов. На рис. 8.4 обозначения входов триггера без скобок соответствуют триггеру типа *S*, а обозначения в скобках – триггеру типа *R*.

На рис. 8.5 приведена функциональная схема *E*-триггера, работающего в соответствии с табл. 7.8.

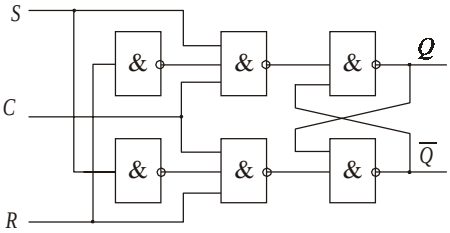


Рис. 8.5

В этой схеме при одновременном сочетании на входах $S = R = 1$ обеспечивается режим хранения информации и состояние триггера не меняется.

Рассмотрим техническую реализацию триггеров типа *D*, *DV*, *T*, *JK*, синтезируя их каноническим методом структурного синтеза, используя в качестве элемента памяти триггер типа *RS*.

Структурная схема любого триггера на базе триггера типа *RS* имеет вид, представленный на рис. 8.6.

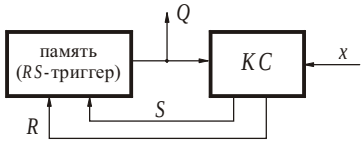


Рис. 8.6

1. Синтез триггеров типа *D* и *DV*. *D*-триггер осуществляет переходы из одного структурного состояния в другое в соответствии с кодированной таблицей переходов 8.1 На основе табл. 8.1 и матрицы переходов *RS*-триггера (7.3) можно составить кодированную таблицу функции возбуждения *D*-триггера (табл. 8.2).

Таблица 8.1

	0	1
0	0	0
1	1	1

Таблица 8.2

D	0		1	
	R	*S*	*R*	*S*
0	*b*	0	1	0
1	0	1	0	*b*

Из табл. 8.2 можно с помощью диаграммы Вейча (рис. 8.7) найти логическое выражение для функции возбуждения *R* и с помощью другой диаграммы Вейча (рис. 8.8) – для функции возбуждения *S*.

$$R = \overline{D} \text{ и } S = D.$$

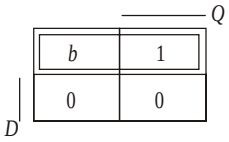


Рис. 8.7

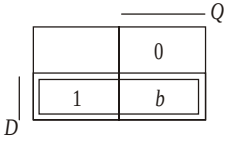


Рис. 8.8

Таким образом, функциональная схема *D*-триггера на основе *RS*-триггера должна состоять из *RS*-триггера с дополнительным инвертором на входе *R*. Логическая структура такого *D*-триггера и его условное обозначение приведены на рис. 8.9, *а* и *б*.

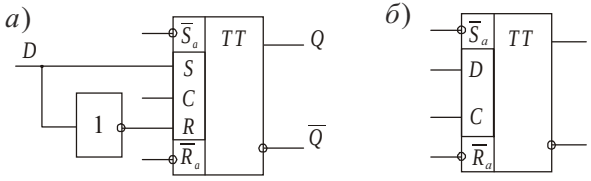


Рис. 8.9

Поскольку *DV*-триггер представляет собой модификацию *D*-триггера, его схему можно получить преобразованием последнего путем добавлением входа *V*, который должен быть логически связан операцией И с управляющим входом *C* (рис. 8.10).

2. Синтез триггера типа *T*. Функционирование *T*-триггера задается кодированной таблицей переходов (табл. 8.3).

Таблица 8.3

	1	0
0	1	0
1	0	1

Таблица 8.4

T	1		0	
	R	*S*	*R*	*S*
0	0	*b*	*b*	0
1	1	0	0	0

Пользуясь матрицей переходов *RS*-триггера (7.3) и табл. 8.3, составим кодированную таблицу функций возбуждения *T*-триггера (табл. 8.4). Из табл. 8.3 можно с помощью диаграммы Вейча (рис. 8.11) найти логическое выражение для функции возбуждения *R* и с помощью другой диаграммы Вейча (рис. 8.12) – для функции возбуждения *S*. Получим $R = TQ$ и $S = T\overline{Q}$, откуда $\overline{R} = \overline{TQ}$ и $\overline{S} = \overline{T\overline{Q}}$. В соответствии с этими выражениями схема асинхронного *T*-триггера будет иметь вид, представленный на рис. 8.13, *а*. На рис. 8.13, *б* представлено условное обозначение асинхронного *T*-триггера. Вместо схемы задержки для обеспечения устойчивой работы *T*-триггера можно использовать двухступенчатую схему (рис. 8.14, *а*).

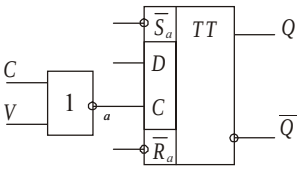


Рис. 8.10

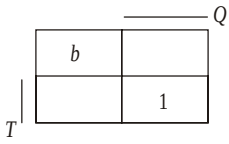


Рис. 8.11

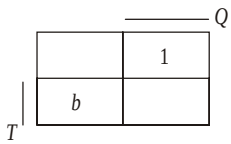


Рис. 8.12

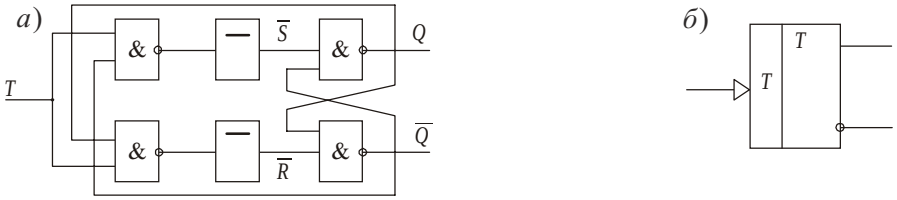


Рис. 8.13

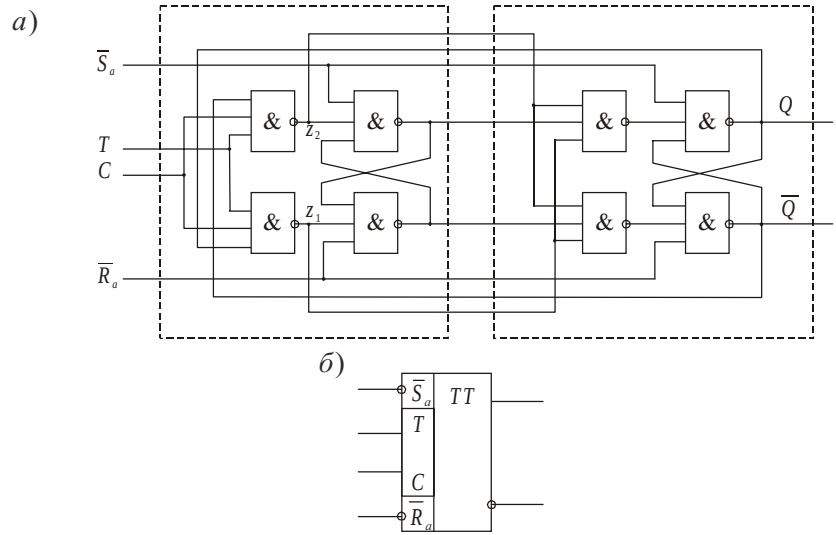


Рис. 8.14

В этой схеме синхронизирующий сигнал $\overline{C}$, подаваемый на второй триггер, образуется конъюнкцией z_1 и z_2 (z_1&z_2), которая равна 1, если $C = 0$, и первый триггер не перебрасывается. Тогда в этот момент во второй триггер происходит перенос информации.

Условное обозначение двухтактного *T*-триггера приведено на рис. 8.14, *б*.

