МАТЕМАТИЧЕСКАЯ ЛОГИКА
И
ОСНОВАНИЯ МАТЕМАТИКИ

ГОСУДАРСТВЕННОЕ ИЗДАТЕЛЬСТВО
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
МОСКВА 1962

В. М. ГЛУШКОВ

СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

ГОСУДАРСТВЕННОЕ ИЗДАТЕЛЬСТВО
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
МОСКВА 1962

Виктор Михайлович Глушков
Синтез цифровых автоматов
М., Физматгиз, 1962 г., 476 стр. с илл.
(серия «Математическая логика и основания математики»)

Редактор *Б. В. Бирюков*

Техн. редактор *К. Ф. Брудно* Корректор *Л. О. Сечейко.*

Сдано в набор 31/V 1962 г. Подписано к печати 28/IX 1962 г. Бумага 84×108/32. Физ. печ. л. 14,875. Условн. печ. л. 24,4. Уч.-изд. л. 25,17. Тираж 15 000 экз. Т-10950. Цена книги 1 р. 41 к. Заказ № 3061.

Государственное издательство физико-математической литературы
Москва, В-71, Ленинский проспект, 15

Отпечатано в гос. типографии «Пяргале», Вильнюс, ул. Латако 6
Заказ № 2266

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Одним из значительных достижений науки и техники середины двадцатого столетия явилось создание и широкое использование электронных цифровых машин с программным управлением. Неудивительно, что вопросы рационального конструирования, или, как принято обычно говорить, синтеза схем таких машин привлекают внимание большого числа ученых, конструкторов и инженеров.

Электронные цифровые машины с программным управлением представляют собою пример одного из наиболее распространенных в настоящее время типов преобразователей дискретной информации, называемых *дискретными*, или *цифровыми*, *автоматами*. Поэтому задача синтеза схем электронных цифровых машин с программным управлением входит в качестве частного случая в более общую задачу синтеза схем цифровых автоматов. Важность этой задачи обусловливается тем, что в настоящее время в целом ряде областей происходит процесс быстрого вытеснения старых средств автоматики, основанных на принципе непрерывности, дискретными автоматическими устройствами.

Процесс синтеза сложного автомата подразделяется на несколько этапов. На первом этапе, называемом этапом *блочного синтеза*, осуществляется разбиение схемы автомата на отдельные блоки, определяются задачи, которые

должны решаться отдельными блоками, намечается общий план обмена информаций между блоками.

На втором этапе, который мы будем называть этапом *абстрактного синтеза*, на основании задач, решаемых каждым отдельным блоком, определяется объем памяти, необходимый для данного блока, и устанавливаются те изменения состояний памяти под воздействием входных сигналов, которые должны реализоваться данным блоком для того, чтобы он мог выполнять поставленные перед ним задачи.

На третьем этапе, называемом этапом *структурного синтеза*, осуществляется выбор логических и запоминающих элементов для построения схемы данного блока — и автомата в целом — и выписываются так называемые *канонические уравнения*, сводящие общую задачу синтеза автомата (или блока) к синтезу схем из элементов дискретного действия, не обладающих памятью.

Следующим, четвертым этапом является этап синтеза этих последних схем, называемый обычно этапом *комбинационного синтеза*.

Наконец, на пятом и последнем этапе производится преобразование и дополнение построенных схем с целью обеспечения надежности их функционирования. На этом этапе, который мы будем условно называть этапом *надежностного синтеза*, выявляются искажения сигналов, возникающие вследствие неидеальности применяемых элементов, и принимаются меры к устранению этих искажений.

Разумеется, указанное разделение на этапы дает лишь общую ориентировку в вопросе о том, через какие стадии проходит решение задачи синтеза цифровых автоматов. В ряде случаев приходится делать те или иные отступления от приведенной выше последовательности этапов. Например, при синтезе автоматов, обладающих относительно малой сложностью, этап блочного синтеза обычно опус-

кается. Наоборот, при синтезе особо сложных автоматов возможно многократное возвращение к этому этапу. При некоторых специальных приемах синтеза этапы абстрактного, структурного и комбинационного синтеза так переплетаются между собой, что не всегда удается провести четкое разграничение между ними. Наконец, учет соображений надежности обычно начинается на более ранних этапах, находя на последнем этапе окончательное решение.

Исторически наиболее рано начала развиваться та часть теории синтеза автоматов, которая имеет дело с этапом комбинационного синтеза. В работах В. И. Шестакова [1]) и К. Шеннона [2]) была впервые продемонстрирована плодотворность идеи применения развитого ранее в рамках математической логики аппарата так называемой *булевой алгебры* к проблемам комбинационного синтеза релейно-контактных схем.

Развиваясь в рамках логико-математической теории релейно-контактных схем, теория комбинационного синтеза достигла значительных успехов и после появления электронных цифровых машин стала успешно приспосабливаться к проблемам синтеза схем из электронных логических элементов. В этот период она пополнилась также существенными фрагментами теории структурного и блочного синтеза автоматов (см., например, работы Д. Хафмена [3]), Г. Мили [4]), В. И. Шестакова [5]), М. Уилкса [6]) и др.).

---

[1]) См ЖТФ, т. 11 : 6, 1941, стр. 532.

[2]) См. Trans. AIEE, v. 57, 1938, p. 713—723.

[3]) См Journal of the Franklin Inst., v. 257, № 3 and 4, 1954, p. 161—190, 275—303.

[4]) См. Bell system Tech. J., v. 34, 1955, p. 1045—1079.

[5]) См Автоматика и телемеханика, т. 15, № 4, 1954, стр. 310—324. Докл. АН СССР, т. 99, № 6, 1954, стр. 987—990.

[6]) См. Proc. East. Joint. Comput. Conf. NT-114, New York, 1959, p. 18—20.

Несколько отстала в своем развитии теория абстрактного и надежностного синтеза автоматов. Первый существенный сдвиг в этих разделах теории автоматов произошел в результате появления известного сборника статей «Автоматы» под редакцией К. Шеннона и Дж. Маккарти. В первую очередь этот сдвиг был обязан помещенным в указанном сборнике работам С. К. Клини [1]), Э. Мура [2]) и Д. Неймана [3]).

В упомянутых работах Клини и Мура было положено начало возникновению абстрактной теории автоматов, хотя сами полученные в них результаты были еще достаточно далеки от возможности реального их использования на этапе абстрактного синтеза автоматов, где в это время продолжали господствовать эмпирические методы. В работе Неймана было положено начало созданию общей математической теории для одной из самых интересных областей надежностного синтеза, именно — теории синтеза надежных схем из ненадежных элементов.

