ЖУРНАЛ
ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ и МАТЕМАТИЧЕСКОЙ ФИЗИКИ

Том 1 Май 1961 Июнь № 3

НЕКОТОРЫЕ ПРОБЛЕМЫ СИНТЕЗА ЦИФРОВЫХ АВТОМАТОВ

В. М. ГЛУШКОВ
(Киев)

Введение

Бурный рост современной дискретной вычислительной техники предъявляет большие требования к теории синтеза схем дискретных преобразователей информации, которые мы на всем протяжении настоящей статьи будем условно называть *цифровыми автоматами*. Разумеется, такие преобразователи могут иметь дело не обязательно с цифровой, но и с произвольной буквенной информацией, однако название *цифровые автоматы* в настоящее время уже достаточно укоренилось, и мы не будем отступать от установившейся традиции.

Процесс синтеза схем современных цифровых автоматов естественно разделяется на ряд самостоятельных этапов.

1. Первый этап, который мы будем называть также подготовительным этапом или этапом блочного синтеза, применяется только в случае синтеза достаточно сложных автоматов, какими являются, например, современные универсальные электронные цифровые машины. Суть этого этапа состоит в том, что схему будущего автомата представляют в виде совокупности отдельных крупных частей — блоков — и описывают частные задачи, которые должен решать каждый блок. Разбиение на блоки производится обычно в соответствии с блок-схемой алгоритма, который должен реализоваться синтезируемым автоматом.

Этап блочного синтеза может подразделяться на отдельные еще более мелкие этапы, на каждом из которых полученная на предыдущем этапе более грубая блок-схема подвергается дальнейшему расщеплению на блоки. Так обычно поступают при синтезе особо сложных автоматов. Наоборот, при синтезе достаточно простых автоматов этап блочного синтеза, как правило, опускается.

2. На втором этапе производится точное описание задачи, решаемой синтезируемым автоматом (или отдельным блоком сложного автомата) в терминах того или иного алгоритмического языка. При этом фиксируется входной и выходной алфавиты для перерабатываемой автоматом информации и устанавливается закон соответствия (алгоритм) между входной и выходной информацией. Этот этап будем условно называть этапом постановки и уточнения задачи.

3. На третьем этапе, который мы будем называть этапом абстрактного синтеза автомата, определяется необходимый объем памяти (число внутренних состояний автомата) и устанавливается закон перехода в памяти автомата под влиянием входных сигналов. На этом

же этапе устанавливается и закон появления того или иного выходного сигнала в соответствии с входным сигналом и состоянием памяти автомата, приведенный к одному и тому же моменту времени.

4. На четвертом этапе производится структурный синтез запоминающей части автомата. При этом осуществляется фактический выбор элементов памяти (триггеры, линии задержки и т. п.), устанавливается способ кодирования букв входного и выходного алфавита конечными совокупностями сигналов, циркулирующих в выбранных элементах памяти (в подавляющем большинстве случаев эти сигналы имеют двоичную природу).

Наконец, на этом этапе выписываются так называемые *канонические уравнения* для логических функций, описывающих обратную связь в синтезируемом автомате. Канонические уравнения задают сигналы, которые требуется подать на входы выбранных элементов памяти, как функции выходных сигналов всех этих элементов и входных сигналов всего автомата в целом. Алфавит, в котором задаются сигналы на этом этапе синтеза, носит название *внутреннего алфавита автомата*. Как уже отмечалось выше, чаще всего этот алфавит является двоичным.

5. На пятом этапе осуществляется так называемый комбинационный синтез. Для этой цели выбирается набор логических элементов (элементарных автоматов без памяти) и строится схема из этих элементов, реализующая функции, задаваемые каноническими уравнениями.

6. Наконец, на шестом этапе производится анализ схем с точки зрения надежности циркуляции сигналов в них и, в случае необходимости, дополнение схемы различного рода усилительными и восстанавливающими элементами.

Помимо описанных задач, на ряде этапов (в особенности на первом, третьем и пятом) производится минимизация найденного решения. Разумеется, такая поэтапная минимизация не может, вообще говоря, привести к выбору самого лучшего варианта. Такой вариант, как правило, может быть найден лишь в результате эквивалентных преобразований всей схемы в целом (не разделенной на запоминающую и комбинационную логическую части). Однако огромная сложность такого пути приводит к тому, что на практике гораздо более удобной и эффективной является именно поэтапная минимизация.

В настоящее время наиболее разработанным является пятый этап, т. е. этап комбинационного синтеза. Теория комбинационного синтеза начала развиваться со второй половины 30-х годов как теория синтеза релейно-контактных схем. Как выяснилось впоследствии, бо́льшая часть разработанных методов синтеза релейно-контактных схем может быть использована и при синтезе схем, составленных из других элементов.

Развитие теории релейно-контактных схем привело к решению (в рамках этой теории) также и некоторых задач четвертого этапа синтеза и, прежде всего, задачи составления канонических уравнений (см. [1], [2]).

Успешно развивается общая теория алгоритмов и алгоритмических языков, составляющая теоретическую основу второго этапа синтеза.

Несколько хуже обстоит дело с остальными этапами. Что касается

первого и последнего этапов, то они до настоящего времени не имеют сколько-нибудь законченного вида и находятся, фактически, вне рамок математики. Здесь господствуют эмпирические методы, в значительно большей степени опирающиеся на интуицию проектировщика, традиции и удобство технологии, чем на законченную математическую теорию. Об общей теории синтеза применительно к этим этапам говорить в настоящее время еще рано.

До самого последнего времени подобное положение сохранялось и на третьем этапе: определение необходимых переходов в памяти автомата производилось на основании интуиции проектировщика. Сейчас положение здесь резко изменилось. После выхода в свет работы Клини [3] и ряда последовавших за ней работ этот этап синтеза сделался предметом достаточно стройной и разработанной математической теории.

Общий материал в вопросе о синтезе цифровых автоматов в настоящее время настолько велик, что его невозможно охватить сколько-нибудь полно в одной обзорной статье и даже в монографии. Поэтому в настоящей статье излагаются лишь некоторые результаты, связанные в основном с третьим и отчасти со вторым и четвертым этапами синтеза.

Отбор материала произведен таким образом, чтобы, с одной стороны, изложить основы созданной здесь за последние годы математической теории, а с другой стороны — дать возможность специалистам в области дискретной вычислительной техники использовать результаты этой теории для решения конкретных практических задач, возникающих при синтезе цифровых автоматов.

В связи с этим из статьи полностью исключены вопросы абстрактно-алгебраической теории автоматов, хотя и представляющие подчас большой научный интерес, но относительно менее полезные при решении практических задач синтеза конкретных автоматов. Вопросы такого рода составляют предмет другой статьи автора, которая будет напечатана в журнале «Успехи математических наук».

Статья носит обзорный характер, хотя и опирается в значительной своей части на оригинальные результаты самого автора (в том числе и такие, которые нигде ранее не публиковались). Большое место в статье уделяется таким результатам, полученным Луфенкампом и Хеном [4], [5]. Материал, изложенный в настоящей статье, достаточен для того, чтобы дать возможность математику или инженеру, знакомому с комбинационным синтезом (например, по статье С. В. Яблонского [6]), осуществить полный логический синтез цифровых автоматов относительно небольшой сложности.

§ 1. Абстрактные автоматы и автоматные соответствия

Абстрактным автоматом мы будем называть объект, способный принимать различные состояния из некоторого фиксированного множества (*множества внутренних состояний* автомата), переходить из одного состояния в другое под воздействием сигналов из некоторого фиксированного конечного множества (множества входных сигналов, называемого также *входным алфавитом* автомата) и выдавать выходные

сигналы из некоторого фиксированного конечного множества выходных сигналов (*выходного алфавита* автомата). Одно из внутренних состояний автомата обычно фиксируется в качестве его *начального состояния*.

Переходы автомата из одного состояния в другое определяются некоторой функцией, называемой обычно *функцией переходов*, а другая функция — так называемая *функция выходов* автомата — определяет появление тех или иных сигналов на его выходе.

Абстрактный автомат функционирует в дискретном времени, принимающем последовательные целые положительные значения 1, 2, ... В любой фиксированный момент (дискретного) времени автомат находится в каком-то определенном внутреннем состоянии, получает определенный входной сигнал и выдаст вполне определенный выходной сигнал.

Итак, абстрактный автомат A определяется заданием трех множеств (множества внутренних состояний $\mathfrak{A}$, *входного алфавита* $\mathfrak{X}$, *выходного алфавита* $\mathfrak{Y}$), *элемента* a_1 *множества* $\mathfrak{A}$ (*начального состояния*) *и двух функций (функции переходов* φ *и функции выходов* ψ), *определяющих функционирование автомата в дискретном времени. Функция переходов определяет состояние* $a(t+1)$ *автомата в момент времени* $t+1$ *в зависимости от его состояния* $a(t)$ *и входного сигнала* $x(t)$ *в момент времени* t:

$$a(t+1) = \varphi[a(t),\ x(t)].$$

Функция выходов определяет зависимость выходного сигнала $y(t)$ *от состояния автомата и входного сигнала в тот же самый момент времени* t:

$$y(t) = \psi(a(t),\ x(t)).$$

Как уже отмечалось выше, входной и выходной алфавиты автомата всегда предполагаются конечными. Что же касается множества $\mathfrak{A}$ внутренних состояний, то в общей теории автоматов на него не накладывается обычно никаких ограничений. Если же это множество конечно, то и соответствующий ему абстрактный автомат также называется *конечным автоматом*.

В приведенном выше определении абстрактного автомата предполагается, что обе функции φ и ψ (переходов и выходов) однозначны и всюду определены. Если же, сохранив условие однозначности, предполагать, что эти функции заданы лишь на некоторых, а не обязательно на всех парах $(a(t),\ x(t))$, то мы придем к понятию *частичного автомата*.

Мы ограничимся при этом рассмотрением лишь таких частичных автоматов, для которых функции переходов и выходов на любой заданной паре либо одновременно определены, либо одновременно не определены.

В дальнейшем мы будем рассматривать почти исключительно конечные (полные или частичные) автоматы. Для конечных автоматов функции переходов и выходов задаются обычно конечными таблицами с двумя входами, называемыми, соответственно, таблицами переходов и выходов автомата. Условимся, что столбцы в этих таблицах будут обозначаться внутренними состояниями, а строки — входными сигналами (буквами входного алфавита). Если рассматриваемый автомат — частичный, то в тех местах таблицы, в которых функции переходов и выходов не определены, будем ставить черточки. Зависимость рассматриваемых величин

от времени в таблицах обычно опускается. Начальному состоянию всегда будем относить самый левый столбец таблицы. Иногда таблицы переходов и выходов совмещаются в одну таблицу.

Более наглядным, чем задание с помощью таблиц, является задание абстрактных автоматов с помощью *направленных графов*. При этом вершины графа отождествляются с внутренними состояниями автомата, а соединяющие их стрелки (направленные ребра) обозначаются входными сигналами, вызывающими соответствующий переход в автомате. Вершина, соответствующая состоянию a_i, соединяется стрелкой, обозначаемой через x_j, с вершиной, соответствующей состоянию a_k, тогда и только тогда, когда входной сигнал x_j переводит автомат из состояния a_i в состояние a_k. Обозначенная через x_j стрелка, выходящая из вершины a_i, получает обычно и второе обозначение — выходным сигналом $y_r = \psi(a_i, x_j)$, соответствующим паре (a_i, x_j).

Переход от задания автомата с помощью таблиц к заданию с помощью графа и обратный переход совершаются вполне тривиальным образом и не требуют дальнейших разъяснений. Если, например, автомат A с множеством внутренних состояний (1, 2, 3), входным алфавитом (x, y) и выходным алфавитом (u, v) был задан таблицами переходов и выходов

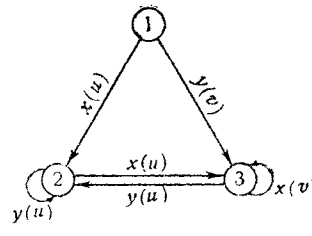
	1	2	3
x	2	3	3
y	3	2	2

	1	2	3
x	u	u	
y	v	u	u

то его можно задать, очевидно, графом, изображенным на фигуре.

Разобранное нами общее понятие абстрактного автомата называется также понятием автомата в смысле Мили или просто *автоматом Мили*.

Как в теории, так и в ряде практических задач синтеза большое значение имеет один частный случай автоматов Мили, называемый обычно *автоматом Мура*.



Автоматом Мура называется такой абстрактный автомат, у которого выходной сигнал в произвольный момент времени t однозначно определяется внутренним состоянием автомата в момент времени $t+1$: $y(t) = \eta[a(t+1)]$. Функция $y = \eta(a)$ называется при этом сдвинутой функцией выходов, а соответствующая ей (однострочечная) таблица — сдвинутой таблицей выходов.

На первый взгляд, в особенности применительно к задачам практического характера, допущение зависимости выхода в настоящий момент времени от состояния автомата в последующий момент времени представляется несколько странным. Однако не следует забывать, что в действительности отсчет моментов времени достаточно условен в силу неявно сделанного нами предположения о мгновенности переходов из одного состояние в другое.

На практике конечная длительность переходных процессов позволяет варьировать момент изменения состояния в пределах одного элементарного промежутка времени, благодаря чему в случае автоматов Мур

можно интерпретировать выходной сигнал так, что он будет зависеть от состояния автомата в тот же самый момент времени. В абстрактной же теории автоматов удобно остаться при сделанном определении, поскольку в противном случае пришлось бы строить две параллельные теории: одну для зависимостей

$$a(t+1) = \varphi[a(t), x(t)], \quad y(t) = \psi[a(t), x(t)],$$

а другую — для зависимостей

$$a(t+1) = \varphi[a(t), x(t)], \quad y(t+1) = \psi[a, (t+1), x(t)].$$

Будучи частным случаем автомата Мили, автомат Мура допускает те же самые способы задания. Удобнее, однако, вместо таблицы выходов задавать его сдвинутой таблицей выходов. Соответственно, на графе автомата Мура выходными сигналами обозначаются не ребра, а те состояния, в которые они входят. Мы условимся говорить, что состояния автомата a_i *отмечаются* соответствующими им выходными сигналами $y = \eta(a_i)$. В случае автоматов Мура счет времени удобно начинать не с первого, а с нулевого момента времени.

Отмеченной таблицей переходов автомата Мура мы будем называть таблицу переходов, над которой помещена (единственная) строка его сдвинутой таблицы выходов, так что над состоянием a_i, обозначающим i-й столбец таблицы, стоит отмечающий его выходной сигнал $\eta(a_i)$.