3. Синтез триггера типа *JK*. Работа *JK*-триггера задается кодированной таблицей переходов (табл. 8.5).

Пользуясь матрицей переходов *RS*-триггера (7.3) и табл. 8.5, составим кодированную таблицу функций возбуждения J*K*-триггера (табл. 8.6).

Из табл. 8.6 с помощью диаграммы Вейча (рис. 8.15) можно найти логическое выражение для функции возбуждения *S* и с помощью другой диаграммы (рис. 8.16) – для функции возбуждения *R*.

Таблица 8.5

	1	0
00	0	1
01	1	1
10	0	0
11	1	0

Таблица 8.6

KJ	0		1	
	R	*S*	*R*	*S*
00	*b*	0	0	*b*
01	0	1	0	*b*
10	*b*	0	1	0
11	0	1	1	0

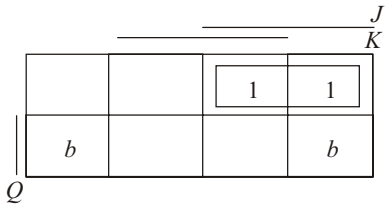


Рис. 8.15

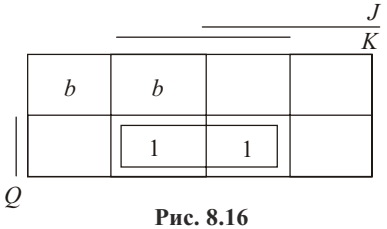


Рис. 8.16

Получим $S = J\overline{Q}$ и $R = KQ$, откуда $\overline{S} = \overline{J\overline{Q}}$ и $\overline{R} = \overline{KQ}$.

В соответствии с этими выражениями схема асинхронного *JK*-триггера будет иметь вид, представленный на рис. 8.17, *а*. На рис. 8.17, *б* представлено условное изображение асинхронного *JK*-триггера.

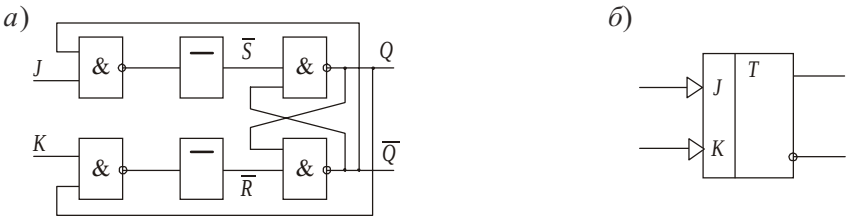
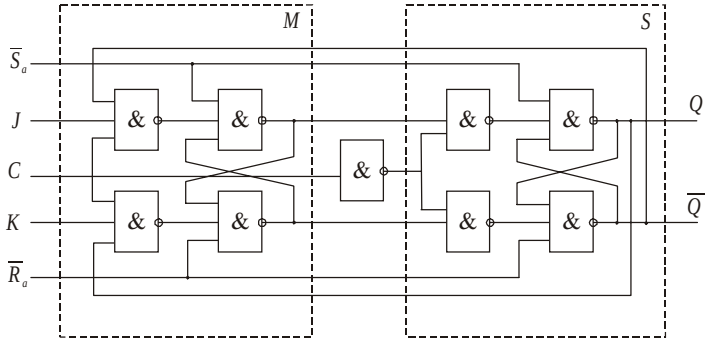


Рис. 8.17

На практике описанный триггер не применяется из-за сложности изготовления и жестких требований к длительности входных сигналов. Так же, как и в случае *T*-триггера, проблема устойчивости эффективно решается в триггерах с двухступенчатым управлением. Функциональная схема двухступенчатого *JK*-триггера показана на рис. 8.18, *а*, его условное обозначение – на рис. 8.18, *б*.

а)



б)

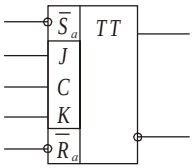


Рис. 8.18

От двухступенчатого *RS*-триггера (рис. 8.3) она отличается наличием обратной связи с выходов Q и $\overline{Q}$ на входы элементов 1 и 2.

Следует заметить, что асинхронная установка любого двухступенчатого триггера в единичное и нулевое состояние (выходы $\overline{R}_a$ и $\overline{S}_a$) производится импульсами логического 0. Состояния остальных входов при асинхронном управлении безразличны. Когда входы $\overline{R}_a$ и $\overline{S}_a$ незадействованы, на них следует поддерживать уровень логической 1.

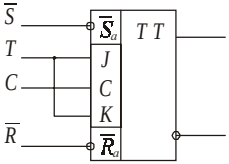


Рис. 8.19

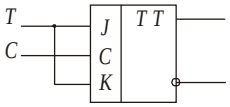


Рис. 8.20

На основе *JK*-триггера путем несложной коммутации входных каналов можно получить схемы триггеров типа *RST* (рис. 8.19) и типа *T* (рис. 8.20).

9. МИНИМИЗАЦИЯ ПОЛНОСТЬЮ ОПРЕДЕЛЕННЫХ АВТОМАТОВ

Метод минимизации полностью определенных абстрактных автоматов изложен в литературе. Основная идея этого метода состоит в разбиении всех состояний исходного абстрактного автомата на попарно не пересекающиеся классы эквивалентных состояний и замене каждого класса эквивалентности одним состоянием. Таким образом, получающийся в результате минимальный автомат имеет столько же состояний, на сколько классов эквивалентности разбиваются состояния исходного автомата.

Два состояния автомата a_m и a_s, называются эквивалентными ($a_m \equiv a_s$), если $\lambda(a_m, \xi) = \lambda(a_s, \xi)$ для всевозможных входных слов ξ. Если a_m и a_s не эквивалентны, они различимы. Более слабой эквивалентностью является k-эквивалентность. Состояния a_m и a_s k-эквивалентны ($a_m \overset{k}{\equiv} a_s$) если $\lambda(a_m, \xi_k) = \lambda(a_s\, \xi_k)$ для всевозможных входных слов ξ_k длины k. Если состояния k-эквивалентны, они k-различимы.

Введенные отношения эквивалентности и k-эквивалентности рефлексивны, симметричны и транзитивны, следовательно, они являются отношениями эквивалентности, а потому могут быть использованы для разбиения множества A состояний автомата на попарно не пересекающиеся классы (классы эквивалентности). Соответствующие разбиения на классы эквивалентных и k-эквивалентных состояний будем обозначать Π и Π_k. Разбиение Π позволяет определить избыточные элементы в множестве состояний A. Пусть, например, a_m и a_s эквивалентны. Это значит, что с точки зрения реакций автомата на всевозможные входные слова неважно, находится автомат в состоянии a_m или a_s, и одно из них, например a_s, может быть удалено из множества A. Если каждый класс эквивалентности содержит только одно состояние, множество A несократимо. Если же один или несколько классов содержат более одного элемента, все элементы, кроме одного в каждом классе могут быть исключены из множества A , в результате чего получается автомат с минимальными числом состояний.

Алгоритм минимизации числа состояний автомата $S = \{a_1, A, Z, W, \delta, \lambda\}$ состоит из следующих шагов.

1. Находятся последовательные разбиения Π_1, Π_2, ..., Π_k, Π_{k+1} множества A на классы одно-, двух-, ..., k, k+1-эквивалентных состояний, до

тех пор пока на каком-то k+1-м шаге не окажется, что $П_{k+1} = П_k$. Нетрудно показать, что тогда разбиение $P_k = P$, т. е. что k-эквивалентные состояния являются в этом случае эквивалентными и число шагов k, при котором $П_k = П$, не превышает M–1, где M – число элементов в множестве A.

2. В каждом классе эквивалентности разбиения $П$ выбираются по одному элементу, которые образуют множество A' состояний минимального автомата S'={a_1', A', Z', W', δ', λ'}, эквивалентного автомату S.