В работах автора [4]) теория абстрактного синтеза автоматов была доведена до такого уровня, который обеспечивает ей возможность непосредственного приложения к решению реальных задач, возникающих при синтезе цифровых автоматов. В эту теорию в качестве ее органической составной части вошли также результаты, полученные

---

[1]) См. сб. «Автоматы», под ред. К. Шеннона и Дж. Маккарти, перев с англ., ИЛ, М., 1956, стр. 15—67.

[2]) См. сб. «Автоматы». под редакцией К. Э. Шеннона и Дж. Маккарти. стр. 179—212.

[3]) См сб. «Автоматы», под редакцией К. Э. Шеннона и Дж. Маккарти, стр. 68—139.

[4]) См. Укр. матем. журнал, т. 12, № 2, 1960, стр. 147—156. Докл. АН УССР, т. 12, № 9, 1960, стр. 1151—1154 Вычислит. математика и математич. физика, т. I, № 3, 1961, 371—411.

другими авторами (см., например, работу Д. Ауфенкампа и Ф. Хона [1]).

В работе автора [2]) была решена также задача сопряжения этапов абстрактного и структурного синтеза и представление структурной теории в виде, пригодном для решения проблем синтеза цифровых автоматов с запоминающими элементами любой природы. Вопросы сопряжения этапов структурного и комбинационного синтеза и развитие самой теории комбинационного синтеза были успешно решены на более ранних этапах развития теории в трудах многих авторов, часть из которых была уже отмечена выше. Существенные сдвиги произошли также в теории надежностного и блочного синтеза.

Таким образом, в настоящее время возникла возможность изложения теории синтеза цифровых автоматов как единой математической теории. Задача систематического изложения этой теории как раз и является той целью, которую преследует настоящая книга. Эта задача определяет собою специфику построения книги и отбора материала для нее.

Что касается характера изложения, то он определяется следующими соображениями, казавшимися автору существенными для достижения указанной цели.

Во-первых, по мере возможности строить книгу таким образом, каким обычно строятся математические теории. Это означает, в частности, необходимость заботы о том, чтобы вводимые понятия были точно определены, а высказанные утверждения строго доказаны. Это означает также стремление к органической взаимосвязи излагаемого материала с целью избежать превращения книги в набор не связанных друг с другом рецептов. Разумеется,

[1]) См. IRE Trans., v. EC-6, № 4, 1957, p. 276—285 (русский перевод в сб. переводов «Математика», 1959, № 3 : 3, стр. 129—146).

[2]) См. Вычислит. математика и математич. физика, т. 1, № 3, 1961.

осуществить эту идею в полной мере в настоящее время затруднительно прежде всего потому, что разделы теории, касающиеся надежностного и блочного синтеза, все еще находятся в начальной стадии своего формирования и пока далеки от завершения.

Во-вторых, изложение строится таким образом, чтобы достичь максимальной общности при сохранении возможности непосредственного практического применения теории. Поэтому из книги исключены вопросы, связанные с особенностями конкретной физической реализации используемых элементов и устройств, но не оказывающие существенного влияния на процесс синтеза схем на принятом уровне абстракции.

В-третьих, автор считал необходимым введение различных уровней абстракции, в частности в принятой выше классификации последовательных этапов синтеза. Особенно важно отметить введение уровня абстракции, промежуточного между тем уровнем, который имеет место при применении булевой алгебры, и тем, с которым имеют дело при использовании заимствованных из радиотехники и импульсной техники методов расчета схем. Именно этот уровень, будучи тесно связанным с уровнем абстракции, принятым в абстрактной и структурной теории автоматов, позволяет осуществить тесную связь этих двух разделов теории с задачами, возникающими при синтезе реальных схем цифровых автоматов. Благодаря введению такого уровня абстракции оказывается возможным учесть особенности синтеза схем различных типов (потенциальных, импульсных и смешанных) безотносительно к конкретной природе элементов, из которых построены эти схемы. Вводимые абстрактные физические модели логических и запоминающих элементов могут допускать конкретные реализации на основе самых различных физических принципов (не обязательно на основе принципов электроники).

Наконец, главной и определяющей задачей изложения является такой подбор материала и такой характер его изложения, который дал бы возможность широкому кругу лиц и прежде всего широкому кругу математиков, не знакомых с радиотехникой, электроникой и импульсной техникой, понять суть проблем, встающих при синтезе схем современных сложных цифровых автоматов и прежде всего электронных вычислительных машин. Изложение построено таким образом, чтобы после изучения материала внимательный читатель мог самостоятельно синтезировать различные варианты логических схем больших универсальных электронных цифровых вычислительных машин, а также схем относительно небольших цифровых автоматов произвольного назначения с учетом простейших соображений надежности их работы.

Подобная целенаправленность книги заставила автора произвести весьма жесткий отбор излагаемого в ней материала. Пришлось полностью исключить из рассмотрения многие интересные разделы современной математической теории автоматов, содержащие подчас весьма глубокие научные результаты, но имеющие лишь косвенное отношение к проблемам практического синтеза автоматов; в книгу не включались и материалы, имеющие большое практическое значение при синтезе схем из тех или иных конкретных элементов, но не обладающие достаточной степенью общности. По той же причине исключены подробности, связанные с методикой выполнения арифметических операций в автоматах, нашедшие к тому же уже достаточное отражение в монографической литературе, изданной на русском языке.

Автору казалось, что реализация указанных общих идей в книге должна привести к тому, что такая книга будет существенно отличаться от всех изданных до настоящего времени руководств по синтезу схем цифровых

автоматов. О том, насколько автору удалось справиться с этой задачей, предоставляется судить читателю.

В заключение скажем несколько слов о характере распределения материала в книге. Книга состоит из семи глав.

Первая глава фактически заменяет собою введение. Основная цель этой главы — установить степень общности тех задач, которые решаются в последующих главах, и уточнить некоторые понятия (прежде всего понятие *алгоритма*), которые имеют существенное значение для дальнейшего изложения, но которые сами по себе не составляют предмета настоящей книги.