Легко видеть, что автомат Мура однозначно задается своей отмеченной таблицей переходов. Для того чтобы построить обычную таблицу выходов для автомата Мура, достаточно, очевидно, в его таблицу переходов вместо состояний $a(t+1) = \varphi[a(t), x(t)]$ подставить отмечающие их выходные символы $\eta[a(t+1)]$. Столь же легко осуществляется и обратный переход.

Условимся еще об одном упрощенном обозначении: если a — внутреннее состояние автомата, а x — произвольный входной сигнал, то через ax будем обозначать состояние $\varphi(a, x)$, в которое переходит автомат из состояния a под действием входного сигнала x.

Полезно распространить также определение функций переходов и выходов на произвольные конечные упорядоченные последовательности входных сигналов (слова во входном алфавите). Если $l = x(1)x(2)\ldots x(n)$ — произвольное слово длины n во входном алфавите, то при $n = 1$ функции переходов $\varphi(a, l)$ и выходов $\psi(a, l)$ были уже определены ранее. Для $l_1 = lx(n+1)$ положим:

$$\varphi(a, l_1) = \varphi[\varphi(a, l), x(n+1)], \quad \psi(a, l_1) = \psi[\varphi(a, l), x(n+1)].$$

Тем самым по индукции дано определение продолжения на все слова во входном алфавите.

Подобно тому, как это уже имело место для отдельных входных сигналов (слов длины 1), для произвольного слова l во входном алфавите состояние $\varphi(a, l)$, в которое переводится автомат из состояния a последовательностью входных сигналов l, будем обозначать также через $a \cdot l$.

Для произвольного слова $l_n = x(1)\,x(2)\ldots x(n)$ во входном алфавите автомата, или, более кратко, для произвольного *сложного слова* l_n, будем рассматривать процесс последовательного развертывания этого

...во времени. При этом будут последовательно появляться слова $l_1 = x(1)$, $l_2 = x(1)\,x(2)$, ..., $l_i = x(1)\,x(2)\ldots x(i)$, $l_n = x(1)\,x(2)\ldots x(n)$, которые мы будем называть *начальными отрезками* входного слова l_n. Если слово l_n подать на вход автомата A, находившегося первоначально в некотором начальном состоянии a, то автомат выдаст конечную последовательность $q_n = y(1)\,y(2)\ldots y(n)$ выходных сигналов — слово в выходном алфавите, имеющее ту же самую длину, что и слово l_n. Ясно, что, подавая на вход автомата A любой начальный отрезок l_i слова l_n, мы получим на выходе автомата соответствующий начальный отрезок q_i слова q_n, если, разумеется, исходить при этом из того же самого начального состояния a.

Таким образом, абстрактный автомат A (с фиксированным начальным состоянием a) осуществляет соответствие ξ между словами во входном и выходном алфавитах, удовлетворяющее следующим двум условиям:

1) каждому слову l во входном алфавите $\mathfrak{X}$ соответствие ξ сопоставляет слово $\xi(l)$ в выходном алфавите $\mathfrak{Y}$, имеющее одинаковую со словом l длину;

2) если слово l_1 совпадает с начальным отрезком слова l, то слово $\xi(l_1)$ совпадает с начальным отрезком слова $\xi(l)$, имеющим такую же длину, как слово l_1.

Условием сформулированные условия *условиями автоматности* соответствия ξ, а всякое соответствие между словами в алфавитах $\mathfrak{X}$ и $\mathfrak{Y}$, удовлетворяющее этим двум условиям, назовем *автоматным соответствием*, или *автоматным оператором*.

Нетрудно показать, что *всякое автоматное соответствие может быть реализовано с помощью некоторого абстрактного автомата (не обязательно конечного)*.

В самом деле, пусть автоматное соответствие ξ отображает множество слов в алфавите $\mathfrak{X} = (x_1, \ldots, x_n)$ в множество слов в алфавите $\mathfrak{Y} = (y_1, \ldots, y_m)$. Построим автомат A, внутренними состояниями которого будут всевозможные слова в алфавите $\mathfrak{X}$, причем начальным состоянием будет служить пустое слово e. Функция переходов φ определяется следующим образом: если l — любое состояние автомата (слово в алфавите $\mathfrak{X}$), а x_j — любой входной сигнал, то $\varphi(l, x_j)$ полагается равной слову lx_j. Определив функцию выходов ψ соотношением $\psi(l, x_j) = y_j$, где x_j — последняя буква слова $\xi(lx_j)$, мы получим автомат, который, как нетрудно видеть, реализует исходное соответствие ξ.

Если отображение ξ (соответствие) множества слов в алфавите $\mathfrak{X}$ в множество слов в алфавите $\mathfrak{Y}$ задается частичным автоматом, то оно будет, разумеется, лишь частичным отображением, определенным не на всех словах. Однако для него по-прежнему будут выполняться оба условия автоматности при дополнительном предположении, что $\xi(l)$ существует. Первое условие автоматности при этом приобретает усиленную форму:

1') если $\xi(l)$ существует, а l_1 — начальный отрезок слова l, то $\xi(l_1)$ существует и совпадает с некоторым начальным отрезком слова $\xi(l)$.

Назовем перефразированные таким образом условия условиями *автоматности частичного соответствия* ξ, а всякое частичное соответствие, удовлетворяющее этим условиям, — *частичным автоматным соответствием*.

Легко установить справедливость следующего предложения: *всякое частичное автоматическое соответствие может быть реализовано с помощью некоторого частичного автомата* (*не обязательно конечного*).

Доказательство этого предложения проводится точно таким же способом, как и в случае полного соответствия. Отличие состоит в том, что состояниями частичного автомата будут считаться не все слова входного алфавита, а лишь те из них, на которых определено отображение ξ.

Условия автоматности, на первый взгляд, сильно суживают класс отображений, которые можно задавать с помощью абстрактных автоматов. Хорошо известно, в частности, что требование равенства длин входных и выходных слов не выполняется для большей части алгоритмов, которые должны выполняться теми или иными конкретными автоматами. Это затруднение, представляющееся на первый взгляд весьма серьезным, в действительности легко устраняется с помощью перекодирования входной и выходной информации на основе весьма простого приема.

Стандартный прием превращения любого частичного соответствия ξ между словами в алфавитах $\mathfrak{X}$ и $\mathfrak{Y}$ в частичное автоматное соответствие основан на введении в алфавиты $\mathfrak{X}$ и $\mathfrak{Y}$ не содержавшейся в них ранее буквы α, которую мы будем называть *пустой буквой*. Появление пустой буквы на входе автомата будет соответствовать случаю, когда на вход автомата в действительности ничего не подается. Аналогично, появление пустой буквы в качестве выходного сигнала означает отсутствие каких-либо сигналов на выходе автомата.

Рассмотрим произвольное слово l длины n в алфавите $\mathfrak{X}$, которому первоначально заданное нам частичное соответствие ξ сопоставляет слово $q = \xi(l)$ в алфавите $\mathfrak{Y}$, имеющее длину m. Обозначим через l_1 слово в алфавите $\mathfrak{X}_1 = \mathfrak{X} \cup (\alpha)$, получающееся в результате приписывания к слову l с п р а в а m экземпляров буквы α. Аналогично, через q_1 обозначим слово в алфавите $\mathfrak{Y}_1 = \mathfrak{Y} \cup \{\alpha\}$, получающееся в результате приписывания к слову q с л е в а n экземпляров букв α. Назовем этот прием стандартным приемом выравнивания длин слов.

Определим новое частичное соответствие ξ_1 между словами в алфавитах $\mathfrak{X}_1$ и $\mathfrak{Y}_1$, полагая $q_1 = \xi_1(l_1)$ и повторяя этот прием для любого слова l в алфавите $\mathfrak{X}$, на котором соответствие ξ определено. Доопределим это соответствие на всех начальных отрезках $l_1^{(i)}$ слов l_1, полагая, что $\xi_1(l_1^{(i)})$ совпадает с начальным отрезком слова $\xi(l_1)$, имеющим равную с $l_1^{(i)}$ длину. ■

При таком доопределении возникает угроза потери однозначности соответствия ξ_1, поскольку слово $l_1^{(i)}$ может входить в качестве начального отрезка не только в исходное слово l_1, но и в другие слова, например в слово s_1, полученное в результате применения стандартного приема выравнивания длин слов из некоторого слова s в алфавите $\mathfrak{X}$.

Так как слово s_1 имеет вид $s_1 = s\alpha\alpha\ldots\alpha$, а слово l_1 — вид $l_1 = l\alpha\alpha\ldots\alpha$, где слова s и l не содержат буквы α, то в случае, если слово $l_1^{(i)}$ имеет справа хотя бы одну букву α ($l_1^{(i)} = p\alpha\alpha\ldots\alpha$), $p = s = l$. В этом случае слова s_1 и l_1 должны совпадать между собой и опасности возникновения неоднозначности не возникает.

Остается, таким образом, рассмотреть случай, когда слово $l_1^{(i)} = p$

состоит исключительно из букв алфавита $\mathfrak{X}$. В этом случае, очевидно, длина слова p не превосходит длин слов l и s. Но тогда, в силу стандартного приема уравнивания длин слов, начальные отрезки слов $\xi_1(l_1)$ и $\xi_1(s_1)$, имеющие равную со словом $l_1^{(i)} = p$ длину, состоят сплошь из букв α и, следовательно, совпадают между собой. Таким образом, возникновение неоднозначности исключено и в этом случае.

Построенное нами частичное соответствие ξ_1 между словами в алфавитах $\mathfrak{X}$ и $\mathfrak{Y}$ по самому способу своего построения удовлетворяет обоим условиям автоматности для частичных соответствий и представляет собой искомое частичное автоматное соответствие.

Описанный прием превращения любого частичного соответствия в автоматное является совершенно общим, однако именно в силу своей общности он не всегда приводит к самому экономному (с точки зрения расходования дополнительных букв) решению. Это обстоятельство особенно легко уяснить на случае, когда само исходное частичное соответствие ξ удовлетворяло обоим условиям автоматности. Ясно, что самым экономным решением в рассматриваемом случае будет $\xi_1 = \xi$. Между тем описанный стандартный прием (который действует и в этом случае) приведет к удвоению длин исходных слов, участвующих в соответствии.

Таким образом, найденный общий прием не избавляет нас от необходимости поиска более экономных решений. Заметим, что большой интерес представляет проблема нахождения экономного перекодирования соответствия, заданного в том или ином алгоритмическом языке (например, на языке нормальных алгоритмов) с целью превращения его в автоматное соответствие.

Представляет существенный интерес также задача построения теории алгоритмов, удовлетворяющих условиям автоматности, сокращенно называемых *автоматными алгоритмами*. Один из возможных подходов к теории автоматных алгоритмов развивается в следующем параграфе.

§ 2. События и представление событий в автоматах

Пусть A — произвольный частичный автомат, $\mathfrak{X} = (x_1, \ldots, x_n)$ — его входной, а $\mathfrak{Y} = (y_1, \ldots, y_m)$ — его выходной алфавит. Зафиксируем некоторое начальное состояние a_1 автомата A и для каждого $j = 1, 2, \ldots, m$ рассмотрим множество R_j всех слов l_i в алфавите $\mathfrak{X}$ таких, что $\psi(a_1, l_i) = y_j$ (ψ — функция выходов автомата A).

Иначе говоря, если обозначить через ξ частичное соответствие между словами в алфавитах $\mathfrak{X}$ и $\mathfrak{Y}$, реализуемое частичным автоматом A, то для каждого $j = 1, 2, \ldots, m$ через R_j обозначается множество всех слов l_i в алфавите $\mathfrak{X}$, для которых слова $\xi(l_i)$ существуют и оканчиваются буквой y_j. Множество R_j состоит, таким образом, из всех входных слов, которые, будучи применены к автомату A, вызывают в момент подачи последней буквы входного слова появление выходного сигнала y_j.

Назовем определенное таким образом множество R_j событием, представленным в частичном автомате A выходным сигналом y_j ($j = 1, 2, \ldots, m$). Если M — любое множество выходных сигналов, то событием, представленным в автоматичном автомате A множеством M, мы будем называть объединение событий, представляемых всеми элементами этого множества.

Легко видеть, что множества R_j попарно не пересекаются, а множество S всех слов в алфавите $\mathfrak{X}$, не вошедших ни в одно из множеств R_j ($j = 1, 2, \ldots, m$), состоит из всех з а п р е щ е н н ы х для данного частичного автомата слов. Запрещенными здесь и далее мы будем называть все слова во входном алфавите, которые при подаче их на вход данного частичного автомата приведут хотя бы для одного составляющего их входного сигнала к не определенному в автомате переходу (согласно принятому нами условию, ему соответствует черточка в таблице переходов автомата) или к появлению неопределенного выходного сигнала.

Совокупность всех запрещенных слов S условимся называть областью запрета данного частичного автомата A.

Условимся еще событием в алфавите $\mathfrak{X}$ называть любое множество слов в этом алфавите.

Учитывая введенные определения, мы можем сформулировать полученный выше результат в виде следующего предложения:

2.1. *Задание частичного автоматного соответствия ξ, реализуемого частичным автоматом A с входным алфавитом $\mathfrak{X} = (x_1, \ldots, x_n)$ и с выходным алфавитом $\mathfrak{Y} = (y_1, \ldots, y_m)$, однозначно определяет разбиение множества F всех слов в алфавите $\mathfrak{X}$ на $m + 1$ попарно непересекающихся событий в алфавите $\mathfrak{X}$, а именно событий $R_1, \ldots, R_m$, представленных в автомате A выходными сигналами $y_1, \ldots, y_m$, и области запрета S данного (частичного) автомата A.*

Обратно, зная события $R_1, \ldots, R_m$, представленные в некотором частичном автомате A выходными сигналами $y_1, \ldots, y_m$, мы можем однозначно восстановить реализуемое этим автоматом частичное соответствие ξ между словами входного алфавита $\mathfrak{X}$ и выходного алфавита $\mathfrak{Y}$, не п о л ь з у я с ь функциями переходов и выходов автомата.

В самом деле, пусть нам дано произвольное слово $l = x_{i_1} x_{i_2} \ldots x_{i_n}$ в алфавите $\mathfrak{X}$. Для каждого k ($1 \leqslant k \leqslant n$) найдем выходной сигнал y_{j_k} по правилу: y_{j_k} есть выходной сигнал, представляющий в автомате A событие R_{j_k}, которое содержит начальный отрезок $x_{i_1} x_{i_2} \ldots x_{i_k}$ длины k слова l. Если для всех $k = 1, 2, \ldots, n$ существуют соответствующие им y_{j_k}, то полагаем

$$\xi(l) = \xi(x_{i_1}, \ldots, x_{i_n}) = y_{j_1} y_{j_2} \ldots y_{j_n}.$$

В случае же несуществования хотя бы для одного $k = 1, 2, \ldots, n$ выходного сигнала y_{j_k} с требуемыми свойствами полагаем, что частичное отображение ξ на слове l не определено.