3. Функции переходов δ' и выходов λ' автомата определяются на множестве A'×Z. Для этого в таблице переходов и выходов вычеркиваются столбцы, соответствующие не вошедшим во множество A' состояниям, а в оставшихся столбцах таблицы переходов все состояния заменяются на эквивалентные из множества A'.

4. В качестве a_1' выбирается одно из состояний, эквивалентное состоянию a_1. В частности, удобно за a_1' принимать само состояние a_1.

В качестве примера рассмотрим минимизацию автомата Мили S, заданного таблицами переходов и выходов (табл. 9.1 и 9.2). Непосредственно по таблице выходов получим разбиение $П_1$ на классы одноэквивалентных состояний, объединяя в эквивалентные классы одинаковые столбцы: $П_1$={B_1,B_2}, B_1={a_1, a_2, a_5, a_7, a_8}, B_2={a_3, a_4, a_6, a_9, a_{10}, a_{11}, a_{12}}. Действительно, два состояния 1-эквивалентны, если их реакция на всевозможные входные слова длины 1 совпадают, т. е. соответствующие этим состояниям столбцы в таблице выходов должны быть одинаковы.

Таблица 9.1

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
Z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7	a_3	a_{10}	a_7	a_1	a_5	a_2
Z_2	a_5	a_8	a_6	a_{11}	a_9	a_{11}	a_6	a_4	a_6	a_8	a_9	a_8

Таблица 9.2

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
Z_1	W_1	W_1	W_2	W_2	W_1	W_2	W_1	W_1	W_2	W_2	W_2	W_2
Z_2	W_2	W_2	W_1	W_1	W_2	W_1	W_2	W_2	W_1	W_1	W_1	W_1

Строим таблицу $П_1$ (табл. 9.3), заменяя состояния в табл. 1 соответствующими классами 1-эквивалентности. Очевидно, что 1-эквивалент-

ные состояния a_m, a_s будут 2-эквивалентными, если они переводятся любым входным сигналом также в 1-эквивалентные.

Таблица 9.3

	B_1					B_2						
	a_1	a_2	a_5	a_7	a_8	a_3	a_4	a_6	a_9	a_{10}	a_{11}	a_{12}
Z_1	B_2	B_2	B_2	B_2	B_2	B_1	B_1	B_1	B_1	B_1	B_1	B_1
Z_2	B_1	B_1	B_2	B_2	B_2	B_2	B_2	B_2	B_2	B_1	B_2	B_1

По табл. 9.3 получаем разбиение Π_2 на классы 2-эквивалентных состояний (табл. 9.4): $\Pi_2=\{C_1, C_2, C_3, C_4\}$, $C_1=\{a_1, a_2\}$, $C_2=\{a_5, a_7, a_8\}$, $C_3=\{a_3, a_4, a_6, a_9, a_{11}\}$, $C_4=\{a_{10}, a_{12}\}$.

Аналогично построим $\Pi_3=\{D_1, D_2, D_3, D_4, D_5\}$, $D_1=\{a_1, a_2\}$, $D_2=\{a_5, a_7\}$, $D_3=\{a_8\}$, $D_4=\{a_3, a_4, a_6, a_9, a_{11}\}$, $D_5=\{a_{10}, a_{12}\}$ (табл. 9.5) и, наконец, $\Pi_4=\{E_1, E_2, E_3, E_4, E_5\}$, которое совпадает с Π_3. Процедура разбиения завершена. Разбиение Π_3 есть разбиение множества состояний автомата Мили S на классы эквивалентных между собой состояний.

Таблица 9.4

	C_1		C_2			C_3					C_4	
	a_1	a_2	a_5	a_7	a_8	a_3	a_4	a_6	a_9	a_{11}	a_{10}	a_{12}
Z_1	C_4	C_4	C_3	C_3	C_4	C_2	C_2	C_2	C_2	C_2	C_1	C_1
Z_2	C_2	C_2	C_3	C_3	C_3	C_3	C_3	C_3	C_3	C_3	C_2	C_2

Таблица 9.5

	D_1		D_2		D_3	D_4					D_5	
	a_1	a_2	a_5	a_7	a_8	a_3	a_4	a_6	a_9	a_{11}	a_{10}	a_{12}
Z_1	D_5	D_5	D_4	D_4	D_5	D_2	D_2	D_2	D_2	D_2	D_1	D_1
Z_2	D_2	D_2	D_4	D_4	D_4	D_4	D_4	D_4	D_4	D_4	D_3	D_3

Выберем произвольно из каждого класса D_1, D_2, D_3, D_4, D_5 по одному состоянию. Пусть, например, $A'=\{a_1, a_5, a_8, a_3, a_{10}\}$. Удаляя из первоначальных таблиц переходов (табл. 9.1) и выходов (табл. 9.2) “лишние” состояния a_2, a_7, a_4, a_6, a_9, a_{11}, a_{12}, определяем минимальный автомат Мили S_1 (таблица переходов 9.6 и таблица выходов 9.7), эквивалентный автомату S.

Таблица 9.6

	a_1	a_5	a_8	a_3	a_{10}
Z_1	a_{10}	a_3	a_{10}	a_5	a_1
Z_2	a_5	a_3	a_3	a_3	a_8

Таблица 9.7

	a_1	a_5	a_8	a_3	a_{10}
Z_1	W_1	W_1	W_1	W_2	W_2
Z_2	W_2	W_2	W_2	W_1	W_1

При минимизации автоматов Мура вводится понятие 0-эквивалентности состояний и разбиения множества состояний на 0-классы: 0-эквивалентными называются любые одинаково отмеченные состояния автоматов Мура. Если два 0-эквивалентных состояния любым входным сигналом переводятся в два 0-эквивалентных состояния, то они называются 1-эквивалентными. Все дальнейшие классы эквивалентности состояний для автоматов Мура определяются аналогично приведенному выше для автоматов Мили. В результате применения алгоритма минимизации к автомату Мура S_3 (табл.9.8), имеющему 12 состояний, получим автомат S_4 с 4 состояниями (табл. 9.9). Опуская промежуточные таблицы, приведем лишь последовательность разбиений

$$П_0=\{B_1, B_2, B_3\}.$$

Таблица 9.8

	W_1	W_1	W_3	W_3	W_3	W_2	W_3	W_1	W_2	W_2	W_2	W_2
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
Z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7	a_3	a_{10}	a_7	a_1	a_5	a_2
Z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}	a_6	a_4	a_6	a_8	a_9	a_8

$B_1=\{a_1, a_2, a_8\}$; $B_2=\{a_6, a_9, a_{10}, a_{11}, a_{12}\}$; $B_3=\{a_3, a_4, a_5, a_7\}$;
$П_1=\{C_1, C_2, C_3, C_4\}$; $C_1=\{a_1, a_2, a_8\}$; $C_2=\{a_6, a_9, a_{11}\}$; $C_3=\{a_{10}, a_{12}\}$;
$C_4=\{a_3, a_4, a_5, a_7\}$;
$П_2=\{D_1, D_2, D_3, D_4\}$; $П_2=П_1$; $D_1=C_1$; $D_2=C_2$; $D_3=C_3$; $D_4=C_4$.

Таблица 9.9

	W_1	W_2	W_2	W_3
	a_1	a_6	a_{10}	a_3
Z_1	a_{10}	a_3	a_1	a_3
Z_2	a_3	a_6	a_1	a_6

Если заданный автомат частичный, то для того, чтобы воспользоваться рассмотренным методом минимизации, необходимо его доопределить так, чтобы можно было найти максимальное число эквивалентных состояний.