Вторая глава посвящена изложению проблем, возникающих на этапе абстрактного синтеза автоматов. Эта глава почти полностью построена на основании собственных результатов автора [1]), изложенных в упомянутых выше его работах, а также некоторых новых результатов, излагаемых впервые. То же самое относится к первым двум, а также к последнему, параграфам третьей главы, которая посвящена структурной теории автоматов. В остальных параграфах этой главы содержится теоретический материал, подготавливающий читателя к усвоению проблематики и методов комбинационного синтеза. Наряду с некоторыми новыми вопросами здесь изложены также основы булевой алгебры, включая теорему о функциональной полноте; § 7 гл. III существенным образом использует идеи и методы, содержащиеся в работе С. В. Яблонского «Функциональные построения в k-значной логике» [2]).

Последующие две главы, четвертая и пятая, посвящены проблемам комбинационного синтеза и минимизации комбинационных схем. В этих главах излагаются

---

[1]) См. Вычислит. математика и математич. физика, т. I, № 3, 1961, стр. 371—411.

[2]) См. Труды Матем. ин-та им. В. А. Стеклова, т. 51, 1958, стр. 5—142.

в основном известные результаты и методы комбинационного синтеза, принадлежащие К. Шеннону, Г. Н. Поварову, Квайну, Блейку, Карнау и др., хотя в ряде случаев их интерпретация и доказательства существенно изменены в соответствии с изложенными выше общими установками; произведены также уточнения постановок задач о графических (табличных) приемах минимизации булевых функций. И в эти главы включено изложение некоторых новых понятий и результатов.

Глава шестая посвящена некоторым проблемам надежностного синтеза схем автоматов. Введенные в ней понятия и определения позволяют реализовать тот — промежуточный между логическим и радиотехническим — уровень абстракции, о котором шла речь выше.

В главе произведено обобщение постановки и методов решения известных проблем гонок и риска. Изложены также (в несколько измененной интерпретации) известные результаты Шеннона — Мура о синтезе надежных схем из ненадежных элементов. Приведены простейшие сведения о самокорректирующихся кодах.

В последней, седьмой главе излагается принцип программного управления и блочный синтез схем универсальных программных автоматов, дается понятие о микропрограммировании как в общем случае, так и применительно к схеме, предложенной Уилксом и Стринджером[1]), разбирается вопрос о синтезе некоторых специальных схем (сумматоров, счетчиков, сдвиговых регистров), играющих значительную роль при синтезе схем универсальных программных автоматов.

Причина, по которой глава, посвященная блочному синтезу, оказалась последней, а не первой (что было бы

---

[1]) См Proc. Cambridge Philos. Soc., v. 49, № 4:2, 1955, p. 230—238.

естественно, имея в виду указанную выше последовательность этапов синтеза), заключается в том, что о б щ е й теории блочного синтеза цифровых автоматов в настоящее время не существует. Предпосылать же частные приемы решения задачи первого этапа общим приемам решения задач последующих этапов вряд ли было бы разумно, тем более, что с помощью указанных общих приемов полностью решается задача синтеза цифровых автоматов относительно небольшой сложности, имеющая весьма важное самостоятельное значение.

Имея в целом прикладную направленность, излагаемый в книге материал основан на широком использовании идей, понятий и средств математической логики. Так, гл. IV, бо́льшая часть гл. III, а также гл. V представляют собой, по существу, разделы современной прикладной логики, точнее, прикладной теории булевых функций. Абстрактная теория алгоритмов, излагаемая в гл. II, может рассматриваться как один из разделов теории алгоритмов, также имеющий ярко выраженную прикладную направленность. Один из аспектов прикладной теории алгоритмов составляют и вопросы алгоритмической структуры универсальных цифровых машин, рассматриваемые в §§ 1 и 3 гл. VII. Всем сказанным и объясняется включение этой книги в серию «Математическая логика и основания математики».

Для чтения книги не требуется никакой специальной математической или логической подготовки. Из числа вопросов, выходящих за рамки школьного курса математики, от читателя требуется только знакомство с двоичной системой счисления и с простейшими понятиями теории вероятностей (последние, впрочем, используются лишь в шестой главе).

В. М. Глушков

Киев,
май 1961 г.

ГЛАВА 1

ОБЩИЕ СВЕДЕНИЯ О ПРЕОБРАЗОВАНИЯХ ИНФОРМАЦИИ

§ 1. Понятие об информации и ее преобразованиях

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

С наиболее общей точки зрения процесс получения информации есть не что иное, как процесс снятия неопределенности в результате того, что из некоторой совокупности возможных в данной конкретной ситуации явлений выделяется явление, фактически имевшее место. Таким образом, в понятии информации существенно не само происшедшее явление, а лишь его отношение к совокупности явлений, которые могли произойти.

Это обстоятельство позволяет при информационном подходе к изучению явлений абстрагироваться от многих их свойств. Ясно, например, что произвольное одновременное изменение количественных масштабов рассматриваемых явлений с информационной точки зрения ничего не меняет. В частности, при передаче одних и тех же сообщений можно пользоваться как большими, так и малыми энергиями. Несущественным в информационном плане является также одновременное изменение пространственных и временных масштабов явлений.

Существуют два различных подхода к изучению явлений с информационной точки зрения: непрерывный и дискретный. При непрерывном подходе все изучаемые явления рассматриваются как переменные

векторные поля. Конкретная физическая природа таких векторных полей, а также их количественные, пространственные и временные масштабы считаются при этом несущественными. Задание информации состоит в выборе какого-нибудь определенного (переменного) поля из фиксированной заранее совокупности таких полей. Характерным для непрерывного подхода является то, что все описывающие явление величины (компоненты векторов, пространственные и временные координаты) являются вещественными числами и могут изменяться непрерывно.

При дискретном подходе также имеют дело с переменными векторными полями. Однако, в отличие от предыдущего случая, компоненты векторов, а также пространственные и временные координаты принимают дискретные ряды значений. Наиболее употребительным является случай, когда число значений, принимаемых компонентами векторов и пространственными координатами, конечно (поле задано в конечном числе точек). Что же касается временной координаты, то ее область значений при дискретном подходе отождествляют обычно с множеством целых чисел (положительных, отрицательных и нуля). Нулевой момент времени считается при этом начальным, а остальные моменты времени получают названия в соответствии с их номерами: первый, второй, минус первый, минус второй и т. д. При этом чаще всего ограничиваются рассмотрением конечных временных промежутков, начиная либо с нулевого, либо с первого момента времени.