Нетрудно видеть, что в силу определения событий, представленных в автомате, частичное соответствие ξ как раз и будет тем, которое реализуется заданным частичным автоматом A.

Проведенные рассуждения вместе с предложением 2.1 позволяют сформулировать следующее предложение:

2.2. *Задание частичного автоматного соответствия ξ между словами в алфавитах $\mathfrak{X}$ и $\mathfrak{Y} = (y_1, \ldots, y_n)$ эквивалентно заданию событий $R_1, \ldots, R_m$, представленных в реализующем соответствие ξ частичном автомате A выходными сигналами $y_1, \ldots, y_m$.*

Предложение 2.2 подводит базу под изучение автоматных соответ-

ствий (и, в частности, автоматных алгоритмов). Для описания таких соответствий оказывается достаточным задавать разбиение множества всех слов входного алфавита на конечное число попарно непересекающихся событий. Для того чтобы соответствующие описания имели конструктивный характер, необходимо ограничиться рассмотрением лишь таких событий, которые допускают эффективное описание.

Естественно, что простое конструктивное описание допускают прежде всего конечные события, т. е. события, состоящие из конечного числа слов. Их можно описывать с помощью простого перечисления входящих в них элементов. Для характеристики некоторых важных классов бесконечных событий оказывается целесообразным ввести ряд операций на множестве событий, превратив тем самым это множество в алгебру — алгебру событий.

Для тех целей, которые ставит перед собой настоящая статья, наиболее удобной является система из трех операций, представляющих собой модификацию операций, введенных впервые Клини [3] (см. также [7] и [8]).

Первой операцией является операция теоретико-множественного объединения событий. Мы будем обозначать эту операцию символом $\vee$ и называть *дизъюнкцией событий*.

Второй операцией является операция *умножения событий*, которую не следует путать с операцией теоретико-множественного пересечения. Если событие S состоит из слов l_α ($\alpha \in M$), а событие R — из слов q_β ($\beta \in N$), то *произведением* событий S и R называется событие, состоящее из всевозможных слов вида $l_\alpha q_\beta$ ($\alpha \in M$, $\beta \in N$). Операция умножения событий некоммутивна: вообще говоря, события SR и RS различны.

Третьей операцией является так называемая *итерация события*, для обозначения которой мы будем употреблять фигурные скобки, так что $\{S\}$ означает итерацию события S. Итерация любого события S определяется как объединение пустого слова, события $S = S^1$, события $SS = S^2$, события $SSS = S^3$ и т. д. до бесконечности. Иначе говоря, если событие S состоит из слов l_α ($\alpha \in M$), то его итерация $\{S\}$ состоит из всевозможных слов, имеющих вид $l_{\alpha_1} l_{\alpha_2} \ldots l_{\alpha_n}$, где $\alpha_1, \alpha_2, \ldots, \alpha_n \in M$, а $n = 0, 1, 2, 3, \ldots$

В связи с употреблением фигурных скобок для обозначения итерации мы будем называть их *итерационными скобками*. Для обозначения порядка действий будем пользоваться круглыми скобками, которые будут также называться *обыкновенными*. При отсутствии скобок, изменяющих обычный порядок действий, первыми должны выполняться итерации, затем умножения и, наконец, дизъюнкции.

Условимся одноэлементные события, т. е. события, состоящие из одного слова, обозначать просто символом этого слова. Если $\mathfrak{A} = (x_1, \ldots, x_n)$, то *элементарными событиями в этом алфавите* называются $n + 1$ одноэлементных событий $x_1, \ldots, x_n, e$.

Здесь и ниже через e мы будем обозначать *пустое слово*, состоящее из пустого множества букв и имеющее, следовательно, нулевую длину.

Во всем последующем изложении пустое слово e играет лишь служебную, вспомогательную роль. Мы условимся, в частности, не делать различия между событиями, которые отличаются одно от другого лишь

пустым словом. Таким образом, слово можно по желанию либо присоединить, либо исключить из любого рассматриваемого события. Это связано с тем, что в силу принятых нами определений пустое слово не может быть представлено в автомате.

Введем теперь понятие, являющееся одним из центральных во всех последующих рассмотрениях.

Регулярным событием в конечном алфавите $\mathfrak{X} = (x_1,\ldots,x_n)$ *называется любое событие, которое может быть получено из элементарных событий* $x_1,\ldots,x_n$, *е в этом алфавите с помощью применения конечного числа операций дизъюнкции, умножения и итерации.*

Это определение восходит к определению регулярного события, данному еще Клини [3], хотя по форме существенно отличается от него (см. [7]). Заметим, что одно и то же событие допускает различные представления через элементарные события. Каждое такое представление (формулу алгебры событий) мы будем в дальнейшем называть *регулярным выражением*.

Одной из основных задач в алгебре событий является установление законов *эквивалентных преобразований* регулярных выражений, т. е. таких преобразований, которые не меняют представляемых этими выражениями событий (с точностью до пустого слова e).

К числу законов, особенно часто употребляющихся при эквивалентных преобразованиях в алгебре событий, относятся законы ассоциативности для дизъюнкции и для умножения, закон коммутативности для дизъюнкции, левый и правый дистрибутивный законы для умножения по отношению к дизъюнкции $(S(R \vee Q) = SR \vee SQ, (R \vee Q)S = RS \vee QS)$ и др.

Законы дистрибутивности позволяют, в частности, раскрывать скобки и выносить общие множители за скобки, подобно тому как это делается в обычной алгебре. Следует лишь помнить при этом, что умножение в алгебре событий, вообще говоря, некоммутативно.

Любое слово можно представить как произведение элементарных событий — отдельных букв, составляющих это слово. Любое же конечное событие представляется в виде дизъюнкции составляющих его слов. Отсюда, в частности, следует, что все конечные события регулярны.

Использование итерации приводит к построению бесконечных регулярных событий. Вместе с тем нетрудно построить простые примеры бесконечных событий, не являющихся регулярными. Для этой цели достаточно выбрать возрастающую последовательность целых чисел n_1, n_2, ..., n_i, ..., такую, что разности $n_{i+1} - n_i$ $(i = 1,2,\ldots)$ не ограничены в совокупности (этому условию удовлетворяет, например, последовательность квадратов чисел натурального ряда), и в любом входном алфавите $\mathfrak{X}$ построить событие S, состоящее из всех слов в алфавите $\mathfrak{X}$, имеющих длины, равные n_1, n_2 и т. д.

Построенное таким образом событие S обязательно нерегулярно. В самом деле, допуская противное, мы могли бы найти для S некоторое регулярное выражение R. Поскольку событие S бесконечно, это выражение содержит хотя бы одни итерационные скобки, заключающие внутри себя выражение, отличное от пустого слова e. Заменим все остальные итерационные скобки в выражении R пустым словом, а выделенные скобки —

выражением $\{p\}$, где p — произвольное непустое слово из события, заключенного в выделенные скобки. В результате получим регулярное выражение R_1 для некоторого события, содержащегося в событии S.

Из рассмотрения выражения R_1 непосредственно следует, что в событие S входят слова вида *rs*, *rps*, *rpps*, *rppps*,..., длины которых составляют бесконечную возрастающую арифметическую прогрессию. Но это обстоятельство очевидным образом противоречит способу построения события S. Следовательно, событие S не может быть представлено никаким регулярным выражением, т. е. является нерегулярным событием, что и требовалось доказать.

Определим еще понятие *циклической глубины* регулярного выражения, понимая под нею максимальное число вложенных друг в друга пар итерационных скобок, содержащихся в этом выражении. Например, выражение $\{x\{y\}\{x\}\}$ имеет циклическую глубину 2, а выражение $\{x \vee y\} \times \{y\}\{x\}$ — циклическую глубину 1. Под *циклической глубиной регулярного события* условимся понимать минимальную циклическую глубину представляющих его регулярных выражений.

Регулярные события имеют особое значение для абстрактной теории автоматов, поскольку *класс регулярных событий совпадает с классом событий, представимых в конечных автоматах*. В последующих параграфах мы дадим доказательство этого важного предложения, а пока рассмотрим вопрос о соотношении классов событий, представимых в автоматах Мили и Мура.

Общее определение представления событий в автомате, сделанное в начале настоящего параграфа, относилось к автомату Мили. Поскольку автомат Мура является частным случаем автоматов Мили, то это определение применимо в полной мере и к нему. Однако на практике в случае автоматов Мура оказывается удобным представлять события не свойством выхода в момент подачи последнего входного сигнала слов, составляющих события, а свойством состояния автомата, в котором он окажется после поступления на вход автомата слова того или иного события.

Иначе говоря, в случае автоматов Мура принято считать, что события представляются некоторыми м н о ж е с т в а м и с о с т о я н и й а в т о м а т а. В силу определения автоматов Мура, этот способ представления событий оказывается для них совершенно эквивалентным способу представления событий множествами выходных сигналов. Разница состоит лишь в том, что при представлении событий множествами состояний автомата оказывается представимым (с помощью начального состояния) пустое слово *e*, которое не может быть представлено никаким выходным сигналом (если, разумеется, не начинать счет времени с отрицательных моментов времени).

Однако выше мы условились не считать различными события, отличающиеся друг от друга лишь на пустое слово. Поэтому оба способа представления событий (состояниями или выходными сигналами) в случае автоматов Мура оказываются эквивалентными.

Поскольку автоматы Мура представляют собой частный случай автоматов Мили, кажется естественным, что класс событий, представимых

в автоматах Мура, беднее, чем класс событий, представимых в автоматах Мили. Покажем, что на самом деле это не так.

Предположим, что некоторое событие S представлено в автомате Мили A множеством выходных сигналов M. Построим автомат B, внутренними состояниями которого будем считать начальное состояние a_0 автомата A и всевозможные пары вида (a_i, x_j), где a_i пробегает множество N внутренних состояний автомата A, а x_j — всевозможные буквы входного алфавита.

Будем считать, что под влиянием входного сигнала x_k автомат B переходит из состояния (a_i, x_j) в состояние $(a_i x_j, x_k)$, а из состояния a_0 — в состояние (a_0, x_k).

Если $\psi(a_i, x_j)$ — функция выходов автомата A, то превратим B в автомат Мура, определив его с д в и н у т у ю функцию выходов $\eta(b_i)$ соотношением

$$\eta[(a_i, x_j)] = \psi(a_i, x_j)$$

(на начальном состоянии a_0 автомата B сдвинутая функция переходов определяется произвольным образом).

Если $h = gx_j$ — произвольное непустое входное слово, то в автомате A последней буквой соответствующего ему выходного слова будет, очевидно, буква $y = \psi(a_0 g, x_j)$. Что же касается автомата B, то, как нетрудно видеть, слово h переведет его из начального состояния a_0 в состояние $(a_i g, x_j)$.

Таким образом, все непустые слова исходного события будут представлены в автомате B множеством K всевозможных состояний (a_i, x_j), для которых имеет место соотношение $\psi(a_i, x_j) \in M$.

Поскольку событие S было представлено в автомате Мили, оно не могло содержать пустого слова. Следовательно, в автомате Мура B множеством K его внутренних состояний представлено в точности событие S.

Таким образом, нами доказано следующее предложение:

2.3. *Если событие S в m-буквенном алфавите представлено в автомате Мили A некоторым множеством выходных сигналов, то его можно представить также в автомате Мура B некоторым множеством его внутренних состояний. При этом автомат Мура B может быть выбран таким образом, что число его внутренних состояний не превзойдет $mn + 1$, где n — число внутренних состояний автомата A.*

Нетрудно понять, что предложение 2.3 остается справедливым при замене термина *автомат* термином *частичный автомат*.

§ 3. Анализ конечных автоматов

Проблема анализа автомата состоит в нахождении событий, представляемых в автомате множествами выходных сигналов (в случае автоматов Мили) или множествами внутренних состояний (в случае автоматов Мура). Поскольку всякий автомат Мура можно интерпретировать как автомат Мили, достаточно научиться анализировать лишь автоматы Мили.

Мы будем решать проблему анализа лишь для случая конечных автоматов Мили. При этом оказывается, что все представляемые в таких

автоматах события непременно регулярны. Алгоритм анализа должен исходить из таблиц переходов и выходов анализируемого автомата, а в качестве заключительной информации давать регулярные выражения для событий, представимых каждым из выходных сигналов автомата. Событие, представляемое произвольным множеством выходных сигналов, представится как дизъюнкция событий, представленных отдельными выходными сигналами, составляющими данное множество.

Рассмотрим произвольный конечный автомат Мили A с множеством внутренних состояний $(a_1, a_2, \ldots, a_p)$, с входным алфавитом $\mathfrak{X} = (x_1, \ldots, x_n)$ и выходным алфавитом $\mathfrak{Y} = (y_1, y_2, \ldots, y_m)$. Считая заданными начальное состояние a_1, функцию переходов $\varphi(a_i, x_j)$ и функцию выходов $\psi(a_i, x_j)$ автомата A, будем искать регулярное выражение R для события, представленного каким-либо выходным сигналом, например сигналом y_1.

Выпишем внутренние состояния автомата $a_1, a_{j_1}, \ldots, a_{j_k}$, в которые автомат A переводится из начального состояния a_1 последовательными начальными отрезками e, x_{i_1}, $x_{i_1}x_{i_2}$, $\ldots$, $x_{i_1}x_{i_2}\ldots x_{i_k}$ какого-либо входного слова q. Вставляя символы полученных состояний в слово q после соответствующих им начальных отрезков, мы превратим это слово в новое слово

$$q' = a_1 x_{i_1} a_{j_1} x_{i_2} \ldots a_{j_{k-1}} x_{i_k} a_{j_k},$$

которое условимся называть *путем, соответствующим слову* q. Удаляя в заданном пути символы внутренних состояний a_j, мы получим входное слово, соответствующее данному пути.

Мы будем еще употреблять так называемые *урезанные пути*, получающиеся из обычных путей выбрасыванием крайнего правого символа внутреннего состояния a_{j_k}. Путь, соответствующий данному входному слову q, условимся обозначать через q', а урезанный путь — через q''.

Очевидно, что для принадлежности непустого входного слова q событию R_i, представляемому в автомате A выходным сигналом y_i, необходимо и достаточно, чтобы соответствующий слову q урезанный путь q'' оканчивался парой $a_{j_{k-1}}x_{i_k}$, для которой функция выходов принимает значение, равное y_i. Назовем все такие (урезанные) пути *путями типа* y_i или (для любого i) *путями представляющего типа*.