10. МЕТОДЫ КОДИРОВАНИЯ СОСТОЯНИЙ АБСТРАКТНЫХ АВТОМАТОВ

Процесс кодирования состояний абстрактных автоматов является первым этапом канонического метода структурного синтеза автоматов. Кодирование заключается в установлении взаимно однозначного соответствия между множеством $A = \{a_1, ..., a_m\}$ состояний автомата и множеством R-компонентных векторов $\{k_1, ..., k_m\}$, $k_m=\{\lambda_{m1}, ..., \lambda_{mR}\}$, где λ_{m1} – состояние r-го элемента памяти (триггера). Обычно кодирование производится с помощью символов двоичного структурного алфавита ($\lambda_m \in \{1,0\}$), поскольку при этом решается задача определения необходимого числа триггеров, имеющих два устойчивых состояния. Зависимость числа триггеров от количества состояний заданного абстрактного автомата определяется формулой [2]:

$$R \geq]\log_2 M[,$$

где $]b[$ означает ближайшее целое число, большее b или равное ему, если b – целое.

Кодирование состояний автомата можно осуществлять различными способами. Это может быть *произвольное кодирование*, когда каждому состоянию ставится в соответствие случайный набор двоичных символов, количество которых равно R. Синтезированный на основе такого кодирования структурный автомат не будет оптимальным, так как, во-первых, его комбинационная схема может обладать повышенной сложностью и, во-вторых, при отсутствии синхронизации и двойной памяти в процессе функционирования этого автомата могут появиться состязания [4].

Явление состязаний возникает вследствие того, что элементы памяти имеют различные, хотя и достаточно близкие, времена срабатывания. Кроме того, различны также задержки сигналов возбуждения, поступающих на входные каналы элементарных автоматов по логическими цепям неодинаковой длины. Если при переходе автомата из одного состояния в другое должны изменить свои состояния сразу несколько триггеров (что характерно для произвольного кодирования), то между ними начинаются состязания. Тот триггер, который выигрывает эти состязания, т. е. изменит свое состояние раньше, чем другие элементы памяти, может через цепь обратной связи изменить сигналы на входах некоторых триггеров до того, как другие участвующие в состязаниях изменят свое состояние. Это может привести автомат в состояние, не

предусмотренное графом. Тогда возникшие состязания называются критическими состязаниями или гонками.

Гонки могут быть устранены различными способами, в том числе и с помощью *противогоночного кодирования*.

10.1. Противогоночное кодирование методом развязывания пар переходов

В литературе [4] предлагается метод противогоночного кодирования, основная идея которого сводится к следующему.

Пусть (a, β) и (γ, δ) – две пары двоичных кодов произвольной длины, например

$$\alpha - 1|0|1\ 1 \quad \gamma - 0|1|1\ 1$$
$$\beta - 0|0|1\ 1 \quad \delta - 0|1|0\ 0$$

Если некоторый *r*-й разряд кода принимает одно значение на паре (α, β) и противоположное – на паре (γ, δ), то такие пары кодов называются развязанными.

Доказана следующая теорема [3]: в автомате, состояния которого закодированы двоичными кодами конечной длины, гонки отсутствуют тогда и только тогда, когда для любых двух переходов (a_m, a_s) и (a_k, a_l), $a_s \neq a_l$, происходящих под действием одного и того же входного сигнала, соответствующие пары кодов развязаны. (Если автомат синхронный, то развязывать нужно пары переходов, для которых $(a_m, a_s) \cap (a_k, a_l) = \varnothing$). В этой же работе приведен основанный на этой теореме алгоритм противогоночного кодирования состояний конечных автоматов, основная идея которого достаточно проста: последовательно просматривая все пары переходов, для которых имеется хотя бы один общий входной сигнал, осуществляющий эти переходы, следует присвоить разрядам кодов такие значения, чтобы существующие пары кодов состояний были развязаны.

Алгоритм противогоночного кодирования заключается в последовательном развязывании подлежащих развязыванию пар переходов. На промежуточных этапах алгоритма состояниям автомата будут соответствовать коды, значения некоторых разрядов которых могут быть не определены. Такие коды будем называть неполными. В дальнейшем неопределенные разряды кодов отмечаются черточкой. Пусть (a_m, a_s), (a_k, a_l) – пара переходов автомата *S*, а α, β, γ, δ – соответственно четверка кодов (быть может, неполных) состояний a_m, a_s, a_k, a_l длины *i*.

Операция развязывания пары переходов (a_m, a_s), (a_k, a_l) сводится к нескольким этапам.

1. Положить $i = 0$. Перейти к п.2.

2. Если $i = 0$, то переход к п.8, иначе переход к п.3.

3. Если при некотором r $(1 \le r \le i)$ значения r-го разряда четверки α, β, γ, δ образует набор 0011 или набор 1100, то переход к п.9, иначе к п.4.

4. Если при некотором r $(1 \le r \le i)$ значения r-го разряда четверки α, β, γ, δ образует один из наборов

– 0 1 1	– – 1 1	– – – 1
0 – 1 1	0 – – 1	0 – – –
0 0 – 1	0 0 – –	– 0 – –
0 0 1 –	– 0 – 1	– – 1 –
– 0 1 –	0 – 1 –	– – – –,

то переход к п.5, иначе к п.6.

5. Доопределить неопределенные значения r-го разряда четверки α, β, γ, δ так, чтобы его значения образовывали набор 0011. Переход к п.9.

6. Если при некотором r $(1 \le r \le i)$ значения r-го разряда четверки α, β, γ, δ образует один из наборов

– 1 0 0	– – 0 0	– – – 0
1 – 0 0	1 – – 0	1 – – –
1 1 – 0	1 1 – –	– 1 – –
1 1 0 –	– 1 – 0	– – 0 –
– 1 0 –	1 – 0 –	– – – –,

то переход к п.7, иначе переход к п.8.

7. Доопределить неопределенные значения r-го разряда четверки α, β, γ, δ так, чтобы значения этого разряда образовывали набор 1100. Переход к п.9.

8. Дополнить коды состояний автомата одним неопределенным разрядом. Увеличить r на единицу. Переход к п.4.

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

Длина кода, получаемая в результате применения изложенного алгоритма, в большинстве случаев оказывается неминимальной, так как при введении нового разряда кода могут развязываться пары переходов, которые уже были развязаны ранее. В связи с этим желательно минимизировать длину получаемых кодов состояний, что делается следующим образом. Исключаем один из разрядов кодов, в результате чего

некоторые пары переходов могут оказаться связанными, и применяем алгоритм развязывания пар переходов. После этого исключаем еще один разряд, вновь применяем алгоритм противогоночного кодирования и т.д., до тех пор пока длина кода не перестанет уменьшаться. Если в результате работы алгоритма значения не всех разрядов будут определены, то их можно доопределить произвольно.

Проиллюстрируем алгоритм противогоночного кодирования на примере автомата, функция переходов которого задана табл. 10.1.

Таблица 10.1

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Z_1	a_2	a_2	a_4	a_4	a_6	a_6	–
Z_2	a_1	a_3	a_3	a_1	a_3	–	–
Z_3	–	a_5	a_7	–	a_5	–	a_7

Очевидно, что пары должны быть развязаны в каждом из массивов переходов M_1, M_2, M_3, происходящих под действием сигналов Z_1, Z_2, Z_3:

M_1	M_2	M_3
(a_1, a_2)	(a_1, a_1)	(a_2, a_5)
(a_2, a_2)	(a_2, a_3)	(a_3, a_7)
(a_3, a_4)	(a_3, a_3)	(a_5, a_5)
(a_4, a_4)	(a_4, a_1)	(a_7, a_7)
(a_5, a_6)	(a_5, a_3)	
(a_6, a_6)		

Развязывание пар переходов в M_1 начнем с первого перехода (a_1, a_2). Согласно сформулированной выше теореме пару (a_1, a_2) и (a_2, a_2) развязывать не надо из-за совпадения состояний перехода. Первая пара переходов, которая подлежит развязыванию, есть (a_1, a_2), (a_3, a_4). Вводим переменную t_1 и образуем по этой переменной четверку (0011) для состояний a_1, a_2, a_3, a_4. Рассматриваемая пара переходов развязана (табл.10.2).