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

Алфавитный способ употребляется, как известно, при задании лексической (языковой) и числовой информации. В случае лексической информации алфавит совпадает с обычным алфавитом языка, на котором задается информация, в случае числовой информации алфавитом является совокупность десяти цифр.

Следует особо подчеркнуть, что введенное выше понятие *слова в алфавите* существенно отличается от понятия слова в обычном языке даже в том случае, когда исходный алфавит совпадает, например, с русским алфавитом. Различие заключается в том, что, в силу принятых нами определений, мы будем называть словами не только все фактически существующие слова русского языка, но также и любые бессмысленные сочетания букв. Сочетания букв «автомат», «теорема», «кклмизръых», «ппд» с этой точки зрения в равной мере заслуживают названия слов в русском алфавите.

Если теперь к обычному русскому алфавиту добавить еще знаки раздела и знаки препинания, то словами в этом расширенном алфавите можно считать куски фраз, фразы, абзацы и даже целые книги. Отметим также относительность обычного разбиения лексической информации на буквы: при желании можно, разумеется, считать отдельными буквами слова и даже целые фразы.

Алфавитный способ задания информации является достаточно универсальным. Действительно, как уже было показано выше, этим способом можно осуществить представление любой дискретной информации. Что же касается информации, задаваемой в непрерывной форме, то на практике, применяя методы аппроксимации, ее всегда удается представить с любой наперед заданной степенью точности в дискретной форме.

Ввиду принципиальной важности процесса сведéния непрерывной формы задания информации к дискретной рассмотрим его несколько подробнее. Пусть непрерывная информация задается в виде переменных векторных полей на участке пространства, имеющем ограниченные размеры. Пусть далее эта информация воспринимается некоторым объектом (автоматическим устройством или человеком). Всякое реальное устройство, воспринимающее непрерывную информацию, обладает рядом ограничений, не

позволяющих ему различать многие тонкие свойства несущих информацию процессов.

Так, ограниченная чувствительность воспринимающего устройства дает ему возможность различать лишь конечное число градаций значений величин (компонентов векторных полей), характеризующих процесс. Ограниченная разрешающая способность устройства приводит к тому, что достаточно малые участки пространства воспринимаются им как отдельные точки.

В результате этого, с точки зрения воспринимающего устройства, участок пространства, на котором задаются несущие информацию векторные поля, представляется в виде конечного числа различных точек. Наконец, ограниченность полосы пропускания воспринимающего информацию устройства не позволяет ему различать очень близкие моменты времени. В результате (опять-таки с точки зрения воспринимающего устройства) можно ограничиться дискретным временем, дробя реальное непрерывное время на столь малые временные интервалы, чтобы концы этих интервалов воспринимались объектом как один момент времени.

Суммируя все отмеченные выше реальные ограничения, которым подвержены воспринимающие устройства, мы приходим к выводу, что с точки зрения устройства, воспринимающего информацию, любая непрерывная информация сводится, фактически, к дискретной, а следовательно, в конечном счете — к алфавитной информации. Отсюда следует универсальность алфавитного способа задания информации.

Следует, впрочем, сразу же подчеркнуть, что, несмотря на универсальность алфавитного способа задания информации, пользование им далеко не всегда является естественным, так что специфические методы для изучения непрерывной информации полностью сохраняют свое значение. Вместе с тем существуют и такие виды информации, при изучении которых дискретные методы являются не только вполне естественными, но, по-видимому, и единственно возможными. Наиболее важным примером такого вида информации является лексическая информация.

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

Поскольку целью настоящей книги является изучение дискретных преобразователей информации, то в дальнейшем под словом информация мы всегда будем понимать (если специально не оговорено противное) информацию, заданную в дискретной — и даже, точнее, в алфавитной форме. При этом процесс задания информации состоит в выборе определенного слова в некотором фиксированном конечном алфавите из совокупности всех возможных слов в этом алфавите.

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

Предположим, что любое явление α из некоторого класса A явлений влечет за собой некоторое определенное явление β из того же самого или любого другого класса B явлений. В таком случае говорят, что нам задано преобразование информации $\alpha \to \beta = f(\alpha)$[1]). Следовательно, с абстрактной точки зрения преобразование информации есть не что иное, как о т о б р а ж е н и е одного класса явлений в другой класс явлений.

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

---

[1]) Стрелка «$\to$» здесь служит в качестве знака преобразования информации, или, говоря иначе, в качестве обозначения отображения явления α в явление β.

§ 2. Преобразования алфавитной информации

В соответствии с высказанными в предыдущем параграфе соображениями мы будем рассматривать исключительно дискретно заданную алфавитную информацию и ее преобразования.

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

Оказывается целесообразным также в самом общем случае считать φ частичным отображением, иначе говоря, задавать отображение φ не обязательно на всем множестве F, а лишь на части этого множества (не исключая, впрочем, случай совпадения этой части со всем множеством F). Введение частичных отображений позволяет вместо отображений одного множества слов в другое рассматривать лишь отображения таких множеств в себя. Для этой цели достаточно ввести объединенный алфавит $\mathfrak{Z}=(x_1, \ldots, x_m, y_1, \ldots, y_n)$ и множество $H=H(x_1, \ldots, x_m, y_1, \ldots, y_n)$ слов над этим алфавитом. Ясно теперь, что вместо отображения (или частичного отображения) множества F в множество G можно рассматривать частичное отображение множества H в себя. Это частичное отображение будет определено для слов, состоящих только из букв $x_1, x_2, \ldots, x_m$.

Однозначность отображения φ множества F в множество G не означает, вообще говоря, как известно, однозначности

обратного отображения φ^{-1}. Если же такая однозначность имеет место, то отображение φ называется *взаимно однозначным*. Мы будем говорить также в этом случае, что отображение φ осуществляет эквивалентное преобразование информации. В случае эквивалентного преобразования заключительная (выходная) информация полностью определяет исходную (входную) информацию.

Пример 1. Преобразование информации в произвольном конечном алфавите заключается в зеркальном отображении слов: порядок следования букв в слове меняется после преобразования на прямо противоположный. Ясно, что такое преобразование будет эквивалентным.

Пример 2. В алфавите, состоящем из всех десятичных цифр и знака суммы, задается отображение слов вида $a+b$, где a и b — целые числа, в сумму этих двух чисел; например, слово 2+5 преобразуется в слово 7, слово 21+ +15 в слово 36 и т. д. Легко видеть, что задаваемое таким образом преобразование не эквивалентно, поскольку задание суммы не определяет слагаемых.