Пути (неурезанные), соответствующие входным словам, переводящим автомат из какого-либо состояния a_j в то же самое состояние a_j, назовем *путями типа* a_j или *путями циклического типа*. Если в некотором пути q' циклического типа a_j не содержатся символы каких-либо внутренних состояний $a_{k_1}, \ldots, a_{k_r}$, то путь q' будем называть также путем типа $a_j\,[a_{k_1}, \ldots, a_{k_r}]$ (символы, стоящие в квадратных скобках, будут при этом называться *запрещенными*).

Путь q' произвольного типа называется *простым*, если соответствующий ему урезанный путь q'' не содержит двух одинаковых символов внутренних состояний. В конечном автомате существует лишь конечное число различных простых путей. Все простые пути любого данного типа могут быть найдены непосредственно по таблице переходов или (для путей представляющего типа) по таблице переходов и таблице выходов автомата.

Построим теперь некоторые вспомогательные события в алфавите $\mathfrak{Z} = (x_1, \ldots, x_n, a_1, \ldots, a_p)$, непустыми элементами которых будут урезанные пути в заданном автомате A. Событие $S(y_1)$ — типа y_1 — определим как событие, состоящее из всех (урезанных) путей типа y_1, простое событие $\mathscr{P}(y_1)$ типа y_1 — как дизъюнкцию всех простых (урезанных) путей типа y_1. Простым событием $\mathscr{P}(t)$ любого данного циклического типа $t = a_j [a_{k_1}, \ldots, a_{k_r}]$ $(j = 1, \ldots, p,\ 0 \leqslant r \leqslant p-1)$ будет называться итерация дизъюнкции всех простых путей типа t (дизъюнкция пустого множества путей есть невозможное событие, итерация которого совпадает с пустым словом e), а событием $S(t)$ типа t — событие, состоящее из всех урезанных путей типа t. Наконец, *условно простым событием* $U(t)$ типа $t = a_j [a_{k_1}, \ldots, a_{k_r}]$ назовем часть события $S(t)$, содержащую слова лишь с одним вхождением символа a_j.

Пусть дано некоторое множество (урезанных) путей типа y_1, заданное с помощью регулярного выражения Q. Ясно, что, вставляя в это регулярное выражение перед любым входящим в него символом внутреннего состояния a_j регулярное выражение события $S(a_j)$ типа a_j (или события типа $a_j [a_{k_1}, \ldots, a_{k_r}]$), мы получим новое регулярное выражение, представляющее по-прежнему лишь пути типа y_1. Назовем такую операцию *вставкой* события $S(a_j)$ в событие Q.

Пусть теперь q'' — произвольный (урезанный) путь типа y_1.

Первым (слева) символом внутреннего состояния, входящим в этот путь, будет символ a_1. Выделим также последнее (крайнее правое) вхождение символа a_1 в путь q'': $q'' = a_1 \ldots a_1 s$, где слово s уже не содержит символа a_1. Тогда путь q'' можно представить в виде произведения некоторого числа слов условно простого события $U(a_1)$ типа a_1 и слова $a_1 s$.

В слове s выделяем первый (слева) символ внутреннего состояния: $s = x_{i_k} a_{j_k} \ldots$ Снова, находя последнее вхождение этого символа в слово s, получаем возможность представить слово s в виде произведения буквы x_{i_k}, некоторого числа слов условно простого события типа $a_{j_k} [a_1]$ и некоторого слова $a_{j_k} r$, где слово r не содержит уже двух символов внутренних состояний: a_1 и a_{j_k}.

Продолжая этот процесс далее, придем к выводу, что путь q'' содержится в событии, которое получается в результате вставки в некоторый простой путь типа y_1 перед единственной входящей в него буквой a_1 условно простого события типа a_1, а перед следующими по порядку символами внутренних состояний $a_{j_k}, a_{j_l}, \ldots$, входящими в этот путь,— условно простых событий типа $a_{j_k} [a_1]$, $a_{j_l} [a_1, a_{j_k}]$, ... и т. д.

Но точно такой же процесс может, очевидно, быть повторен со словами условно простых событий, которые были выделены из исходного пути q''. После этого придем к выводу, что путь q'' типа y_1 входит в событие, которое получается в результате вставки в простое событие $\mathscr{P}(y_1)$ типа y_1 уже не условно простых, а обычных простых событий типов a_1, $a_{j_k} [a_1]$,... и последующей вставки в пути, из которых состоят вставленные простые события, условно простых событий некоторых типов $a_x [a_1], a_y [a_1, a_x], \ldots$ — для вставленного на первом этапе простого события типа a_1, некоторых типов $a_u [u_1, a_{j_k}], a_v [a_1, a_{j_k}, a_n], \ldots$ — для простого события типа $a_{j_k} [a_1]$ и т. д.

Продолжая подобным образом, придем к выводу, что и на втором этапе можно вставлять не условно простые, а обычные простые события, вставляя, в свою очередь (уже на третьем этапе), в составляющие их слова события еще более высоких (в смысле числа запрещенных букв) циклических типов.

Увеличивая число этапов последовательных вставок, придем к вставке событий циклических типов, у которых все буквы, кроме одной, являются запрещенными. Поскольку для такого рода типов разницы между условно простыми и обычными простыми событиями одинакового типа уже не существует, то процесс увеличения числа этапов и новых вставок тем самым завершается.

В результате приходим к выводу, что путь q'' входит в событие S_1, получающееся в результате конечного числа последовательных вставок (разбитых на ряд этапов) в простое событие типа y_1 простых событий все более и более высоких циклических типов. Ввиду произвольности выбора пути q'' событие S_1 содержит в себе событие $S(y_1)$.

Вместе с тем, как уже отмечалось выше, процесс вставок, подобный описанному, не может привести к событию, содержащему пути отличного от y_1 типа. Следовательно, $S_1 = S(y_1)$, а описанный нами процесс вставок дает регулярное выражение R_1 для события $S(y_1)$, состоящее из всех путей типа y_1.

Опуская теперь в регулярном выражении R_1 все символы внутренних состояний (заменяя их пустым словом), получим регулярное выражение R, которое, как легко видеть, является регулярным выражением для искомого события, представленного в автомате A выходным сигналом y_1.

Нами доказано следующее предложение:

3.1. *Событие, представленное в произвольном конечном автомате Мили (а следовательно, и в произвольном конечном автомате Мура) любым множеством выходных сигналов, обязательно регулярно. Существует общий конструктивный прием (алгоритм анализа конечных автоматов), позволяющий находить регулярные выражения для событий, представленных множествами выходных сигналов в произвольном конечном автомате.*

Описанному алгоритму анализа конечных автоматов можно придать практически более удобную форму (см. [9]).

Для этой цели будем оперировать не с событиями в множестве путей, а с формальными выражениями, называемыми *комплексами*.

Для любого множества M выходных сигналов заданного конечного автомата Мили A *комплексом типа* M (или *комплексом выходного типа*) называется дизъюнкция всех простых урезанных путей, кончающихся парами $a_j x_i$, которым соответствуют выходные сигналы, содержащиеся в множестве M.

Комплексом типа $a_i[a_{i_1}, \ldots, a_{i_r}]$ (*комплексом циклического типа*) называется формальное выражение, получаемое в результате объединения знаком дизъюнкции всех простых путей типа $a_i[a_{i_1}, \ldots, a_{i_r}]$ с вычеркнутой буквой a_i и заключения полученного формального многочлена в итерационные скобки ($a_i, a_{i_1}, \ldots, a_{i_r}$ — любые попарно различные внутренние состояния автомата и $0 \leqslant r \leqslant p - 1$, где p — число внутренних состояний автомата A).

Первый шаг алгоритма анализа. По таблицам переходов и выходов автомата и данному (представляющему) множеству выходных сигналов путем перебора всех возможных вариантов простых путей находятся комплекс $K(M)$ типа M и комплексы $K(a_i)$ для всех внутренних состояний a_i автомата A.

Второй шаг. По комплексам $K(a_i)$ исключением ненужных термов в итерационных скобках находятся нужные для дальнейших построений комплексы высших циклических типов $a_i[a_{i_1}, \ldots, a_{i_r}]$ $(r \geqslant 1)$.

Третий шаг. Исходя из комплекса типа M последовательно заменяются все символы внутренних состояний a_i комплексами циклических типов, пока не получим выражение R, не содержащее ни одного из символов внутренних состояний. Правило замены можно сформулировать следующим образом: *если путь* $a_1 x_{i_1} a_{j_1} x_{j_2} a_{j_2} \ldots$ *входит* j *в комплекс выходного типа* M, *то буква* a_1 *заменяется комплексом типа* a_1, *буква* a_{j_1} — *комплексом типа* $a_{j_1}[a_1]$, *буква* a_{j_2} — *комплексом типа* $a_{j_2}[a_1, a_{j_1}]$ *и т. д. Если член* $x_{i_1} a_{j_1} x_{i_2} a_{j_2} \ldots$ *входит в комплекс типа* $a_i[N]$, *где* N — *множество* (*быть может, пустое*) *внутренних состояний, отличных от* a_i, *то буква* a_{j_1} *заменяется комплексом типа* $a_{j_1}[a_i, N]$, *буква* a_{j_2} — *комплексом типа* $a_{j_2}[a_i, a_{j_1}, N]$ *и т. д.*

На третьем шаге в результате применения конечного числа раз правила замены получаем искомое регулярное выражение R для события, представленного в автомате A множеством M выходных сигналов.

Из описанного алгоритма непосредственно вытекает предложение:

3.2. *Всякое событие, представленное в конечном автомате Мили (или Мура), имеющем* n *внутренних состояний, допускает регулярное выражение, циклическая глубина которого не превосходит* n.

В качестве примера найдем регулярное выражение для события S, представленного выходным сигналом v в автомате, таблицы переходов и выходов которого были выписаны в § 1.

Непосредственно по таблицам находим комплекс $K(v)$ типа v:

$$K(v) = {}_1y \vee {}_1y_3x \vee {}_1x_2x_3x,$$

и комплексы типов 1, 2 и 3:

$$K(1) = e, \quad K(2) = \{y \vee x_3y\}, \quad K(3) = \{x \vee y_2x\}.$$

Выпишем некоторые комплексы высших циклических типов:

$$K(2[1]) = K(2), \quad K(3[1,2]) = \{x\},$$
$$K(2[3]) = K(2[1,3]) = \{y\}, \quad K(3[1]) = K(3).$$

Обозначая операцию вставки комплексов стрелкой, получим следующую последовательность вставок:

$$\begin{aligned} K(v) &\to y \vee yK(3[1])\,x \vee xK(2[1])\,xK(3[1,2])\,x = \\ &= y \vee y\{x \vee y_2x\}\,x \vee x\{y \vee x_3y\}\,x\{x\}\,x \to \\ &\to y \vee y\{x \vee yK(2[1,3])\,x\}\,x \vee x\{y \vee xK(3[1,2])\,y\}\,x\{x\}\,x = \\ &= y \vee y\{x \vee y\{y\}\,x\}\,x \vee x\{y \vee x\{x\}\,y\}\,x\{x\}\,x. \end{aligned}$$

Последнее из полученных регулярных выражений и будет искомым регулярным выражением для события S. Оно допускает дальнейшие

преобразования и упрощения с использованием соотношений, существующих в алгебре событий. Однако мы не будем приводить соответствующих выкладок.

§ 4. Абстрактный синтез конечных автоматов

Задача абстрактного синтеза противоположна задаче анализа конечных автоматов: необходимо найти эффективный метод, позволяющий по регулярным выражениям представленных в автомате событий найти таблицы переходов и выходов этого автомата.

Нетрудно понять, что задача синтеза автоматов Мура является более общей, чем задача синтеза автоматов Мили. В самом деле, поскольку всякий автомат Мура можно интерпретировать как автомат Мили, то, научившись синтезировать автоматы Мура, мы тем самым научимся синтезировать и автоматы Мили. Мы будем поэтому решать задачу синтеза автоматов Мура.

Пусть в некотором конечном алфавите $\mathfrak{X} = (x_1, \ldots, x_n)$ задано p регулярных выражений: $R_1, \ldots, R_p$. Занумеруем все вхождения букв алфавита $\mathfrak{X}$ в выражения $R_1, \ldots, R_p$ последовательными натуральными числами, которые в дальнейшем будем называть *индексами соответствующих мест* этих выражений. Особо подчеркнем, что различные вхождения одной и той же буквы алфавита $\mathfrak{X}$ получают при этом различные индексы.

При разворачивании регулярного выражения в слово каждая из последовательно выписываемых букв этого слова отождествляется нами с тем или иным вхождением соответствующей буквы в разворачиваемое выражение. Условимся говорить, что при таком отождествлении мы попадем в те или иные м е с т а регулярного выражения. А именно, при отождествлении последней выписанной буквы со вхождением, занумерованным индексом j, будем говорить, что мы находимся в j-м месте соответствующего регулярного выражения.

Будем говорить, что j-е место регулярного выражения R x_k-следует за i-м местом, если после отождествления последней буквы некоторого слова q с вхождением, имеющим индекс i, возможно осуществить отождествление последней буквы слова qx_k со вхождением, имеющим индекс j. В каждом регулярном выражении выделяется еще *начальное место*, которому приписывается индекс 0 (одинаковый для всех данных регулярных выражений). Если в процессе отождествления п е р в а я буква x_k некоторого слова отождествляется с каким-либо ее вхождением в регулярное выражение, имеющим индекс j, то говорят, что j-е место x_k-следует за нулевым (начальным) местом (общим для всех заданных регулярных выражений).

Наконец, если в процессе отождествления последняя буква некоторого слова, принадлежащего событию с регулярным выражением R, отождествляется с ее вхождением в R, имеющим индекс j, то j-е место выражения R называется *конечным местом* этого выражения.

Для любого конечного множества регулярных выражений $R_1, \ldots, R_p$ в одном и том же алфавите $\mathfrak{X}$, пользуясь определенным выше порядком действий в алгебре событий, нетрудно составить *таблицу следования*

мест. Строки такой таблицы обозначаются буквами алфавита $\mathfrak{X}$, а столбцы — индексами всех мест выражений $R_1, \ldots, R_p$. На пересечении x_i-й строки с j-м столбцом таблицы следования выписываются индексы всех мест, которые x_i-следуют за j-м местом. В случае если таких индексов нет, в соответствующем месте таблицы ставится специальный значок, обозначающий пустое множество индексов. Условимся в качестве такого значка употреблять звездочку.

Построим теперь автомат Мура A, внутренними состояниями которого будут всевозможные подмножества индексов мест в заданных регулярных выражениях $R_1, \ldots, R_p$ (включая пустое подмножество). Функция переходов φ этого автомата строится следующим образом: для любого состояния a_i автомата A (множества индексов мест заданных событий) и любой буквы x_j входного алфавита состояние $a_k = \varphi(a_i, x_j)$ определяется как множество индексов всех мест, которые x_j-следуют хотя бы за одним из мест, индексы которых входят в a_i.