Паре переходов (a_1, a_2), (a_4, a_4) соответствует четверка (0011) (табл.10.2), т. е. эта пара тоже развязана.

Пара (a_1, a_2),(a_5, a_6) образует четверку (00 – –). Для развязывания этой пары доопределим эту четверку до (0011), для чего состояниям a_5, a_6 ставим в соответствие t_1= 1 (табл. 10.3).

Таблица 10.2

	τ_1
a_1	0
a_2	0
a_3	1
a_4	1
a_5	–
a_6	–
a_7	–

Таблица 10.3

	τ_1
a_1	0
a_2	0
a_3	1
a_4	1
a_5	1
a_6	1
a_7	–

Таблица 10.4

	τ_1	τ_2
a_1	0	–
a_2	0	–
a_3	1	0
a_4	1	0
a_5	1	1
a_6	1	1
a_7	–	–

Из табл.10.3 видно, что пара (a_1, a_2), (a_6, a_6) развязана (четверка (0011)). Точно так же развязаны пары, образованные переходом (a_2, a_2) и всеми последующими переходами в M_1. Обратимся к паре (a_3, a_4), (a_5, a_6). Из табл. 10.3 получаем соответствующую четверку (1111) – пара не развязана. Вводим переменную t_2 и полагаем для a_3 и a_4 значение τ_2=0, а для a_5 и a_6 τ_2= 1 (табл. 10.4).

После чего остальные переходы в M_1 тоже развязаны. Аналогично для M_2 и M_3 получим табл.10.5 и 10.6.

Таблица 10.5

	τ_1	τ_2	τ_3
a_1	0	1	1
a_2	0	0	0
a_3	1	0	0
a_4	1	0	1
a_5	1	1	0
a_6	1	1	–
a_7	–	–	–

Таблица 10.6

	τ_1	τ_2	τ_3	τ_4
a_1	1	1	1	–
a_2	0	0	0	0
a_3	1	0	0	1
a_4	1	0	1	–
a_5	1	1	0	0
a_6	1	1	–	–
a_7	–	0	–	1

Переходим к минимизации. Исключаем переменную τ_1 (табл. 10.7) и повторяем процесс развязывания пар переходов.

Оказывается, что пара (a_1, a_2), (a_5, a_6) не развязана, в связи с чем добавляем переменную τ_5 и развязываем эту пару (табл. 10.8).

Все остальные пары развязаны. Далее исключаем переменную τ_2 и получаем табл.10.9 с тремя переменными τ_3,τ_4, τ_5, в которой после проверки оказываются развязанными все пары.

Таблица 10.7

	τ_2	τ_3	τ_4
a_1	1	1	–
a_2	0	0	0
a_3	0	0	1
a_4	0	1	–
a_5	1	0	0
a_6	1	–	–
a_7	0	–	1

Таблица 10.8

	τ_2	τ_3	τ_4	τ_5
a_1	1	1	0	0
a_2	0	0	0	0
a_3	0	0	1	–
a_4	0	1	1	–
a_5	1	0	0	1
a_6	1	–	–	1
a_7	0	–	1	–

Таблица 10.9

	τ_3	τ_4	τ_5
a_1	1	0	0
a_2	0	0	0
a_3	0	1	–
a_4	1	1	–
a_5	0	0	1
a_6	–	0	1
a_7	–	1	–

Таблица 10.10

	τ_3	τ_4	τ_5
a_1	1	0	0
a_2	0	0	0
a_3	0	1	0
a_4	1	1	0
a_5	0	0	1
a_6	1	0	1
a_7	0	1	1

Дальнейшая минимизация невозможна, так как для кодирования семи состояний нужно не менее трех переменных. После доопределения прочерков в табл. 10.1 получаем табл. 10.10 противогоночного кодирования состояний исходного автомата.

10.2. Противогоночное соседнее кодирование

Второй способ кодирования, позволяющий избавиться от гонок, – это кодирование соседних состояний автомата соседними кодами (*соседнее кодирование*). Соседние состояния – это состояния, связанные дугой на графе автомата, а соседние коды – это двоичные наборы, отличающиеся только одним разрядом. Расстояние по Хэммингу у таких кодов равно 1. При соседнем кодировании при переходе автомата из одного состояния в другое меняется состояние только у одного элемента памяти и состязания становятся невозможными. Соседнее кодирование не всегда оказывается возможным [4] и в этом случае приходит-

ся на графе между соседними состояниями автомата вставлять дополнительные, так называемые неустойчивые состояния. Неустойчивое состояние автомата (состояние a_k, на рис. 10.1) отличается тем, что под действием некоторого входного сигнала Z_k, по длительности превышающего время перехода в это состояние a_k, автомат может его "проскочить", перейдя сразу в следующее состояние a_s.

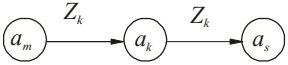


Рис. 10.1

В процессе добавления неустойчивых состояний необходимо следить за тем, чтобы значение выходного сигнала при этом не менялось. Для того чтобы удобнее было находить коды соседних состояний, целесообразно воспользоваться диаграммой Вейча. Число клеток необходимой диаграммы определяется как 2^k, где k – число соседей у того состояния автомата, которое имеет их больше всех остальных состояний. Рассмотрим пример кодирования соседними кодами состояний фрагмента графа некоторого автомата, приведенного на рис. 10.2.

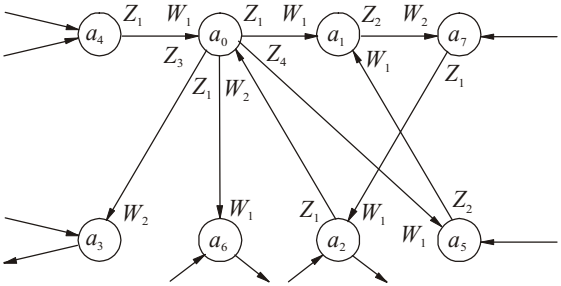


Рис. 10.2

Состоянием с максимальным числом соседей (k = 6) является состояние a_0. Следовательно, для кодирования удобно воспользоваться диаграммой Вейча размером 8×8=64=42 клеток (рис. 10.3). Поместим состояние a_0 в произвольную клетку диаграммы, например соответствующую коду 110110 ($Q_1Q_2\overline{Q}_3Q_4Q_5\overline{Q}_6$). Эта клетка имеет 6 соседних клеток, куда целесообразно помещать все состояния, соседние a_0. Тогда получим следующие коды состояний:

$k(a_0)$ – 110110, $k(a_1)$ – 110100, $k(a_2)$ – 110010, $k(a_3)$ – 100110

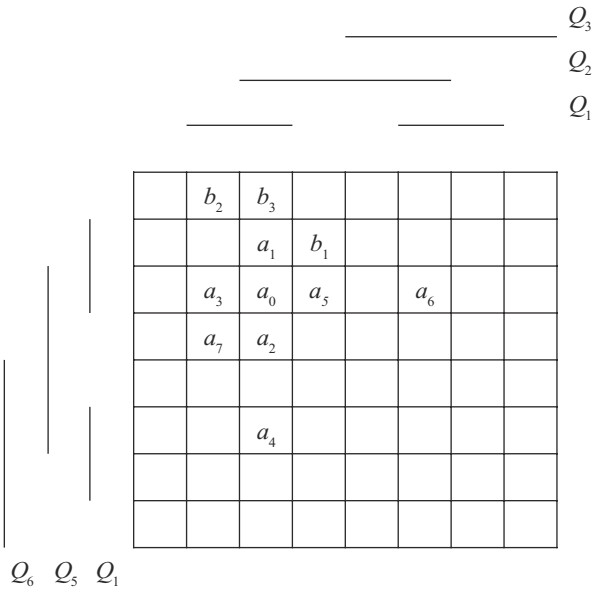


Рис. 10.3

Очевидно, коды состояний a_1, a_2, a_3 отличаются от кода состояния a_0 только одним разрядом. Поскольку состояние a_5 является одновременно соседним и для a_0, и для a_1, необходимо поместить его в такую клетку, которая имела бы минимальные расстояния от обоих этих состояний. Такими клетками являются клетки с кодами 010100 и 010110. Выберем одну из них (например, 010110). Тогда, чтобы обеспечить соседнее кодирование, придется ввести дополнительное неустойчивое состояние b_1, используя для него второй из этих кодов, а именно код 010100.