Со всяким преобразованием информации связывается интуитивное представление о его сложности. Ясно, например, что преобразование какого-нибудь русского текста, заключающееся в замене букв их порядковыми номерами в алфавите, много проще, чем перевод того же текста, скажем, на английский язык. По критерию сложности оказывается целесообразным выделить один класс преобразований информации, которые естественно называть простейшими.

Простейшими, или *побуквенными*, мы будем называть преобразования, заключающиеся в замене каждой буквы исходного алфавита некоторой определенной, имеющей заранее фиксированную, одинаковую для всех букв длину, комбинацией букв нового алфавита.

Приведем примеры простейших преобразований информации.

Пример 3. Исходная информация — произвольное слово в русском алфавите. Преобразование информации состоит в замене каждой буквы алфавита порядковым номером, записанным в десятичной системе счисления. При этом для соблюдения условия равенства длин комбинаций букв нового алфавита, представляющих буквы старого

алфавита, необходимо первую букву обозначать комбинацией 01, вторую — комбинацией 02 и т. д. В результате такого преобразования слово «дом» преобразуется в слово «051412», слово «бкзд» — в слово «02100805» и т. д.

Пример 4. Исходная информация — произвольное целое десятичное число (слово в алфавите, состоящем из десяти цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9); преобразование информации состоит в замене каждой четной цифры нулем, а каждой нечетной цифры — единицей. Слово «125» при этом преобразуется в слово «101», слово «0342» — в слово «0100» и т. д.

Заметим, что преобразование в примере 3 является не только простейшим, но и эквивалентным. В то же время преобразование в примере 4, хотя и простейшее, тем не менее не эквивалентно.

Легко видеть, что с помощью простейших эквивалентных преобразований, информацию, заданную в любом конечном алфавите, можно записать в алфавите, содержащем только две буквы. Такой алфавит мы будем называть *стандартным двухбуквенным*, или *двоичным*, алфавитом, а две его буквы будем обозначать нулем и единицей. Такие преобразования выполняются следующим образом: ищется такое натуральное число m, чтобы число различных слов длины m в двоичном алфавите превышало или было бы равным числу n различных букв исходного алфавита A, после чего каждая из букв алфавита A заменяется словом длины m в двоичном алфавите так, чтобы различным буквам соответствовали различные слова. Поскольку, как легко показать, число различных слов длины m в двоичном алфавите равно 2^m, то для возможности выполнения указанного преобразования достаточно выбрать m так, чтобы удовлетворялось неравенство

$$2^m \geqslant n.$$

Ясно, что кодирование информации в двоичном алфавите неоднозначно. Более того, для любого исходного алфавита существует бесконечное множество таких преобразований. Все эти преобразования носят специальное название *двоичного кодирования* исходной информации.

Объединяя приведенные здесь рассуждения с тем, что было сказано в предыдущем параграфе, мы получаем воз-

можность высказать следующее утверждение: при различных преобразованиях информации можно, не нарушая общности, предполагать, что как исходная, так и заключительная информация задана в некотором стандартном (например, двоичном) алфавите.

В самом деле, пусть некоторая информация A с помощью преобразования f преобразуется в вид B. Закодируем как информацию A, так и информацию B в некотором стандартном алфавите, обозначая соответствующие кодирующие отображения через α и β, а закодированную информацию — через A' и B', т. е. $A'=A\alpha$, $B'=B\beta$ [1]).

Легко видеть, что преобразование f индуцирует некоторое преобразование f', переводящее информацию A' в информацию B'. Обозначая через α^{-1} и β^{-1} преобразования, обратные соответственно преобразованиям α и β, мы получим:

$$\alpha f' \beta^{-1} = f, \quad \alpha^{-1} f \beta = f'. \tag{1}$$

Приведенные соотношения легко усмотреть из схемы, изображенной на рис. 1.

Само собой разумеется, что в качестве стандартного может быть выбран не обязательно двоичный, а любой другой конечный алфавит, состоящий более чем из одной буквы. Правила кодирования при этом сохраняются. Нужно лишь вместо неравенства $2^m \geqslant n$ пользоваться неравенством $k^m \geqslant n$, где k — число букв в выбранном стандартном алфавите.

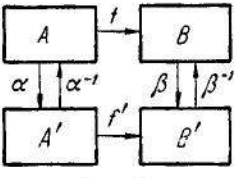

Рис. 1

Выполнение простейших преобразований информации и обратных им преобразований обычно не вызывает затруднений и может выполняться с помощью относительно простых автоматических устройств. Поэтому вся трудность, связанная с тем или иным конкретным преобразованием информации, может быть сведена к установлению некоторого соответствия слов в стандартном алфавите. Если в качестве стандартного берется, например, двоичный

[1]) Если C — информация, а γ — отображение, то через $C\gamma$ обозначается результат воздействия отображения γ на информацию C.

алфавит, то суть проблемы состоит в получении из одного ряда нулей и единиц другого ряда нулей и единиц, отличающегося от первого либо числом символов, либо их взаимным расположением, либо тем и другим вместе.

Приведем пример сведения сложного процесса преобразования информации к преобразованию слов в двоичном алфавите.

Пусть исходной информацией служат черно-белые рисунки стандартного формата, изображающие тот или иной предмет из некоторого фиксированного множества предметов. Преобразование информации состоит в том, что каждому рисунку сопоставляется название изображенного на нем предмета, записанное на русском языке. Произведем кодирование исходной и заключительной информации в двоичном алфавите. Для заключительной информации такое кодирование может быть выполнено очень просто: достаточно каждой из 32 букв русского алфавита сопоставить одно из $32=2^5$ возможных слов длины 5 в двоичном алфавите. Например, букве *а* может быть сопоставлен код 00000, букве *б* — код 00001 и т. д.

Несколько более сложно кодируется исходная информация. Для того чтобы выполнить кодирование, разобьем системой горизонтальных и вертикальных прямых площадь каждого рисунка на элементарные квадратики и занумеруем эти квадратики в том или ином фиксированном порядке (например, слева направо и сверху вниз). Величину каждого элементарного квадратика выбираем настолько малой, чтобы глаз не различал на нем деталей, а мог различать только степень его яркости. Выбираем, далее, максимальное число градаций яркости, воспринимаемых глазом как различные. Ввиду ограниченной способности глаза различать степени яркости изображения, это число будет конечным. Предположим, что оно равно, например, 200. Поскольку имеется ровно $256=2^8$ различных слов длины 8 в двоичном алфавите, мы можем закодировать яркости элементарных квадратиков восьмерками нулей и единиц, взятых в различных комбинациях. Записывая теперь коды яркостей элементарных квадратиков в порядке роста номеров квадратиков, мы получаем возможность кодировать каждый рисунок рядом нулей и единиц.