Сдвинутая функция выходов η автомата Мура A строится для выходного алфавита $\mathfrak{Y}$, состоящего из всевозможных подмножеств (включая пустое подмножество) множества всех символов $R_1, \ldots, R_p$ заданных регулярных выражений. Для любого состояния a_i (множества индексов) автомата A в качестве $\eta(a_i)$ выбирается множество всех тех регулярных выражений $R_1, \ldots, R_p$, для которых хотя бы один из индексов, входящих в a_i, является индексом конечного места.

Нами построен некий конечный автомат Мура A. Из способа построения его функций переходов и выходов непосредственно следует, что он представляет (при выборе начального состояния 0) каждое из заданных событий $R_1, \ldots, R_p$ и дополнение S их объединения. Событие оказывается представленным множеством всех тех выходных сигналов (множеств символов $R_1, \ldots, R_p$), в состав которых входит символ R_i ($i = 1, 2, \ldots, p$). Что же касается события S, то оно представляется, очевидно, пустым множеством символов $R_1, \ldots, R_p$.

Мы доказали следующее предложение:

4.1. *Любое регулярное событие может быть представлено в конечном автомате. Существует единый конструктивный прием (алгоритм синтеза), позволяющий по любому конечному множеству регулярных событий, заданных регулярными выражениями, построить представляющие эти события конечные автоматы Мура или Мили.*

Объединяя теоремы 3.1 и 4.1, получим:

4.2. *Регулярные события и только они представимы в конечных автоматах.*

Аналогичный результат для автоматов специального вида (нервных сетей) и для более громоздкого по форме определения регулярного события был получен ранее Клини [3].

Из теоремы 3.1 и 4.1 следует:

4.3. *Пересечение двух (а значит, и любого конечного числа) регулярных событий, равно как и дополнение (в множестве всех слов в основном алфавите) любого регулярного события, оказываются также регулярными событиями.*

Описанный выше алгоритм синтеза конечных автоматов допускает также следующую, более удобную для практических целей интерпретацию (см. [7]).

Пусть нам дано p регулярных выражений $R_1,\ldots,R_p$ в произвольном конечном алфавите $\mathfrak{X} = (x_1,\ldots,x_n)$. Если какое-либо из выражений R_i является дизъюнкцией нескольких термов, то можно, не нарушая общности, считать, что оно заключено в обычные (неитерационные скобки). *Местами* в выражениях $R_1,\ldots,R_p$ будем называть специально вводимые знаки раздела (вертикальные черточки), ставящиеся между любыми двумя знаками (буквами, скобками, знаками дизъюнкции) этих выражений, а также слева от выражения (начальное место) и справа от выражения (конечное место).

Места, непосредственно слева от которых стоит буква основного алфавита $\mathfrak{X}$, и начальное место называются *основными местами*, а места, непосредственно справа от которых стоит буква алфавита $\mathfrak{X}$,— *предосновными*. Начальные места всех выражений $R_1,\ldots,R_p$ отождествляются между собой в одно единственное начальное место.

Все основные места обозначаются различными неотрицательными целыми числами — *основными индексами* этих мест. Начальное место получает при этом основной индекс 0.

Действие каждого основного индекса распространяется не только на соответствующее место, но и на места (основные и неосновные), ему подчиненные. Правило подчинения мест выражает порядок действий в алгебре событий. Оно определяется следующим п р а в и л о м р а с п р о с т р а н е н и я и н д е к с о в:

Индексы места перед любыми скобками (итерационными или обыкновенными) распространяются на начальные места всех термов, стоящих внутри этих скобок. Индексы конечного места любого терма, заключенного в скобки, распространяются на место, непосредственно следующее за этими скобками. Индексы места, непосредственно предшествующего итерационным скобкам или символу пустого слова, распространяются на место, непосредственно следующее за этими скобками (соответственно, за данным символом e). Наконец, индексы места, непосредственно следующего за итерационными скобками, распространяются на начальные места всех заключенных в эти скобки термов.

Все индексы, возникшие на основных и неосновных местах в результате применения только что сформулированного правила, называются *неосновными*. Само правило при этом должно применяться до тех пор, пока его применение не будет более приводить к появлению новых индексов ни на одном месте.

Индексация заданных регулярных выражений, разметка мест и распространение индексов согласно сформулированному правилу составляют п е р в ы й ш а г алгоритма синтеза.

В т о р о й ш а г состоит в построении таблицы переходов искомого автомата A. При этом входными сигналами служат буквы исходного алфавита $\mathfrak{X}$, а внутренние состояния автомата отождествляются с множествами основных индексов. Условимся для определенности обозначать эти множества дизъюнкцией составляющих их индексов, а для обозначения пустого множества индексов употреблять звездочку.

П р а в и л о п о с т р о е н и я т а б л и ц ы п е р е х о д о в автомата состоит в следующем:

Начальным состоянием автомата A служит одноэлементное множество, состоящее из индекса 0. Состояние a_i переводится входным сигналом x_k в состояние a_j, состоящее из основных индексов всех основных мест, отделенных буквой x_k от непосредственно предшествующих им предосновных мест, в числе индексов которых (основных или неосновных) содержится хотя бы один индекс из числа индексов, входящих в состояние a_i.

При практическом применении сформулированного правила удобно выделять основные индексы, помещая их над специально проводимой для этой цели горизонтальной чертой. Все индексы (основные и неосновные) предосновных мест также целесообразно выделять, например заключая их в прямоугольную рамку. Заметим также, что при построении таблицы переходов достаточно ограничиться лишь теми состояниями, которые будут появляться в процессе фактического построения таблицы, отправляясь от начального (нулевого) состояния.

Третий шаг синтеза состоит в построении сдвинутой таблицы выходов или — что то же самое — в отметке состояний автомата A соответствующими им выходными сигналами. В качестве выходных сигналов выбираются различные множества символов исходных регулярных выражений (включая пустое множество). Правило отметки состояний состоит в следующем:

Состояние a_i отмечается множеством тех символов выражений $R_1,\ldots,R_p$, в состав индексов (основных и неосновных) конечных мест которых входит хотя бы один индекс из a_i.

Состояния, отмеченные пустым множеством символов $R_1,\ldots,R_p$, будут называться также *неотмеченными*.

Заметим, что построенный автомат Мура A представляет событие R_i множеством всех тех выходных сигналов, в состав которых входит символ R_i ($i = 1,2,\ldots,p$).

Четвертый шаг алгоритма синтеза состоит в переобозначении внутренних состояний и выходных сигналов с целью более простой записи таблицы переходов и сдвинутой таблицы выходов. При этом внутренние состояния чаще всего просто нумеруются последовательными натуральными числами $1,2,\ldots,k,\ldots$

Наконец, пятый шаг алгоритма синтеза употребляется тогда, когда требуется синтезировать не автомат Мура, а автомат Мили. Он заключается в построении обычной (не сдвинутой) таблицы выходов. Как уже указывалось в § 1, для этой цели достаточно в таблицу переходов подставить вместо внутренних состояний отмечающие их выходные сигналы.

При решении практических задач, возникающих при синтезе автоматов, часто оказывается удобным приписывать некоторым основным местам одинаковые основные индексы, производя тем самым отождествление этих мест. Нетрудно показать, что такое отождествление оказывается возможным, если отождествляемым местам подчинены одинаковые множества предосновных и конечных мест (места, удовлетворяющие этому условию, называются *подобными*).

Другой случай, когда оказывается возможным отождествление мест, относится к так называемым *соответственным местам*. Соответственными

называются все те места в разных регулярных выражениях $R_1,\ldots,R_p$ или в различных термах, заключенных в одни и те же скобки, к которым ведут одинаковые пути (множества слов) от начального места или, соответственно, от места, непосредственно предшествующего скобкам.

Ясно, что при применении описанного выше алгоритма синтеза основные индексы соответственных мест будут входить в состояния синтезируемого автомата всегда вместе. Именно этим и обусловливается возможность одинаковой их индексации. Обоснование возможности отождествления подобных мест вытекает из алгоритма минимизации, описываемого в следующем параграфе.

Заметим, что отождествления мест следует делать лишь по одному из признаков (подобия или соответственности), так как одновременные отождествления по обоим признакам могут привести к ошибкам.

В частности, поскольку отождествление начальных мест выполнено, фактически, по признаку соответственности, нельзя, вообще говоря, при наличии более чем одного события отождествлять начальное место в каком-либо событии с другим местом по признаку подобия.

Непосредственно из описанного алгоритма синтеза вытекает справедливость следующего предложения:

4.4. *События, заданные регулярными выражениями* $R_1, \ldots, R_p$ *в некотором конечном алфавите* $\mathfrak{X}$, *могут быть представлены в конечном автомате* (*Мили или Мура*), *имеющем не более чем* 2^{n+1} *внутренних состояний, где* n — *число вхождений букв алфавита* $\mathfrak{X}$ *в выражения* $R_1, \ldots, R_p$.

Рассмотрим теперь, какие изменения в алгоритм синтеза следует ввести в том случае, когда, кроме исходных событий $R_1, \ldots, R_p$, в алфавите $\mathfrak{X} = (x_1, \ldots, x_n)$ задана также область запрета S.

Область запрета может быть задана либо с помощью некоторого регулярного выражения, либо как совокупность слов в алфавите $\mathfrak{X}$, не вошедших ни в одно из событий $R_1, \ldots, R_p$. Оба эти способа, по существу, эквивалентны между собой, поскольку, в силу теорем 4.2 и 4.3, от первого способа задания можно перейти ко второму и наоборот.

Нетрудно заметить, что область запрета S по самому своему определению допускает умножение справа на совокупность F всех слов (включая пустое слово) в алфавите $\mathfrak{X}$: $SF = S$. Поэтому при задании области запрета регулярным выражением R можно, не нарушая общности, предполагать, что выражение R имеет вид

$$R = R_1 \{x_1 \vee x_2 \vee \ldots \vee x_n\}.$$

Ясно, что описанный выше алгоритм синтеза дает решение задачи и в случае наличия области запрета. Однако при этом многие переходы в синтезированном автомате окажутся лишними в том смысле, что они никогда не будут использоваться при реальной работе автомата. Задача состоит в том, чтобы выявить все такие переходы и построить вместо обычного (вполне определенного) автомата частичный автомат, в таблицах переходов и выходов которого в местах запрещенных переходов стоят черточки. Переход к частичному автомату дает дополнительные возможности к последующему упрощению автомата.

Решение указанной задачи в случае задания области запрета как

совокупности слов в алфавите $\mathfrak{X}$, не вошедших ни в одно из заданных регулярных выражений $R_1,\ldots,R_p$, проводится совершенно очевидным способом:

1) после выполнения описанного выше алгоритма синтеза выходной сигнал, обозначенный пустым множеством символов $R_1,\ldots,R_p$, будет, как легко видеть, соответствовать появлению запрещенного входного слова. Следовательно, достаточно заменить этот выходной сигнал в таблице выходов черточкой и поставить также черточки во всех местах таблицы переходов, соответствующих появлению запрещенного выходного сигнала (при наложении таблицы выходов на таблицу переходов места отмеченные черточкой, в обеих таблицах должны совпадать);

2) в случае задания области запрета регулярным выражением S следует применить обычный алгоритм синтеза к выражениям $S,R_1,\ldots,R_p$ и считать запрещенными все выходы, обозначенные множествами, в состав которых входит символ S. Запрещенные выходы в таблице выходов заменяются черточками, которые способом, уже описанным выше, переносятся на таблицу переходов.

Ясно, что такой прием действительно приводит к решению поставленной задачи.

Заметим, что, как уже отмечалось выше, в рассмотренном случае следует считать, что выражение S имеет вид

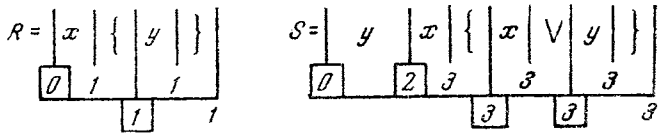
$$S = S_1\{x_1 \vee x_2 \vee \ldots \vee x_n\}.$$

В случае, если первоначально заданное выражение S для области запрета не удовлетворяло этому условию, его следует заменить выражением

$$R = S\{x_1 \vee x_2 \vee \ldots \vee x_n\}.$$

В качестве примера рассмотрим синтез частичного автомата Мили, представляющего событие $R = x\{y\}$ при наличии области запрета $S = yx\{x \vee y\}$.

На первом шаге проведем разметку мест, индексацию и распространение индексов в выражениях R и S, используя при этом возможность отождествления подобных мест:



Применяя второй и третий шаг алгоритма, придем к отмеченной таблице переходов автомата Мура A:

	R S
	0 1 2 * 3
x	1 * 3 * 3
y	2 1 * * 3

На четвертом шаге введем следующие переобозначения:
$0 \to 1$, $1 \to 2$, $2 \to 3$, $* \to 4$, $3 \to 5$, $(\ ) \to u$, $R \to v$, $S \to w$ (через $(\ )$

здесь обозначено пустое множество символов R и S). После этого получаем следующую отмеченную таблицу переходов:

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

Совершая на пятом шаге переход к автомату Мили, будем иметь следующие таблицы переходов и выходов:

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

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

Наконец, совершаем переход к частичному автомату. Запрещенным в нашем случае следует считать сигнал w. Тогда таблицы переходов и выходов предстанут в виде

	1	2	3	4	5
x	2	4	—	4	—
y	3	2	4	4	—

	1	2	3	4	5
x	*v*	*u*	—	*u*	—
y	*u*	*v*	*u*	*u*	—

Нетрудно заметить теперь, что состояние 5 является, в действительности, лишним, поскольку автомат никогда не перейдет в это состояние, отправляясь из начального состояния 1. Отбрасывая это лишнее состояние, придем к окончательным таблицам переходов и выходов:

	1	2	3	4
x	2	4	—	4
y	3	2	4	4

	1	2	3	4
x	*v*	*u*	—	*u*
y	*u*	*v*	*u*	*u*

§ 5. Минимизация абстрактных автоматов

Как известно, абстрактный автомат представляет собой устройство для реализации автоматных соответствий. В связи с этим естественно не различать автоматы, *эквивалентные* между собой, т. е. реализующие одинаковые автоматные соответствия.

Основной задачей, решаемой в настоящем параграфе, является задача минимизации автомата, т. е. , иначе говоря, задача нахождения автомата с минимальным числом состояний в классе всех автоматов, эквивалентных данному. Излагаемые методы решения этой задачи представляют собой развитие идей Мили [10], Ауфенкампа и Хона ([5] и [4]).