После этого оставшиеся состояния a_4 и a_6 можно поместить в клетки с кодами 110111 и 111110 соответственно. Состояние a_7, соседнее состояниям a_3 и a_2, удачно помещается в клетку 100010, но при этом не получается требуемых соседних кодов у состояний a_7 и a_1. Поэтому на переходе $a_7 \rightarrow a_1$ необходимо ввести дополнительные состояния (меньше двух при выбранном кодировании не получается) b_2 и b_3, расположенные соответственно в соседних клетках (a_7 с b_2, b_2 с b_3, b_3 с a_1).

В результате кодирования число состояний графа увеличилось на 4 (рис. 10.4), что, естественно, уменьшает быстродействие автомата.

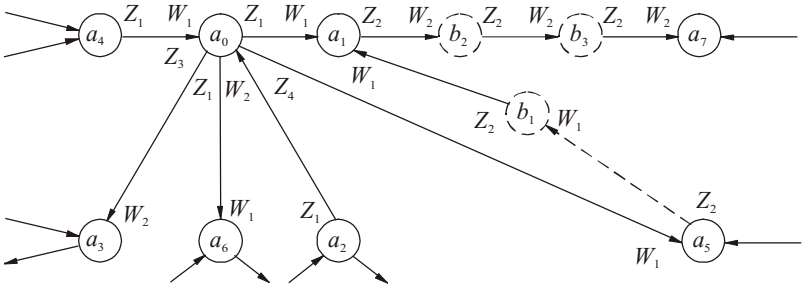


Рис. 10.4

Коды состояний, полученных в результате соседнего кодирования, приведены в табл. 10.11.

Таблица 10.11

Состояние абстрактного автомата	Двоичный код	Соответствующие состояния элементов памяти
a_0	110110	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
a_1	110100	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
a_2	110010	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
a_3	100110	$Q_1\ Q2\ Q_3\ Q_4\ Q_5\ Q_6$
a_4	110111	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
a_5	010110	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
a_6	111110	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
a_7	100010	$Q_1\ \ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
b_1	010100	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
b_2	100000	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$
b_3	110000	$Q_1\ Q_2\ Q_3\ Q_4\ Q_5\ Q_6$

Следует отметить, что в графе переходов можно найти такие пары соседних состояний ($a_m \xrightarrow{z_k} a_s$), которые не требуют соседнего кодирования. В этом случае коды, появляющиеся в результате состязаний (ложные коды), могут быть использованы для кодирования состязаний автомата, удовлетворяющих одному из следующих условий:

1) кодируемое ложным кодом состояние (a_i) должно быть устойчивым по отношению к входному сигналу z_k, (рис. 10.5); поскольку под

действием z_k автомат из a_i никуда перейти не может, состязания, если они и возникнут, являются некритическими;

2) из состояния a_i отсутствует переход по сигналу z_k;

3) из кодируемого ложным кодом состояния a_i под действием сигнала z_k автомат переходит в нужное состояние a_s (рис.10.6).

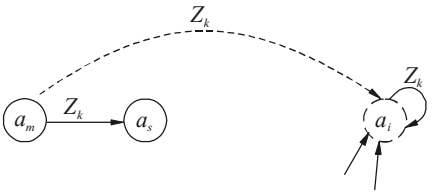


Рис. 10.5

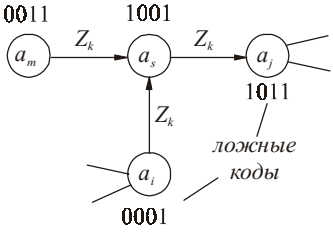


Рис. 10.6

Кодирование соседних состояний кодами с расстоянием по Хэммингу, большим 1, можно использовать также в тех случаях, когда ложные коды, возникающие при состязаниях, не используются в процессе кодирования.

10.3. Кодирование состояний автомата, близкое к соседнему

Анализ канонического метода структурного синтеза автоматов показывает, что различные варианты кодирования состояний автомата приводят к различным выражениям функций возбуждения памяти и функций выходов. Эти выражения обладают различными рангами канонических форм записи, в результате чего реализованные по ним комбинационные схемы обладают различной сложностью, которая в итоге зависит от способа кодирования.

Рассмотрим эвристический алгоритм кодирования состояний [4] и минимизирующий суммарное число изменений элементов памяти на всех переходах автомата.

Введем весовую функцию $W = \sum t_{ms}$, где $t_{ms} = |K_m – K_s|^2$ – расстояние между кодами состояний a_m и a_s, равное числу элементов памяти, изменяющих свое состояние на переходе (a_m, a_s); суммирование производится по всем переходам автомата. Введенная функция W может служить одной из оценок сложности комбинационной схемы автомата S, при этом упрощение комбинационной схемы будет тем больше, чем меньше W.

Алгоритм состоит из нескольких шагов.

1. Построим матрицу

$$\mathbf{M} = \begin{vmatrix} \alpha_1 & & \beta_1 \\ . & . & . \\ \alpha_r & & \beta_r \\ . & . & . \\ \alpha_R & & \beta_R \end{vmatrix},$$

состоящую из всех различных пар номеров (a_r, b_r), таких, что в автомате S есть переход из $a_{\alpha r}$ в $a_{\beta r}$.

2. Переставим строки в матрице так, чтобы выполнялось условие

$$\{\alpha_r, \beta_r\} \cap \{\alpha_1, \beta_1, ..., \alpha_{r-1}, \beta_{r-1}\} \neq \varnothing, \; r=2, ..., R. \qquad (10.1)$$

Условие (10.1) означает, что хотя бы один из элементов r-й строки содержится в какой-нибудь из предыдущих строк. Имеются в виду только связные автоматы S, для которых такая перестановка всегда возможна.

3. Закодируем состояния из первой строки матрицы **M** следующим образом:

$$K_{\alpha 1} = (00 \; ... \; 00); \; K_{\beta 1} = (00 \; ... \; 01).$$

4. Вычеркнем из матрицы M первую строчку, соответствующую закодированным состояниям $a_{\alpha 1}$, и $a_{\beta 1}$. Получим матрицу **M**'.

5. В силу условия (1) в начальной строке матрицы **M** закодирован один элемент. Выберем из первой строчки матрицы **M**' незакодированный элемент и обозначим его через γ.

6. Построим матрицу M_γ, выбрав из M' строчки, содержащие γ. Пусть $B_\gamma = \{\gamma_1, ..., \gamma_f, ..., \gamma_F\}$ – множество элементов из матрицы M_γ, которые уже закодированы. Их коды обозначим $K_{\gamma 1}, ..., K_{\gamma f}, ..., K_{\gamma F}$ соответственно.