Задача распознавания рисунков оказывается тем самым сведенной к преобразованию ряда нулей и единиц, являющегося кодом рисунка, в ряд нулей и единиц, представляющий (в закодированной форме) название изображенного на этом рисунке предмета.

§ 3. Понятие об алгоритме

В соответствии с принятой нами дискретной точкой зрения произвольное преобразование информации есть не что иное, как отображение (вообще говоря, частичное) множества слов в некотором конечном алфавите в множество слов в том же самом или любом другом конечном алфавите. Условимся называть такие отображения *алфавитными операторами*.

Один из первых вопросов, возникающих при изучении алфавитных операторов,— это вопрос о способах их задания. Дело в том, что, вообще говоря, алфавитный оператор определяется на бесконечном множестве слов и не может быть поэтому задан простой таблицей соответствия. Возникает необходимость нахождения способа задания алфавитных операторов на основе конечной информации. Алфавитные операторы, задаваемые с помощью конечных систем правил, принято называть *алгоритмами*. Приведем примеры алгоритмов.

Пример 1. Рассмотрим алгоритм сложения целых чисел в десятичной системе счисления. Этот алгоритм перерабатывает слова вида $a+b$ в алфавите, состоящем из 10 цифр и знака сложения, в слова в том же алфавите, из которого исключен знак суммы. Конечная система правил, определяющих алгоритм, состоит из правила поразрядного сложения, правила нахождения соответствующих друг другу разрядов слагаемых (в том числе младших разрядов), правил сложения цифр (таблица сложения) и правила переноса.

Пример 2. Рассмотрим алгоритм игры в шахматы. Игру в шахматы можно трактовать как преобразование слов в конечном алфавите. В качестве алфавита здесь естественно применить обычно употребляющиеся в шахматной нотации символы. Словами, как обычно, будут считаться конечные последовательности букв алфавита.

Некоторые из таких последовательностей будут представлять собою шахматные позиции, другие — просто бессмысленные наборы символов (например, $3КЛad$). Алгоритм задается лишь для слов, представляющих осмысленные позиции, и заключается в преобразовании любой заданной позиции в позицию, возникающую из нее после выполнения очередного хода.

Как известно, правила движения шахматных фигур еще не определяют однозначно выбора хода в той или иной конкретной позиции и, следовательно, не представляют еще алгоритма. Однако, как показывает опыт, можно составить конечную систему стратегических правил, позволяющих в каждой конкретной позиции выбирать единственный наилучший (в некотором смысле) ход. Присоединяя эту систему правил к правилам движения фигур, мы получаем алгоритм, который естественно назвать алгоритмом шахматной игры.

Следует сразу оговориться, что в зависимости от того, каким образом определяется качество позиции, меняется и смысл понятия «наилучшего» хода. Это обстоятельство определяет наличие не одного единственного алгоритма шахматной игры, а множества таких алгоритмов.

Мы оставляем также в стороне вопрос о вероятностных алгоритмах, где выбор образа для того или иного слова определяется не однозначно, а случайно, в соответствии с какими-либо вероятностными критериями. Ясно, что применительно к шахматным алгоритмам рассмотрение случайных переходов является весьма уместным.

В приведенном выше определении алгоритма как алфавитного оператора, задаваемого конечной системой правил, есть один существенный недостаток — отсутствие математической точности определения. Не преодолев этого недостатка, нельзя сделать понятие алгоритма предметом точной математической теории. Были поэтому предприняты многочисленные попытки уточнения понятия алгоритма, которые привели к установлению способов точного задания произвольных алгоритмов.

Мы приведем один из таких способов, принадлежащий А. А. Маркову [1]).

[1]) А. А. Марков, Теория алгоритмов. Труды Матем. ин-та им. В. А. Стеклова, т. 42, 1954.

Алгоритмы А. А. Маркова, называемые также *нормальными алгоритмами*, преобразуют слова, заданные в любом конечном алфавите $A = A(a_1, \ldots, a_n)$, в слова в том же самом алфавите, причем, как правило, алгоритм оказывается определенным лишь для части слов и задает, следовательно, частичное отображение.

Нормальный алгоритм задается конечной таблицей подстановок слов в данном алфавите. Чтобы не вводить излишних усложнений, объяснение способов задания таблицы подстановок и методов пользования этой таблицей мы проведем на следующем частном примере.

Предположим, что алфавит, в котором функционирует нормальный алгоритм, состоит из трех букв: x, y, z, а таблица подстановок имеет вид: [1])

$$1.\ xy \longrightarrow yx,$$
$$2.\ zxz \longrightarrow x,$$
$$3.\ zz \longrightarrow zx.$$

Пусть теперь нам задано произвольное слово в алфавите (x, y, z), например слово $zzzxyz$. Правила преобразования этого слова с помощью алгоритма — выписанной выше таблицы подстановок — таковы. Прежде всего ищется первая по порядку (считая сверху вниз) формула таблицы, левая часть которой входит в рассматриваемое слово, выделяется первое вхождение (считая слева направо) левой части формулы в слово и это вхождение заменяется правой частью формулы. Эта операция повторяется до тех пор, пока любая из формул таблицы не окажется далее неприменимой (будут отсутствовать вхождения левых частей всех формул таблицы в полученное на предыдущем этапе слово) либо пока не будет применена одна из формул таблицы, выделенных заранее как заключительные формулы и отмеченных специальным значком (в частном случае заключительные формулы могут и отсутствовать).

При применении указанных правил возможны два случая. Процесс может закончиться через конечное число шагов: в этом случае слово, полученное после выполнения последней подстановки, и будет тем словом, в которое наш

[1]) Стрелки служат для выражения подстановки слова, стоящего справа от нее, вместо вхождения слова, стоящего слева от нее.

алгоритм переводит исходное слово. Но может оказаться, что процесс преобразования слова будет происходить без конца: в таком случае считается, что алгоритм к исходному слову неприменим.

Если в рассматриваемом примере не выделять заключительных формул, то процесс преобразования исходного слова будет выглядеть следующим образом.