Пусть a и b — два состояния одного и того же или двух различных автоматов Мили, имеющих общий входной и общий выходной алфавиты. Если для любого входного сигнала x_i выходные сигналы, определяемые парами (a, x_i) и (b, x_i), одинаковы, то состояния a и b называются *1-эквивалентными*.

Если 1-эквивалентные состояния переводятся любым входным сигналом x_i также в 1-эквивалентные между собой состояния, то они называются *2-эквивалентными*. Если 2-эквивалентные состояния переводятся любым входным сигналом в 2-эквивалентные между собой состояния, то они называются *3-эквавалентными* и т. д.

Легко понять, что i-эквивалентные состояния при применении к ним любого входного слова длины i порождают одинаковые выходные слова ($i = 1,2,\ldots$).

Отношение i-эквивалентности для любого $i = 1,2,\ldots$ обладает свойствами рефлексивности, симметричности и транзитивности. Отсюда следует, что множество всех внутренних состояний данного автомата Мили разбивается этим отношением на попарно непересекающиеся классы i-эквивалентных между собой состояний. Такие классы мы будем называть *классами i-эквивалентности* или, сокращенно, просто *i-классами*.

Состояния, являющиеся i-эквивалентными для всех $i = 1,2,\ldots$, называются просто *эквивалентными состояниями*, а определяемые отношением эквивалентности классы — *классами эквивалентности* или *∞-классами*.

Из определения эквивалентных между собой состояний непосредственно вытекает справедливость следующего предложения:

5.1. *Два состояния одного и того же или двух различных автоматов Мили тогда и только тогда эквивалентны между собой, когда применение к ним любого входного слова вызывает появление одного и того же выходного слова.*

Из этого предложения следует:

5.2. *Два автомата Мили тогда и только тогда эквивалентны между собой (в смысле совпадения индуцируемых или автоматных соответствий), когда эквивалентны их начальные состояния.*

Легко видеть, что применение одного и того же входного слова q к двум эквивалентным состояниям a и b переводит их снова в эквивалентные состояния aq и bq. Поскольку же эквивалентные состояния являются вместе с тем и 1-эквивалентными, то для любого входного сигнала x_i пары (a, x_i) и (b, x_i) определяют одинаковые выходные сигналы.

В силу сказанного, для каждого автомата Мили A можно построить новый автомат Мили B с теми же, что у автомата A входным и выходным алфавитами, принимая за множество его внутренних состояний множество всех классов эквивалентности автомата A. Переходы и выходы в автомате B определяются так: класс эквивалентности K_1 переводится входным сигналом x_i в класс эквивалентности K_2, содержащий состояние $a_j x_i$, где a_j — любое состояние, содержащееся в классе K_1. Паре (K_1, x_i) сопоставляется при этом выходной сигнал, определяемый парой (a_j, x_i).

Построенный таким образом автомат Мили B будем называть *канонической минимизацией* автомата Мили A.

Из способа построения автомата B и из предложения 5.1 вытекает справедливость следующего предложения:

5.3. *В канонической минимизации любого автомата Мили любые два различных внутренних состояния не эквивалентны между собой.*

Для реализации автоматных соответствий достаточно ограничиться рассмотрением лишь так называемых связных автоматов, т. е. таких ав-

томатов, у которых каждое состояние является д о с т и ж и м ы м, или, иначе говоря, которые в результате воздействия подходящего входного слова могут быть переведены из начального состояния в любое другое внутреннее состояние.

В самом деле, если соответствие ξ реализуется несвязным автоматом A, то достижимые состояния автомата A образуют новый автомат B, который является связным и реализует то же самое соответствие ξ. Заметим, что в результате применения описанного в предыдущем параграфе алгоритма синтеза всегда получаются связные автоматы.

Рассмотрим какой-либо связный автомат Мили A, реализующий заданное автоматное соответствие ξ. Каноническая минимизация B автомата A также связна и реализует то же самое соответствие ξ (при выборе в качестве начального состояния класса эквивалентности K_0, содержащего начальное состояние автомата A).

Пусть D — произвольный автомат, реализующий то же самое соответствие, а d_0 — его начальное состояние. В силу связности автомата B для каждого его состояния K_i можно выбрать входное слово q_i так, чтобы $K_0 q_i = K_i$ $(i \in M)$. Построим отображение τ множества состояний автомата B в множество состояний автомата D, полагая $\tau(K_i) = d_0 q_i$ $(i \in M)$.

Ясно, что начальные состояния K_0 и d_0 автоматов B и D эквивалентны между собой. Но тогда эквивалентными являются также состояния K_i и $d_i = \tau(K_i)$ при любом $i \in M$. Если $\tau(K_i) = \tau(K_j)$, то отсюда вытекает эквивалентность состояний K_i и K_j. В силу же предложения 5.3 это означает, что $K_i = K_j$. Таким образом, соответствие τ взаимно однозначно, что влечет за собой справедливость следующего предложения:

5.4 *Каноническая минимизация связного автомата Мили A, реализующего любое заданное автоматное соответствие ξ, представляет собой автомат, имеющий наименьшее возможное число внутренних состояний среди всех автоматов Мили, реализующих то же самое соответствие ξ (или, что одно и то же, среди всех автоматов, эквивалентных автомату A).*

Теорема 5.4 полностью решает задачу минимизации автоматов Мили при условии, что имеется конструктивный прием построения классов эквивалентности для любого заданного (связного) автомата Мили. Такой прием был предложен в [5] для случая конечных автоматов Мили. Он основан на следующем легко доказываемом предложении:

5.5. *Если для некоторого i разбиение состояний автомата на $(i+1)$-классы совпадает с разбиением на i-классы, то оно является разбиением и на ∞-классы.*

В самом деле, если любая пара (a_k, a_j) i-эквивалентных состояний будет также $(i+1)$-эквивалентной, а состояния a_k и a_j любым входным сигналом x_r переводятся в i-эквивалентные между собой состояния, то они переводятся этим сигналом также и в $(i+1)$-эквивалентные между собой состояния. Следовательно, состояния a_k и a_j не только $(i+1)$-эквивалентны, но и $(i+2)$-эквивалентны между собой.

Продолжая подобным же образом, получим, что состояния a_j и a_k n-эквивалентны при всех $n = i, i+1, i+2, \ldots$ и, следовательно, являются эквивалентными между собой состояниями. Ввиду произвольности выбора состояний a_j и a_k предложение 5.5 доказано.

Алгоритм Луфенкампа — Хона построения классов эквивалентности (∞-классов) основан на последовательном построении i-классов для всех $i = 1,2,\ldots$ Поскольку разбиение на $(i+1)$-классы является подразбиением разбиения на i-классы, то в случае автомата через конечное число шагов мы получим на основании теоремы 5.5 искомое разбиение на ∞-классы.

Разбиение на 1-классы совершается непосредственно по таблице выходов автомата: в один и тот же 1-класс объединяются все состояния, которым соответствуют одинаковые столбцы в таблице выходов. Далее строится так называемая 1-таблица, получающаяся в результате замены в таблице переходов автомата внутренних состояний содержащими их 1-классами.

В один и тот же 2-класс объединяются все состояния, принадлежащие к одному и тому же 1-классу, которым соответствуют одинаковые столбцы в 1-таблице. Далее поступают аналогичным образом: заменяя в таблице переходов состояния автомата содержащими их 2-классами, приходят к 2-таблице. По 2-таблице находят 3-классы, объединяя в один 3-класс все состояния одного и того же 2-класса, которым соответствуют одиковые столбцы в 2-таблице, и т. д.

Придя через конечное число шагов к разбиению на ∞-классы, строят каноническую минимизацию исходного автомата A непосредственно по его таблицам переходов и выходов.

Таким образом мы построили алгоритм минимизации произвольных конечных автоматов Мили. Для случая автоматов Мура необходимо внести в этот алгоритм некоторые изменения, поскольку, интерпретируя автомат Мура как автомат Мили и минимизируя его в соответствии с описанным алгоритмом, мы построим автомат хотя и эквивалентный исходному, но не являющийся уже, вообще говоря, автоматом Мура.

Для того чтобы при минимизации автомат Мура оставался автоматом Мура, очевидно, необходимо и достаточно, чтобы одинаково отмеченные состояния автомата не относились бы к различным классам эквивалентности.

Наиболее просто удовлетворить этому условию, вводя для автоматов Мура понятие *0-эквивалентности* состояний и разбиения множества состояний на *0-классы*: 0-эквивалентными мы будем называть любые одинаково отмеченные состояния автомата Мура.

Если два 0-эквивалентных состояния любым входным сигналом переводятся в два снова 0-эквивалентных состояния, то они называются 1-эквивалентными.

Все дальнейшие построения (определение i-классов для $i \geqslant 2$, определение классов эквивалентности и построения канонической минимизации) проводятся точно так же, как и в случае автоматов Мили. Разумеется, в случае автоматов Мура при построении канонической минимизации B можно задавать для нее не таблицу выходов, а сдвинутую таблицу выходов, отмечая состояния автомата B (классы эквивалентности) теми же самыми выходными сигналами, которыми отмечены входящие в них состояния исходного автомата.

Следует отметить, однако, что теория минимизации автоматов Мура

в только что описанной форме не вполне эквивалентна соответствующей теории для случая автоматов Мили. В частности, предложение, аналогичное 5.1, для автоматов Мура, вообще говоря, не имеет места.

Для того чтобы сделать обе теории вполне параллельными, необходимо в качестве реакции автомата Мура на входное слово q рассматривать не то выходное слово p, которое получается в силу общего определения автомата, данного в § 1, а слово $y_i p$, где y_i — выходной сигнал, отмечающий то состояние, в котором находился автомат перед тем, как на него начали подавать слово q. При таком определении реакции автомата Мура на входное слово для автоматов Мура будут, очевидно, справедливы предложения, получающиеся из предложений 5.1, 5.2, 5.3, 5.4 заменой всех встречающихся в их формулировках автоматов Мили на автоматы Мура. Предложение 5.5 остается, разумеется, справедливым для автоматов Мура и при обычном определении выходных реакций автоматов.

При переходе к обычному пониманию выходных реакций автомата в особом положении оказываются, как нетрудно видеть, лишь такие состояния, в которые автомат не может перейти, отправляясь из других состояний (для случая связных автоматов таким свойством может обладать лишь начальное состояние). Легко проверить, что для восстановления параллелизма в теориях минимизации автоматов Мура и Мили в этом случае достаточно не отмечать подобные состояния никакими выходными сигналами. Это дает возможность при образовании 0-классов относить такие состояния к любому 0-классу и увеличивает тем самым возможности для минимизации.

Возникающий здесь параллелизм имеет, однако, место не для случая минимизации обычных, всюду определенных автоматов, а при переходе к более общей задаче минимизации частичных автоматов.

Сущность задачи минимизации частичных автоматов состоит в следующем: дан частичный автомат (Мура или Мили) A, реализующий частичное автоматное соответствие ξ, определенное на некотором множестве M слов входного алфавита. Требуется построить частичный автомат (Мура или, соответственно, Мили) B, который реализует частичное автоматное соответствие, совпадающее на множестве M с соответствием ξ, и имеет наименьшее число внутренних состояний среди всех автоматов (Мура или, соответственно, Мили), удовлетворяющих этому условию.

Поскольку имеется лишь конечное число различных частичных автоматов, имеющих общий входной и выходной алфавит с данным конечным частичным автоматом A и число состояний которых не превосходит числа состояний автомата A, то сформулированная задача алгоритмически разрешима. Однако существующие методы ее точного решения связаны с большим перебором (см. [11], [12]) и поэтому оказываются практически мало пригодными.

На практике обычно пользуются следующим приемом минимизации частичных автоматов: мысленно заполняя прочеркнутые места в таблицах переходов и выходов данного частичного автомата A, производят объединение состояний в i-классы и минимизацию по тем же самым правилам, что и для обычных (всюду определенных автоматов). Проверяют все или некоторые из возникающих вариантов объединения состояний в

классы и из полученных канонических минимизаций выбирают ту, которая имеет наименьшее число состояний.

Ясно, что указанный прием действительно решает задачу построения частичного автомата B с, вообще говоря, меньшим числом состояний, чем у исходного автомата A, причем реализуемое автоматом B частичное автоматное соответствие τ совпадает с частичным автоматным соответствием σ, реализуемым автоматом A, на области определения соответствия σ. При этом область определения соответствия τ, вообще говоря, больше, чем область определения соответствия σ.

Покажем, как работает описанный прием минимизации на примере частичного автомата Мили A, заданного следующими таблицами переходов и выходов:

	1	2	3	4
x	2	—	2	4
y	—	3	4	4

	1	2	3	4
x	u	—	u	u
y	—	u	v	v

Производя минимизацию заданного автомата, имеем две исходные возможности объединения в 1-классы, приводящие к наименьшему количеству классов: $a_1 = (1,3,4)$, $b_1 = (2)$ и $a_1 = (1,2)$, $b_1 = (3,4)$. Рассмотрим сначала первую возможность: 1-таблица запишется в виде

	1	2	3	4
x	b_1	—	b_1	a_1
y	—	a_1	a_1	a_1

Из этой таблицы видно, что класс a_1 нужно разделить на два 2-класса: $a_2 = (1,3)$ и $c_2 = (4)$, третий 2-класс b_2 совпадает с 1-классом $b_1 = (2)$: 2-таблица будет иметь вид

	1	2	3	4
x	b_2	—	b_2	c_2
y	—	a_2	c_2	c_2

Из этой таблицы видно, что 3-классы совпадают с 2-классами, которые являются искомыми ∞-классами. Каноническая минимизация в разобранном варианте представится следующими таблицами:

	a_2	b_2	c_2
x	b_2	—	c_2
y	c_2	a_2	a_2

	a_2	b_2	c_2
x	u	—	u
y	v	u	v

Во втором варианте минимизации разбиение на 1-классы $a_1 = (1,2)$, $b_1 = (3,4)$ приводит к 1-таблице

	1	2	3	4
x	a_1	—	a_1	b_1
y	—	b_1	b_1	b_1

и к разбиению на 2-классы: $a_2 = (1,2)$, $b_2 = (3)$, $c_2 = (4)$. 2-таблица запишется в виде

	1	2	3	4
x	a_2	—	a_2	c_2
y	—	b_2	c_2	c_2

Из этой таблицы следует, что полученное решение на 2-классы дальше не измельчается и, следовательно, совпадает с разбиением на ∞-классы. Каноническая минимизация в этом варианте будет задаваться таблицами

	a_2	b_2	c_2
x	a_2	a_2	c_2
y	b_2	c_2	c_2

	a_2	b_2	c_2
x	u	u	u
y	u	v	v

Нетрудно видеть, что полученные канонические минимизации существенно различны: первая на входное слово y реагирует выходным словом v, а вторая — выходным словом u.