7. Для каждого $K_{\gamma f}$ ($f = 1, ..., F$) найдем $C^1_{\gamma_f}$ – множество кодов, соседних с K_{gf} и еще не занятых для кодирования состояний автомата. Построим множество $D^1_\gamma = \bigcup_{f=1}^{F} C^1_{\gamma_f}$. Если $D^1_\gamma = \varnothing$, то строим новое множество $D^2_\gamma = \bigcup_{f=1}^{F} C^2_{\gamma_f}$, где $C^2_{\gamma_f}$ – множество кодов, у которых кодовое

расстояние с кодом $K_{\gamma f}$ равно двум. Если $D_\gamma^2 = \varnothing$, строим аналогично $D_\gamma^3, ..., D_\gamma^n$, до тех пор пока не найдется $D_\gamma^n \neq \varnothing$ (n = 1, 2, 3, ...). Пусть $D_\gamma^n = \{ K_{\delta 1}, ..., K_{\delta g}, ..., K_{\delta G}\}$.

8. Для каждого $K_{\delta g}$ находим $W_{gf} = | K_{\delta g} - K_{gf} |2$ – кодовые расстояния между $K_{\delta g}$ и всеми использованными кодами K_{gf} (f = 1, ..., F). Если в автомате имеется переход из $a_{\gamma f}$ в a_γ и из a_γ в $a_{\gamma f}$ то W_{gf}, входит дважды в W_g (см. ниже примеры переходов (a_4, a_5) и (a_5, a_4)).

9. Находим $W_g = \sum_{f=1}^{F} W_{gf}$, g = 1, ..., G.

10. Из D_γ^n выбираем K_γ, у которого $W_g = \min W_g$. Элемент γ (состояние a_γ) кодируем кодом K_γ.

11. Из матрицы **M**’ вычеркнем строчки, в которых оба элемента закодированы, в результате чего получим новую матрицу, которую такие обозначим через **M**’. Если в матрице **M**’ не осталось ни одной строчки, переходим к п.12, иначе к п. 5.

12. Вычисляем функцию $W = \Sigma t_{ms}$, где $t_{ms} = |K_m - K_s|^2$.

13. Конец.

Оценкой качества кодирования по рассмотренному алгоритму может служить число $k = W/p$, где p – число переходов в автомате. Очевидно, что $k \geq 1$, причем чем меньше значение k, тем ближе кодирование к соседнему , при котором k = 1.

Без подробных объяснений приведем пример кодирования состояний автомата, граф которого изображен на рис. 10.7.

$$\mathbf{M} = \begin{vmatrix} 1 & 2 \\ 2 & 4 \\ 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix} \qquad K_1 = 000, \quad K_2 = 001.$$

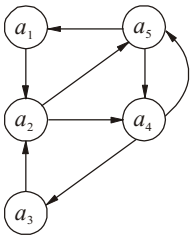
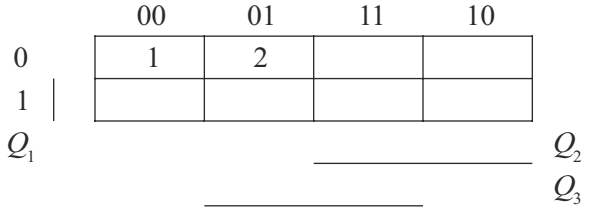


Рис. 10.7

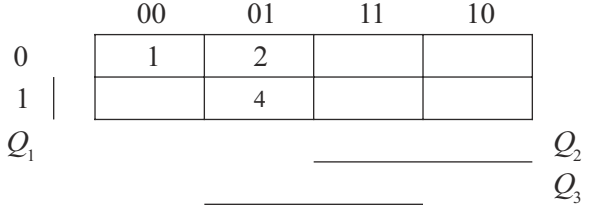
Кодирование будем иллюстрировать диаграммой Вейча.



$$\mathbf{M}' = \begin{vmatrix} 2 & 4 \\ 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix} \quad \gamma = 4; \qquad \mathbf{M}_4 = \begin{vmatrix} 2 & 4 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \end{vmatrix} \quad B_4 = \{2\}.$$

$C_2^1 = \{101, 011\}$; $D_4^1 = C_2^1 = \{101, 011\}$. $W_{101} = |101–001|^2 = 1$; $W_{011} = |011\text{-}001|^2 = 1$.

Выбираем $K_4 = 101$.



$$\mathbf{M}' = \begin{vmatrix} 2 & 5 \\ 3 & 2 \\ 4 & 3 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix} \quad \gamma = 5; \qquad \mathbf{M}_5 = \begin{vmatrix} 2 & 5 \\ 4 & 5 \\ 5 & 4 \\ 5 & 1 \end{vmatrix} \quad B_5 = \{2, 4, 1\}.$$

$C_2^1=\{011\}$; $C_4^1=\{100,111\}$; $C_1^1=\{100,010\}$; $D_5^1=C_2^1\cup C_4^1\cup C_1^1=\{011,100,111,010\}$.

$W_{011}=|011–001|^2+|011–101|^2+|011–101|^2+|011–000|^2=1+2+2+2=7$;

$W_{100}=|100–001|^2+|100–101|^2+|100–101|^2+|100–000|^2=2+1+1+1=5$;

$W_{111}=|111–001|^2+|111–101|^2+|111–101|^2+|111–000|^2=2+1+1+3=6$;

$W_{010}=|010–001|^2+|010–101|^2+|010–101|^2+|010–000|^2=2+3+3+1=9$.

$W_{100}=\min\{W_{001}, W_{100}, W_{111}, W_{010}\}$. Следовательно выбираем $K_5=100$.

	00	01	11	10	
0	1	2			
1	5	4			
Q_1					Q_2
					Q_3

$$\mathbf{M}'=\begin{vmatrix}3 & 2\\ 4 & 3\end{vmatrix} \quad \gamma=3; \quad \mathbf{M}_3=\begin{vmatrix}3 & 2\\ 4 & 3\end{vmatrix} \quad B_3=\{2,4\}.$$

$C_2^1=\{011\}$; $C_4^1=\{111\}$; $D_3^1=C_2^1 \text{ И } C_4^1=\{011,111\}$.

$W_{011}=|011–001|^2+|011–101|^2=1+2=3$;

$W_{111}=|111–001|^2+|111–101|^2=2+1=3$.

$W_{011}=W_{111}$. Следовательно выбираем K_3= 011.

	00	01	11	10	
0	1	2	3		
1	5	4			
Q_1					Q_2
					Q_3

$k = W/p$ = 10:8 = 1,25.

10.4. Соседнее кодирование логически смежных состояний

Существует другой метод кодирования состояний, позволяющий упростить полученную в результате структурного синтеза схему [4]. Суть этого метода заключается в использовании двух следующих правил кодирования.

Правило 1. Те состояния, из которых возможны переходы в одни и те же состояния хотя бы для одного значения входного сигнала, являются логически смежными и должны быть закодированы соседними кодами.

Правило 2. Логически смежными являются состояния, следующие для одного и того же состояния. Их необходимо кодировать соседними кодами.

Если при использовании этих правил невозможно закодировать соседними кодами все логически смежные состояния, то приоритет должен сохраниться за правилом 1.

В таблице переходов состояния, удовлетворяющие правилу 1, должны иметь одинаковые состояния перехода в какой-либо строке. Состояния, удовлетворяющие правилу 2, находятся в одном столбце таблицы переходов.

Рассмотрим таблицу переходов автомата (табл. 10.12).

Таблица 10.12

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
0	a_3	a_2	a_3	a_5	–	a_1	–
1	a_4	a_3	a_5	a_4	a_2	a_5	a_2
α	–	a_1	a_1	a_6	a_6	–	a_1

Перед началом операции кодирования целесообразно сделать все доопределения, если это необходимо. Далее следует выписать группы состояний, у которых имеются одинаковые элементы в какой-либо строке (правило 1).