1-й шаг. Применяется формула 1. Слово $zzzxyz = zzz(xy)z$ преобразуется в слово $zzzyxz$ (скобками выделено первое вхождение левой части применяемой формулы в исходное слово).

2-й шаг. Формулы 1 и 2 неприменимы, применяется 3-я формула. Слово $zzzyxz = (zz)zyxz$ преобразуется в слово $zxzyxz$.

3-й шаг. Формула 1 неприменима, применяется 2-я формула. Слово $zxzyxz = (zxz)yxz$ преобразуется в слово $xyxz$.

4-й шаг. Снова применяется формула 1. Слово $xyxz = (xy)xz$ преобразуется в слово $yxxz$.

5-й шаг. К слову $yxxz$ неприменима ни одна из формул таблицы; поэтому процесс переработки закончен: исходное слово $zzzxyz$ переработано алгоритмом в слово $yxxz$.

Если бы в рассмотренном нами примере выделить, скажем, формулу 2 как заключительную, то процесс оборвался бы уже после 3-го шага, поскольку на этом шаге была бы применена заключительная формула. В результате получился бы уже другой алгоритм, который перерабатывает слово $zzzxyz$ в слово $xyxz$, а не в слово $yxxz$.

Естественно задать вопрос о степени общности понятия нормального алгоритма. Специальные исследования, предпринятые для выяснения этого вопроса, с достаточной убедительностью показывают, что любой алгоритм эквивалентен некоторому нормальному алгоритму. Иначе говоря, для любого алгоритма над произвольным конечным алфавитом A можно построить нормальный алгоритм над тем же или, быть может, расширенным алфавитом, перерабатывающий слова над алфавитом A точно так же, как и исходный алгоритм.

Итак, говоря о произвольном алгоритме, можно, не нарушая общности, иметь в виду только нормальные алгоритмы.

§ 4. Понятие о дискретном (цифровом) автомате

Дискретными автоматами принято называть устройства, служащие для преобразования дискретной информации. Как уже отмечалось в § 2, дискретную информацию можно всегда считать алфавитной и притом заданной в том или ином стандартном алфавите. В современных дискретных автоматах принято обычно отождествлять буквы используемого стандартного алфавита с цифрами той или иной системы счисления (чаще всего двоичной или десятичной). Поэтому дискретные автоматы принято называть также *цифровыми автоматами*.

Основным качеством, выделяющим дискретные автоматы из числа всех других преобразователей информации, является наличие д и с к р е т н о г о (при этом в реальных автоматах всегда конечного) множества внутренних состояний и свойства с к а ч к о о б р а з н о г о п е р е х о д а автомата из одного состояния в другое. Скачкообразность перехода означает возможность трактовать этот переход как мгновенный, причем как такой, который совершается непосредственно, минуя какие-либо промежуточные состояния.

Разумеется, для любого реально существующего автомата имеет место конечная длительность переходных процессов, так что требование скачкообразности перехода в полной мере никогда не удовлетворяется. Удовлетворение этого требования оказывается возможным таким образом лишь на определенной ступени абстракции. Такая абстракция, однако, достаточно хорошо описывает основные свойства реальных цифровых автоматических устройств (прежде всего — электронных цифровых машин) и поэтому может быть принята для построения теории цифровых автоматов.

Второе допущение, которое также хорошо согласуется с действительностью, состоит в том, что после перехода автомата в произвольное состояние переход в следующее состояние оказывается возможным не ранее, чем через некоторый фиксированный для данного автомата промежуток времени $\tau > 0$,— так называемый *интервал дискретности* автомата. Это допущение дает возможность рассматривать функционирование цифрового автомата в так

называемом *дискретном времени*. При построении такого дискретного автоматного времени различаются д в а основных случая.

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

В противоположность синхронным автоматам, в *асинхронных автоматах* моменты переходов из одного состояния в другое заранее не определены и могут совершаться через неравные между собой промежутки времени. Для асинхронных автоматов можно ввести дискретное время, определяемое исключительно лишь одними моментами фактических переходов автомата из одного состояния в другое. Однако при этом теория асинхронных автоматов становится существенно отличной от соответствующей теории для синхронного случая. Поэтому мы в качестве моментов дискретного автоматного времени для асинхронных автоматов будем рассматривать не только моменты фактически имевших место переходов, но также и такие моменты, в которые переходы были возможны, но фактически не произошли. К числу таких моментов мы будем причислять моменты прихода входных сигналов импульсного типа (мгновенных) или изменения уровня напряжения входных сигналов так называемого «потенциального» типа (действующих на временно́м интервале). Разумеется, при этом необходимо считать, что интервал дискретности автомата ограничивает минимально возможное расстояние между дополнительно вводимыми моментами автоматного времени.

При таком допущении теория асинхронных автоматов в ряде случаев может быть сведена к синхронному случаю, поскольку фактические длины интервалов между последовательными моментами автоматного времени в идеализированной теории автоматов (без учета переходных процессов) не имеют никакого значения. Имея в виду это обстоятельство, мы в дальнейшем будем рассматривать *абстрактное автоматное время*, принимающее целые неотрицательные значения: $t=0,1,2,\ldots,n,\ldots$, и стро-

ить теорию, как теорию синхронных автоматов, хотя в действительности она будет охватывать в значительной мере и теорию асинхронных автоматов.

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

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

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

В первом случае принимается, что выходной сигнал $y(t)$ однозначно определяется входным сигналом $x(t)$ и состоянием $a(t-1)$ автомата в предыдущий момент времени, во втором случае сигнал $y(t)$ однозначно определяется парой $(x(t), a(t))$. Мы будем считать, что для любого данного автомата всегда имеет место лишь одна из этих

возможностей (одновременно для всех переходов). Цифровые автоматы, в которых выходной сигнал $y(t)$ определяется парой $(x(t), a(t-1))$, мы будем называть *автоматами первого рода*, а автоматы, в которых сигнал $y(t)$ определяется парой $(x(t), a(t))$,— *автоматами второго рода*.

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

Поскольку состояние $a(t)$ в любом цифровом автомате однозначно определяется парой $(x(t), a(t-1))$, то в идеализированной (сведенной к автоматному времени) теории всякий автомат второго рода может рассматриваться как частный случай автоматов первого рода. Таким образом, автоматы первого рода являются наиболее общим типом цифровых автоматов и именно их теория должна в первую очередь служить объектом изучения. Автоматы п е р в о г о р о д а обычно называются также *автоматами Мили*, по имени американского ученого, который впервые начал их систематическое изучение.