§ 6. Структурный синтез конечных автоматов

Цель настоящего параграфа состоит в том, чтобы наметить пути решения основной задачи четвертого из отмеченных во введении этапов синтеза и дать четкую постановку задач для пятого этапа.

На этапе структурного синтеза производится выбор элементарных автоматов, из которых осуществляется синтез структурной схемы заданного абстрактного автомата Мили или Мура. Мы будем предполагать, что эти элементарные автоматы бывают двух родов: *элементарные автоматы с памятью*, или *запоминающие элементы* (триггеры, линии задержки и т. п.), и *элементарные автоматы без памяти*, называемые также *логическими элементами*. Для простоты ограничимся случаем, когда в нашем распоряжении имеется лишь один тип запоминающих элементов.

Входные и выходные сигналы как элементарных автоматов, так и всего рассматриваемого автомата в целом обозначаются (кодируются) конечными последовательностями букв некоего фиксированного конечного алфавита, называемого *структурным алфавитом*. Обычно в качестве структурного алфавита берется так называемый *двоичный алфавит*, состоящий из двух букв: 0 и 1. Вторым алфавитом, играющим важнейшую роль на этапе структурного синтеза, является алфавит, состоящий из символов внутренних состояний выбранных элементов памяти; назовем его *алфавитом состояний*. Алфавит состояний не обязан, вообще говоря, совпадать со структурным алфавитом, однако на практике в качестве алфавита состояний выбирается обычно также двоичный алфавит. Как уже отмечалось во введении, одной из главных задач, решаемых в процессе структурного синтеза автоматов, является выписывание канонических уравнений, устанавливающих зависимость сигналов, подаваемых на входы запоминающих элементов, от выходных сигналов этих элементов и сигнала, подаваемого на вход всего автомата в целом. Для того чтобы обеспечить правильное функционирование схемы, нельзя допускать, чтобы входные сигналы, подаваемые на входы запоминающих элементов,

участвовали непосредственно в образовании выходных сигналов, которые по цепям обратной связи подавались бы в тот же самый момент времени на эти входы. Иначе говоря, запоминающие элементы с обратной точки зрения должны представлять собой автоматы Мура, а не Мили. Сложный же автомат, образованный этими элементами, может, разумеется, быть при этом как автоматом Мура, так и автоматом Мили.

Итак, пусть используемые при структурном синтезе элементарные автоматы с памятью являются автоматами Мура. Сдвинув соответственно начало отсчета временных интервалов для выходных сигналов, мы будем считать, что выходной сигнал в любой момент времени t каждого элемента памяти определяется внутренним состоянием этого элемента в т о т ж е с а м ы й момент времени.

Далее, для того чтобы иметь возможность синтезировать произвольные автоматы при минимальной затрате элементов памяти, целесообразно в качестве таких элементов выбирать автоматы Мура, обладающие *полной системой переходов и полной системой выходов*, которые мы для краткости будем называть просто *полными автоматами*. Полнота системы переходов означает, что для любой пары состояний автомата найдется входной сигнал, переводящий первый элемент этой пары во второй. Это требование эквивалентно тому, чтобы в каждом столбце таблицы переходов встречались все состояния данного автомата. Полнота системы выходов в случае автоматов Мура означает, что каждому состоянию автомата поставлен в соответствие свой особый, отличный от других состояний, выходной сигнал.

Поэтому для полных автоматов Мура естественно просто отождествлять выходные сигналы с соответствующими внутренними состояниями автомата. Мы будем в дальнейшем изложении придерживаться такого способа.

Итак, выберем в качестве элемента памяти некоторый полный автомат Мура. Внутренние состояния этого автомата будем обозначать через $z_1, z_2, \ldots, z_r$ $(r \geqslant 2)$, они же, согласно принятому условию, будут и выходными сигналами. Для обозначения входных сигналов элемента памяти будем употреблять буквы $s_1, s_2, \ldots, s_q$. Алфавит $(z_1, z_2, \ldots, z_r)$ есть не что иное, как алфавит состояний. Что же касается структурного алфавита, то мы его выберем несколько позднее, а пока покажем, как находить канонические уравнения автомата, память которого составляется из элементов выбранного типа.

Предположим, что нам задан абстрактный конечный автомат Мили или Мура A с входным алфавитом $\mathfrak{X} = (x_1, \ldots, x_n)$, выходным алфавитом $\mathfrak{Y} = (y_1, \ldots, y_m)$ и множеством внутренних состояний $\mathfrak{A} = (a_1, \ldots, a_p)$, задаваемый функцией переходов $\varphi(a, x)$ и функцией выходов $\psi(a, x)$. Пусть выбранный элемент памяти B задается функцией переходов $\lambda(z, s)$. Поставим задачу нахождения канонических уравнений автомата A при условии, что его память строится из нескольких экземпляров $B^{(1)}, \ldots, B^{(k)}$ элементарного автомата B.

Ясно, что для построения автомата A число k элементов памяти должно удовлетворять условию: $r^k \geqslant p$. В этом случае различные внутренние состояния a_i можно отождествить с различными наборами состояний авто-

матов $B^{(1)}, \ldots, B^{(k)}$. Процесс такого отождествления будем называть *кодированием* состояний автомата A в принятом алфавите состояний. Разумеется, процесс кодирования по самому своему существу неоднозначен. Однако мы не будем входить в подробности вопроса, связанного с выбором того или иного варианта кодирования, а будем просто считать этот вариант уже заданным.

После кодирования состояния автомата A будут обозначены k-мерными векторами $(z^{(1)}, \ldots, z^{(k)})$, компонентами которых служат различные буквы $z_1, \ldots, z_p$ алфавита состояний; двуместная функция выходов $\psi(a, x)$ автомата A превратится в $(k+1)$-местную функцию $\psi_0(z^{(1)}, \ldots, z^{(k)}, x)$, которую мы по-прежнему будем называть функцией выходов. Что же касается функции переходов $\varphi(a, x)$, то ее эквивалентом после кодирования будет система k $(k+1)$-местных функций

$$\varphi^{(1)}(z^{(1)}, \ldots, z^{(k)}, x), \ldots, \varphi^{(k)}(z^{(1)}, \ldots, z^{(k)}, x)$$

переходов в элементах памяти. Функция $\varphi^{(i)}$ определяет состояние, в которое должен перейти i-й элемент памяти в момент времени $t+1$, если в момент времени t автомат A находился в состоянии $(z^{(1)}, \ldots, z^{(k)})$, а на его вход был подан сигнал x ($i = 1, 2, \ldots, k$, $t = 0, 1, 2, \ldots$).

Следующим важным этапом является построение функций возбуждения

$$\sigma^{(i)}(z^{(1)}, \ldots, z^{(k)}, x)$$

элементов памяти ($i = 1, \ldots, k$). Значение каждой такой функции $\sigma^{(i)}$ при выбранном состоянии $(z^{(1)}, \ldots, z^{(k)})$ автомата A и входном сигнале x определяется как входной сигнал $s^{(i)}$ i-го элемента памяти, вызывающий переход в i-м элементе памяти, обусловленный i-й функцией переходов, иначе говоря, переход

$$z^{(i)} \to \varphi^{(i)}(z^{(1)}, \ldots z^{(k)}, x) \quad (i = 1, \ldots, k).$$

Выбор входного сигнала $s^{(i)}$ может быть произведен, вообще говоря, не одним способом. Поэтому построение функций возбуждения элементов памяти по функциям переходов в них осуществляется неоднозначно.

Выбор наилучшего способа построения функций возбуждения связан с задачами последующего этапа — этапа комбинационного синтеза и рассматриваться здесь на будет. Впрочем, для многих типов элементов памяти (линии задержки, триггеры со счетным входом и др.) соответствующий переход выполняется однозначно. Для ряда типов элементов можно указать частные комбинаторно-вычислительные приемы, позволяющие более простым по сравнению с только что изложенным общим приемом способом находить функции возбуждения (см. например, [2]).

Функции возбуждения $\varphi^{(i)}$, будучи приравнены определяемым ими входным сигналом $\sigma^{(i)}$, дают искомые канонические уравнения для обратных связей в автомате A. Однако в одном отношении эти функции имеют еще не вполне удовлетворительный вид. А именно в то время, как состояния автомата A закодированы в универсальном (для выбранного типа элементов памяти) алфавите состояний, не зависящем от выбора автомата A, для обозначения входных и выходных

сигналов используются различные алфавиты, в том числе и такие, которые зависят от выбора автомата A.

Необходимо поэтому фиксировать структурный алфавит Ω, определяемый обычно фактически принятым кодированием входных сигналов элементов памяти, и закодировать конечными последовательностями букв этого алфавита не только входные сигналы $\sigma^{(i)}$ элементов памяти, но также входные и выходные сигналы x и y всего автомата в целом. Такое кодирование преобразует найденную выше систему функций возбуждения $\sigma^{(i)}$ в новую систему функций

$$\sigma_j^{(i)}\ (z^{(1)}, \ldots, z^{(k)}, u_1, u_2, \ldots, u_g)\quad (i = 1, \ldots, k,\ j = 1, 2, \ldots, l),$$

где каждая из $\sigma_j^{(i)}$ представляет собой входной сигнал (букву структурного алфавита), который нужно подать на j-й входной канал i-го элемента памяти в то время, когда автомат A находится в состоянии $(z^{(1)}, \ldots, z^{(k)})$, а на его входные каналы подаются сигналы (буквы структурного алфавита) $u_1, u_2, \ldots, u_g$.

Аналогичным образом найденная выше функция выходов

$$\psi_0\ (z^{(1)}, \ldots, z^{(k)}, x)$$

заменится системой функций

$$\psi_j\ (z^{(1)}, \ldots, z^{(k)}, u_1, u_2, \ldots, u_g)\quad (j = 1, 2, \ldots, h),$$

где ψ_j определяет выходной сигнал (букву структурного алфавита), появляющийся на j-м выходном канале автомата A в то время, когда автомат A находится в состоянии $(z^{(1)}, \ldots, z^{(k)})$, а на его входные каналы подаются сигналы $u_1, u_2, \ldots, u_g$.

Полученные функции $\sigma_j^{(i)}$ и ψ_j будем называть, соответственно, *структурными функциями возбуждений* и *структурными функциями выходов* автомата A.

В случае (обычно встречающемся на практике), когда как структурный алфавит, так и алфавит состояний являются двоичными алфавитами, структурные функции возбуждений и выходов можно рассматривать как обычные булевы функции. Задача дальнейшего этапа (этапа комбинационного синтеза) заключается в фактическом построении найденных функций из элементарных логических функций, реализуемых выбранными л о г и ч е с к и м и э л е м е н т а м и. Методы решения этой задачи в настоящее время достаточно далеко и хорошо известны специалистам, работающим в области синтеза дискретных автоматов. Мы не будем здесь излагать эти методы, отсылая желающих детально ознакомиться с вопросом к другим работам, например к статье [6].

В качестве элементов памяти в большинстве современных цифровых автоматов употребляются полные автоматы Мура с двумя внутренними состояниями. Интересно проанализировать вопрос о том, сколько и какие элементарные автоматы удовлетворяют этим свойствам. Рассмотрим случай, когда полный автомат Мура с двумя состояниями 0 и 1 имеет только два входных сигнала x и y. Из условия полноты следует, что в каждом столбце таблицы переходов автомата должны встречаться оба

состояния 0 и 1. Это ограничение приводит к тому, что существуют всего 4 возможные таблицы переходов:

	0	1
x	0	0
y	1	1

	0	1
x	0	1
y	1	0

	0	1
x	1	1
y	0	0

	0	1
x	1	0
y	0	1

Третья таблица совпадает с первой после переобозначения входных сигналов. То же самое имеет место по отношению ко второй и четвертой таблице. Таким образом, имеется лишь два существенно различных автомата требуемого вида, задаваемых таблицами переходов:

	0	1
x	0	0
y	1	1

	0	1
x	0	1
y	1	0

Если положить $x = 0$, $y = 1$, то первая таблица задает хорошо известный элемент памяти, называемый элементом *задержки* (на один такт), а вторая — не менее известный элемент, называемый *триггером со счетным входом*. Заметим, что обычное электромагнитное реле с замыкающим контактом может рассматриваться как элемент задержки.

При увеличении числа входных сигналов появляются новые типы элементов памяти: *триггер с раздельными входами*, задаваемый таблицей переходов

	0	1
x	0	1
y	0	0
z	1	1

смешанный триггер, задаваемый таблицей

	0	1
x	0	1
y	0	0
z	1	1
u	1	0

и др.

Заметим, что при наличии лишь двух внутренних состояний бесполезно увеличивать число входных сигналов более чем до четырех, поскольку при большем числе входных сигналов некоторые из них начнут дублировать друг друга (вызывать одинаковые переходы в автомате).

Нетрудно поэтому составить каталог всех полных существенно различных автоматов Мура с двумя состояниями (существенно различными мы будем считать автоматы, таблицы переходов которых не переходят одна в другую при переобозначениях входных сигналов).

Кроме уже перечисленных четырех типов автоматов, в этот каталог войдут еще три автомата, задаваемых таблицами переходов:

	0	1
x	0	1
y	0	0
z	1	0

	0	1
x	0	1
y	1	1
z	1	0

	0	1
x	0	0
y	1	1
z	1	0

Разумеется, каждый из перечисленных семи типов автоматов допускает различные модификации за счет различного кодирования входных сигналов в двоичном алфавите.

Рассмотрим теперь в качестве примера полный синтез (абстрактный и структурный до получения канонических уравнений) автомата Мура A, представляющего собой последовательный двухзарядный двоичный квадратор. Автомат A работает следующим образом: на его вход разряд за разрядом младшими разрядами вперед подается двухразрядное двоичное целое число. На выходе автомата при этом также последовательно, начиная с младших разрядов, должен появиться квадрат этого числа. Иначе говоря, автомат A должен реализовать следующее частичное соответствие ξ:

$$0000 \to 0000$$
$$1000 \to 1000$$
$$0100 \to 0010$$
$$1100 \to 1001$$

Нетрудно проверить, что соответствие ξ, будучи продолжено на начальные отрезки слов, удовлетворяет условию автоматности. Обозначая нулевой сигнал на входе через x, а на выходе через u и, соответственно, единичный сигнал на входе через y, а на выходе через v, запишем это соответствие в виде:

$$xxxx \to uuuu$$
$$yxxx \to vuuu$$
$$xyxx \to uuvu$$
$$yyxx \to vuuv$$

В полученном соответствии выходным сигналом u представлено событие

$$R = x \vee xx \vee xxx \vee xxxx \vee yx \vee yxx \vee yxxx \vee xy \vee xyxx \vee yy \vee yyx,$$

а выходным сигналом v — событие

$$Q = xyx \vee y \vee yyxx.$$

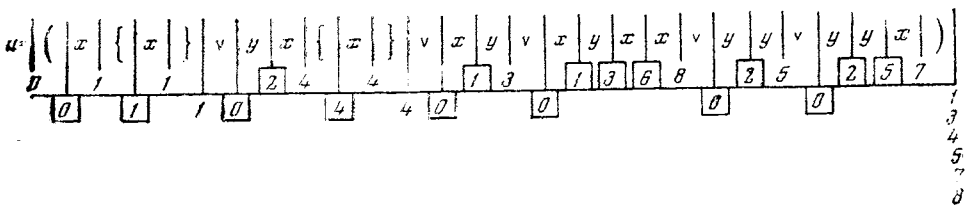
Слова, не вошедшие в события R и Q, являются запрещенными. Ясно, что за счет запрещенных слов можно увеличивать эти события без опасений нарушить в синтезируемом автомате его реакцию на заданные входные слова.