В нашем примере это:

(a_0, a_2) – оба переходят в a_3 по сигналу “0”;
(a_0, a_3) – оба переходят в a_4, по сигналу “1”;
(a_2, a_5) – оба переходят в a_5 по сигналу “1”;
(a_4, a_6) – оба переходят в a_8 по сигналу “1”;
(a_3, a_4) – оба переходят в a_6 по сигналу “α “;

(a_1, a_2, a_6) – все переходят в a_1 по сигналу "α".

Далее необходимо выписать группы состояний, находящихся в одних и тех же столбцах. В нашем примере это (a_3, a_4), (a_2, a_1), (a_2, a_3, a_1), (a_3, a_5, a_1), (a_5, a_4, a_6), (a_2, a_6), (a_1, a_5).

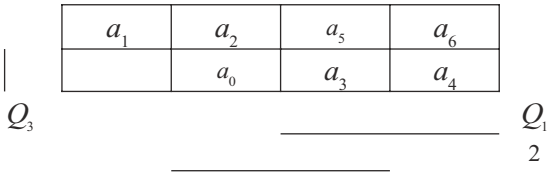
Все состояния, находящиеся в каждой из сформированных групп, должны быть закодированы соседними кодами. Для этого на основе полученных групп следует составить классы состояний, логически смежных с каждым из состояний автомата, причем каждую пару логически смежных состояний целесообразно включать только в один класс.

Сделав это в рассматриваемом примере, получим

K_0	K_1	K_2	K_3	K_4	K_5
$*(a_0, a_3)$	$*(a_1, a_2)$	$*(a_2, a_5)$	$*(a_3, a_4)$	(a_4, a_5)	$*(a_5, a_6)$
$*(a_0, a_2)$	$*(a_1, a_6)$	$*(a_2, a_6)$	(a_3, a_5)	$*(a_4, a_6)$	
	(a_1, a_5)	(a_2, a_3)			
	(a_1, a_3)				

Пары состояний, полученные в соответствии с правилом 1, отмечены знаком *.

Для кодирования логически смежных состояний целесообразно воспользоваться диаграммой Вейча, отдавая приоритет парам состояний, полученным по правилу 1. Проведя эту операцию, получим



Оказалось, что в данном случае не удалось закодировать соседними кодами следующие пары логически смежных состояний:

$$(a_2, a_6), (a_2, a_3), (a_5, a_4), (a_1, a_5), (a_1, a_3).$$

В результате получили следующие коды состояний:

$K(a_1) = 000$ ($\overline{Q}_1\ \overline{Q}_2\ \overline{Q}_3$);

$K(a_2) = 100$ ($Q_1\ \overline{Q}_2\ \overline{Q}_3$);

$K(a_3) = 111$ ($Q_1\ Q_2\ Q_3$);

$K(a_4)$= 011 ($\overline{Q}_1 Q_2 Q_3$);

$K(a_5)$ = 110 ($Q_1 Q_2 \overline{Q}_3$);

$K(a_6)$ = 010 ($\overline{Q}_1 Q_2 \overline{Q}_3$).

Процесс кодирования характеризуется качеством кодирования (k), которое рассчитывается по формуле

$$k = \frac{m}{n},$$

где m – число пар логически смежных состояний, которые удалось закодировать соседними кодами; n – общее количество пар состояний, сформированных в классах K_1, K_2, ..., K_N.

В рассматриваемом примере $k = \frac{9}{13} \approx 0{,}69$.

Хорошим можно считать кодирование, у которого $k \geq 0{,}5$.

Библиографический список

1. *Кузнецов О. П.*, *Адельсон-Вельский Г. М.* Дискретная математика для инженера. М.: Энергоатомиздат, 1988.

2. *Глушков В. М.* Синтез цифровых автоматов. М.: Физматгиз, 1962.

3. *Мацевитый Л. В.*, *Денисенко Е. Л.* О кодировании внутренних состояний некоторых многотактных устройств// Кибернетика. 1966. №1.

4. *Баранов С. И.* Синтез микропрограммных автоматов. Л.: Энергия, 1979.

5. *Козин И. В.*, *Лупал А. М.* Проектирование цифровых автоматов управления и контроля/ ЛИАП. Л., 1985.

6. *Мелехин В. Ф.*, *Дурандин К. П.* Вычислительные машины и системы. СПб.: Высшая школа, 1993.

Оглавление

1. КИБЕРНЕТИКА – НАУКА ОБ УПРАВЛЕНИИ 3
1.1. Создание кибернетики 3
1.2. Предмет и методы исследования кибернетики 4
2. ВВЕДЕНИЕ В ТЕОРИЮ АЛГОРИТМОВ 6
2.1. Определение алгоритма 6
2.2. Предмет теории алгоритмов 7
2.3. Блок-схемы алгоритмов, композиция алгоритмов 9
2.4. Алгоритмические модели 10
3. МАШИНА ТЬЮРИНГА 13
3.1. Структура машины 13
3.2. Детерминированность машины Тьюринга 15
3.3. Работа машины Тьюринга 16
3.4. Конфигурация машины Тьюринга 20
3.5. Тьюрингово вычисление 22
3.6. Тезис Тьюринга 27
4. ВВЕДЕНИЕ В ТЕОРИЮ АВТОМАТОВ 29
4.1. Алфавитные операторы и автоматы 30
4.2. Абстрактные автоматы 32
4.3. Способы задания абстрактных автоматов 34
4.4. Автоматные операторы 37
5. АБСТРАКТНЫЙ СИНТЕЗ АВТОМАТОВ 40
5.1. Построение функций переходов и выходов по алфавитному оператору 40
5.2. Постановка задачи о синтезе автоматов 44
5.3. Классы совместимости автомата 48
5.4. Автомат с минимальным числом состояний 50
5.5. Пример минимизации автомата Мили 52
5.6. Пример минимизации автомата Мура 58
6. СТРУКТУРНЫЙ СИНТЕЗ АВТОМАТОВ 64
6.1. Композиция автоматов 64
6.2. Канонический метод структурного синтеза автоматов 66
7. ЭЛЕМЕНТАРНЫЕ АВТОМАТЫ 74
7.1. Элементарные автоматы с двумя входными сигналами 76
7.2. Элементарные автоматы с тремя входными сигналами 79
7.3. Элементарный автомат с четырьмя входными сигналами 84

8. ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ ЭЛЕМЕНТАРНЫХ АВТОМАТОВ 88
9. МИНИМИЗАЦИЯ ПОЛНОСТЬЮ ОПРЕДЕЛЕННЫХ АВТОМАТОВ . 96
10. МЕТОДЫ КОДИРОВАНИЯ СОСТОЯНИЙ АБСТРАКТНЫХ АВТОМАТОВ 100
10.1. Противогоночное кодирование методом развязывания пар переходов 101
10.2. Противогоночное соседнее кодирование 105
10.3. Кодирование состояний автомата, близкое к соседнему 109
10.4. Соседнее кодирование логически смежных состояний 114
Библиографический список 117

Учебное издание

Лупал Алла Матвеевна

ТЕОРИЯ АВТОМАТОВ

Учебное пособие

Редактор *А. В. Подчепаева*
Компьютерная верстка *А. Н. Колешко*

Лицензия ЛР №020341 от 07.05.97. Сдано в набор 26.06.00. Подписано к печати 11.09.00. Формат 60×84 1/16. Бумага тип. №3. Печать офсетная. Усл. печ. л. 6,51. Усл. кр.-отт. 7,53. Уч. -изд. л. 7,0. Тираж 150 экз. Заказ №

Редакционно-издательский отдел
Лаборатория компьютерно-издательских технологий
Отдел оперативной полиграфии
СПбГУАП
190000, Санкт-Петербург, ул. Б. Морская, 67