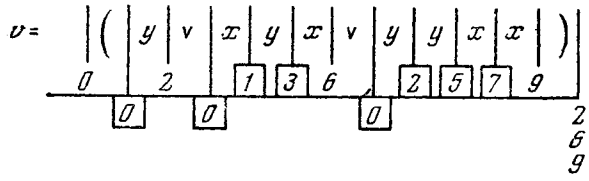
Это обстоятельство позволяет заменить событие R новым событием, которое мы обозначим соответствующим ему выходным сигналом u, имеющим более короткую регулярную запись:

$$u = (x\{x\} \vee yx\{x\} \vee xy \vee xyxx \vee yy \vee yyx).$$

Событие Q также обозначим соответствующим ему выходным сигналом v.

Произведем разметку мест в событиях u и v и синтез представляющего их абстрактного автомата Мура:





		u	*v*	*u*	*u*	*u*	*v*	*u*	*u*	*v*
	0	1	2	3	4	5	6	7	8	9
x	1	1	4	6	4	7	8	9	—	—
y	2	3	5	—	—	—	—	—	—	—

Состояние, соответствующее пустому множеству символов, является, очевидно, запрещенным состоянием (автомат попадает в него лишь в результате воздействия запрещенных слов). Поэтому оно может быть заменено любым другим состоянием, в силу чего мы обозначаем его черточкой и не выписываем соответствующий ему столбец в отмеченной таблице переходов автомата.

Полученный частичный автомат Мура допускает следующее разбиение на 0-классы: $a_0 = (1, 3, 4, 5, 7, 8)$; $b_0 = (0, 2, 6, 9)$. Этому разбиению соответствует 0-таблица:

	0	1	2	3	4	5	6	7	8	9
	b_0	a_0	b_0	a_0	a_0	a_0	b_0	a_0	a_0	b_0
x	a_0	a_0	a_0	b_0	a_0	a_0	a_0	b_0	—	—
y	b_0	a_0	a_0	—	—	—	—	—	—	—

По 0-таблице получаем 1-классы: $a_1 = (1, 4, 5, 8)$, $b_1 = (2, 6, 9)$, $c_1 = (3, 7)$, $d_1 = (0)$ — и 1-таблицу:

	0	1	2	3	4	5	6	7	8	9
	d_1	a_1	b_1	c_1	a_1	a_1	b_1	c_1	a_1	b_1
x	a_1	a_1	a_1	b_1	a_1	c_1	a_1	b_1	—	—
y	b_1	c_1	a_1	—	—	—	—	—	—	—

Строим 2-классы: $a_2 = (1, 4, 8)$, $b_2 = (2, 6, 9)$, $c_2 = (3, 7)$, $d_2 = (0)$, $e_2 = (5)$ — и 2-таблицу:

	0	1	2	3	4	5	6	7	8	9
	d_2	a_2	b_2	c_2	a_2	e_2	b_2	c_2	a_2	b_2
x	a_2	a_2	a_2	b_2	a_2	c_2	a_2	b_2	—	—
y	b_2	c_2	e_2	—	—	—	—	—	—	—

Так как 3-классы совпадают с 2-классами, то минимизация окончена. Искомый автомат A может быть задан, очевидно, следующей таблицей:

	u	*v*	*u*	*u*
	1	2	3	4	5
x	2	2	2	3	4
y	3	4	5	—	—

Переходя к структурному синтезу, выберем в качестве элементов памяти элементы задержки, имеющие таблицу переходов

	0	1
0	0	0
1	1	1

Так как синтезируемый автомат имеет 5 внутренних состояний, а элемент памяти 2 состояния, то нужно выбрать 3 элемента памяти ($2^3 \geqslant 5$). Обозначим их внутренние состояния через z_1, z_2, z_3, а входные сигналы — через s_1, s_2, s_3 (переменные z_i и s_i принимают значения 0 и 1). Состояния автомата A обозначаются векторами (z_1, z_2, z_3).

Выберем следующую систему кодирования состояний: $1 = (0, 0, 0)$; $2 = (0, 0, 1)$; $3 = (0, 1, 0)$; $4 = (0, 1, 1)$; $5 = (1, 0, 0)$. Обозначим переменную, принимающую значения 0 и 1 в соответствии с входным сигналом автомата, через x и перепишем таблицу переходов автомата A в соответствии с принятой системой кодирования (такую таблицу называют обычно *физической таблицей переходов* автомата A, а таблицу переходов в исходной форме — *абстрактной таблицей переходов*):

z / x		$u=0$	$u=1$	$u=0$	$u=0$
	000	001	010	011	100
0	001	001	010	010	011
1	010	011	100	— — —	— — —

Эта таблица задает состояния $z'_1 = z_1(t+1)$, $z'_2 = z_2(t+1)$, $z'_3 = z_3(t+1)$ элементов памяти как булевы функции их состояний $z_1(t) = z_1$, $z_2 = z_2(t)$, $z_3 = z_3(t)$ и входного сигнала $x = x(t)$ автомата A в настоящий момент времени. Из таблицы переходов элемента задержки видно, что его состояние в любой последующий момент времени $t+1$ совпадает с сигналом на его входе в настоящий момент времени. Можно поэтому считать, что выписанная таблица задает структурные функции возбуждений искомого автомата A: $s_i = \sigma_i(z_1, z_2, z_3, x)$ $(i = 1, 2, 3)$.

Выпишем таблицу, определяющую эти функции, в том виде, как обычно принято в теории булевых функций:

z_1	z_2	z_3	x	σ_1	σ_2	σ_3
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	0	1
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	—	—	—
1	0	0	0	0	1	1
1	0	0	1	—	—	—
1	0	1	0	—	—	—
1	0	1	1	—	—	—
1	1	0	0	—	—	—
1	1	0	1	—	—	—
1	1	1	0	—	—	—
1	1	1	1	—	—	—

Во всех местах этой таблицы, в которых стоят прочерки, функции не определены, поскольку соответствующие наборы значений переменных не могут встретиться при работе автомата. Эти значения выбираются так, чтобы обеспечить минимальные затраты логических элементов в процессе комбинационного синтеза схемы (без памяти), реализующей найденные канонические уравнения.

Структурная функция выходов q построенного автомата также определяется как булева функция переменных z_1, z_2, z_3 из отмеченной физической таблицы переходов автомата A. Подобно функциям возбуждения, она также определена лишь на части наборов. Приведем соответствующую (неполную) таблицу.

z_1	z_2	z_3	q
0	0	0	–
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	

Найденными функциональными зависимостями $s_i = \sigma_i(z_1, z_2, z_3, x)$ $(i = 1, 2, 3)$ и $q = q(z_1, z_2, z_3)$ решается задача четвертого из отмеченных во введении этапов синтеза. На следующем (пятом) этапе нужно построить схему (без памяти), фактически реализующую заданные зависимости. Методов решения этой задачи мы в данной статье касаться не будем. Отметим лишь, что, используя карты Карнау, легко найти следующие минимальные дизъюнктивные нормальные формы для полученных функций:

$$\sigma_1 = z_2 x, \quad \sigma_2 = z_1 \bar{z}_2 \vee z_2 z_3 \vee \bar{z}_2 x, \quad \sigma_3 = \bar{z}_2 z_3 \vee \bar{z}_3 \bar{x}, \quad q = z_2 \bar{z}_3.$$

В заключение сделаем краткий обзор источников и рассмотрим содержание статьи в историческом аспекте.

Основные понятия, изложенные в § 1, ведут свое начало от работ [13], [10] и [14]. Понятие автоматного соответствия под различными названиями также неоднократно встречалось в различных работах (см., например, [15], [16] и др.). Метод приведения произвольного соответствия к автоматному принадлежит автору.

Понятие о событии и его представлении в автомате было впервые введено в [3], где использовались автоматы частного вида (нервные сети). В [3] введено также понятие регулярного события, однако использованный там для описаний язык (регулярных множеств таблиц) весьма громоздок и неудобен для практического использования. Дальнейшие усовершенствования этого языка произведены в [17], [8] и [7].

Приведенный в § 2 способ построения примеров нерегулярных событий развивает идею, приведенную в [3]. Вопрос о взаимоотношении автоматов Мили и Мура с точки зрения представления событий в достаточно четкой форме поставлен в настоящей статье, по-видимому, впервые. Однако с несколько иных позиций этот вопрос рассматривался и ранее (см., например, [15], теорема 1).

§ 3 и 4 излагают результаты работ автора [9] и [7]. Заметим, что в работе [9] алгоритм анализа автоматов был приведен без доказательства. Его доказательство, проведенное в § 3, публикуется впервые. Отметим

также, что описываемый в § 4 алгоритм синтеза автоматов содержит ряд усовершенствований по сравнению с алгоритмом, изложенным в [7]. Наиболее важным усовершенствованием является обобщение алгоритма на случай синтеза автоматов при наличии областей запрета. Исправлена неточность, допущенная в [7] при определении подобных мест. Значительно изменены доказательства.

Методы минимизации абстрактных автоматов, изложенные в § 5, ведут свое начало от работы [10] и учитывают усовершенствования, введенные в [4] и [5] (см. также [18], [19] и [20]). В отличие от этих работ, где рассматривался вопрос о минимизации лишь для автоматов Мили, в настоящей статье разбирается минимизация обоих типов автоматов (Мили и Мура), а также систематизирована и упрощена символика.

§ 6 представляет собой первую попытку изложения структурной теории автоматов с точки зрения абстрактной теории. К числу введенных в нем новых понятий относится, в частности, понятие полного автомата. Что же касается канонических уравнений, то методам их построения применительно к тем или иным конкретным видам элементов памяти было посвящено ранее значительное число работ. Укажем в качестве примера на работы [2] и [1].

Проблем комбинационного синтеза, как уже отмечалось выше, настоящая статья совсем не затрагивает.

Отметим еще два направления, тесно связанных с темой настоящей статьи, но не вошедших в нее по причине недостатка места. Первое направление связано с методами нахождения абсолютных минимизаций частичных автоматов путем упорядоченного перебора всех автоматов, реализующих данное соответствие. Оно развивалось в работах [11], [12].

Второе направление связано с разработкой методов непосредственного построения канонических уравнений на основе задания работы схемы на языке одноместного исчисления предикатов. Ему посвящены работы [21], [22].

Следует указать, что методы непосредственного построения структурной схемы автомата применительно к элементам того или иного специального вида (как правило, к элементам, реализующим функции двузначной логики) развивались и другими авторами (см., например, [8]). Однако ввиду того, что вопросы минимизации логических схем с памятью разработаны крайне плохо, минимизация числа состояний при использовании этих методов весьма затруднительна.

Поступила в редакцию
1.02.1961

Цитированная литература

1. М. Л. Цетлин. О непримитивных схемах. Пробл. кибернетики, 1958, вып. 1, 23—45.
2. В. М. Шестаков. Алгебраический метод синтеза многотактных релейных систем. Докл. АН СССР, 1954, 99, № 6, 987—990.
3. С. К. Клини. Представление событий в нервных сетях и конечных автоматах. В сб. «Автоматы». М., Изд-во ин. лит., 1956, 15—67.

4. D. D. Aufenkamp. Analysis of sequential machines II. IRE Trans., 1958, EC-7, № 4, 299—306.
5. D. D. Aufenkamp, F. S. Hohn. Analysis of sequential machines. IRE Trans., 1957, EC-6, № 4, 276—285.
6. С. В. Яблонский. Функциональные построения в k-значной логике. Тр. Матем. ин-та. АН СССР, 1958, 51, 5—142.
7. В. М. Глушков. Об одном алгоритме синтеза абстрактных автоматов. Укр. матем. ж., 1960, 12, № 2, 147—156.
8. I. M. Copi, C. Elgot, J. B. Wright. Realization of events by logical nets. J. Assoc. Comput. Machinery., 1958, 5, № 2, 181—196.
9. В. М. Глушков. Об одном методе анализа абстрактных автоматов. Докл. АН УССР, 1960, № 9, 1151—1154.
10. G. H. Mealy. A method for synthesizing sequential circuits. Bell. System. Techn. J., 1955, 34, 1045—1079.
11. S. Ginsburg. A synthesis technique for minimal-state sequential machine. IRE Trans., 1959, EC-8, № 1, 13—24.
12. S. Ginsburg. Synthesis of minimal-state machines. IRE Trans., 1959, EC-8, № 4, 441—448.
13. D. A. Huffman. The synthesis of sequential switching circuits. J. Franklin Inst., 1954, 257, № 3, 161—190. № 4, 275—303.
14. Э. Ф. Мур. Умозрительные эксперименты с последовательностными машинами. В сб. «Автоматы». М., Изд-во ин. лит., 1956, 179—212.
15. А. Ш. Блох. О задачах, решаемых последовательностными машинами. Пробл. кибернетики, 1960, вып. 3, 81—88.
16. Н. Е. Кобринский, Б. А. Трахтенброт. О построении общей теории логических сетей. В сб. «Логические исследования». М., Изд-во АН СССР, 1959, 352—378.
17. Ю. Т. Медведев. О классе событий, допускающих представление в конечном автомате. В сб. «Автоматы». М., Изд-во ин. лит., 1956, 385—401.
18. S. Ginsburg. A technique for the reduction of a given machine to a minimal-state machine. IRE Trans., 1959, EC-8, № 3, 346—355.
19. D. S. Netherwood. Minimal sequential machines. IRE Trans., 1959, EC-8, № 3, 339—345.
20. M. C. Paull, S. H. Unger. Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans., 1959, EC-8, № 3, 356—367.
21. Б. А. Трахтенброт. Об операторах, реализуемых в логических сетях. Докл. АН СССР, 1957, 112, № 6, 1005—1007.
22. Б. А. Трахтенброт. Синтез логических сетей, операторы которых описаны средствами исчисления одноместных предикатов. Докл. АН СССР, 1958, 118, № 4, 646—649.