Следует отметить, однако, что при синтезе автоматов часто представляет интерес задача построения не какого-нибудь автомата вообще, а автомата того или иного частного вида. Особый интерес на практике имеют п р а в и л ь н ы е а в т о м а т ы в т о р о г о р о д а, известные обычно под более кратким названием *автоматов Мура* [1]). При дальнейшем построении теории автоматов мы будем поэтому, кроме теории автоматов Мили, строить также и теорию автоматов Мура.

Общая теория автоматов при сделанных выше допущениях естественно разбивается на две большие части, которым мы присвоим названия *абстрактной теории автоматов и структурной теории автоматов*. Различие между ними заключается в том, что в а б с т р а к т н о й т е о р и и мы отвлекаемся от структуры как самого автомата, так и его входных и выходных сигналов. Входные и выходные сигналы рассматриваются при этом просто как буквы двух фиксированных для данного автомата алфави-

[1]) Э. Ф. Мур — американский математик, начавший систематическое изучение правильных автоматов.

тов — входного и выходного. Не интересуясь способом построения автомата, абстрактная теория изучает лишь те переходы, которые претерпевает автомат под воздействием входных сигналов, и те выходные сигналы, которые он при этом выдает.

Абстрактная теория автоматов близка, таким образом, теории алгоритмов, являясь по существу ее дальнейшей детализацией. В противоположность абстрактной теории, структурная теория автоматов интересуется прежде всего структурой как самого автомата, так и его входных и выходных сигналов. В структурной теории изучаются способы построения автоматов из элементарных автоматов, способы кодирования входных и выходных сигналов элементарными сигналами, передаваемыми по реальным входным и выходным каналам, и т. п.

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

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

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

ГЛАВА II

АБСТРАКТНАЯ ТЕОРИЯ АВТОМАТОВ

§ 1. Понятие об абстрактном автомате и индуцируемом им отображении

Объектом изучения в абстрактной теории автоматов являются абстрактные автоматы вместе с реализуемыми ими отображениями и событиями.

1. 1. *Абстрактный автомат A задается как совокупность шести объектов: конечного множества $\mathfrak{X}$ входных сигналов, называемого входным алфавитом автомата, конечного множества $\mathfrak{Y}$ выходных сигналов, называемого выходным алфавитом автомата, произвольного множества $\mathfrak{A}$, называемого множеством состояний автомата, элемента a_0 из множества $\mathfrak{A}$, называемого начальным состоянием автомата, и двух функций $\delta(a, x)$ и $\lambda(a, x)$, задающих однозначные отображения множества пар (a, x), где $a \in \mathfrak{A}$ и $x \in \mathfrak{X}$, в множества $\mathfrak{A}$ и $\mathfrak{Y}$. Функция $\delta(a,x)$ называется функцией переходов автомата, а функция $\lambda(a,x)$ — функцией выходов, либо сдвинутой функцией выходов. Автомат, заданный функцией выходов, называется автоматом первого рода; автомат, заданный сдвинутой функцией выходов,— автоматом второго рода.*

Абстрактный автомат функционирует в дискретном времени, принимающем целые неотрицательные значения $t = 0, 1, 2, \ldots$. В каждый момент t этого времени он находится в определенном *состоянии* $a(t)$ из множества $\mathfrak{A}$ состояний автомата, причем в *начальный момент времени* $t=0$ автомат всегда находится в своем *начальном состоянии* a_0, т. е. $a(0)=a_0$. В каждый момент времени t, отличный от начального, автомат способен воспринимать *входной сиг-*

нал $x(t)$ — произвольную букву входного алфавита $\mathfrak{X}$ и выдавать соответствующий *выходной сигнал* $y(t)$ — некоторую букву выходного алфавита $\mathfrak{Y}$.

Закон функционирования абстрактного автомата в случае автомата первого рода задается уравнениями:

$$(1)\quad a(t)=\delta(a(t-1),x(t)),\ y(t)=\lambda(a(t-1),\ x(t)),\quad (t=1,\ 2,\ \ldots),$$

а в случае автомата второго рода — уравнениями:

$$(2)\quad a(t)=\delta(a(t-1),\quad x(t)),\ y(t)=\lambda(a(t),x(t))\quad (t=1,\ \ 2,\ \ldots).$$

Установлением закона функционирования заканчивается определение абстрактного автомата. Теперь необходимо остановиться на смысле понятия абстрактного автомата; этот смысл состоит в реализации некоторого отображения φ множества слов входного алфавита в множество слов выходного алфавита. Отображение φ реализуется следующим образом: каждое слово $p=x_{i_1}, x_{i_2}\ldots x_{i_k}$ входного алфавита $\mathfrak{X}=(x_1, \ldots, x_n)$, или, более кратко, каждое *входное слово*, последовательно, буква за буквой, подается на вход данного абстрактного автомата A, установленного предварительно в начальное состояние. Возникающая таким образом конечная последовательность входных сигналов $x(1)=x_{i_1}$, $x(2)=x_{i_2},\ldots, x(k)=x_{i_k}$ на основании *закона функционирования автомата* вызывает появление однозначно определенной конечной последовательности $q=y(1),y(2), \ldots, y(k)$ выходных сигналов. Эту последовательность мы будем называть *выходным словом*, соответствующим входному слову p.

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

1. 2. *Два абстрактных автомата с общим входным и выходным алфавитами называются эквивалентными между собой, если они индуцируют одно и то же отображение множества слов во входном алфавите в множество слов в выходном алфавите.*

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

Предположим, что нам заданы два абстрактных автомата: $A_1(\mathfrak{X}_1, \mathfrak{Y}_1, \mathfrak{A}_1, a_1, \delta_1(a, x), \lambda_1(a\ x))$ и $A_2(\mathfrak{X}_2, \mathfrak{Y}_2, \mathfrak{A}_2\ a_2, \delta_2(a, x), \lambda_2(a, x))$ одного и того же рода. Если существуют взаимно однозначные отображения: α, отображающее множество $\mathfrak{X}_1$ на множество $\mathfrak{X}_2$, β, отображающее множество $\mathfrak{Y}_1$ на множество $\mathfrak{Y}_2$, и γ, отображающее множество $\mathfrak{A}_1$ на множество $\mathfrak{A}_2$, и если удовлетворяются условия:

$$(1)\ \gamma(a_1) = a_2,$$

$$(2)\ \gamma(\delta_1(a\ x)) = \delta_2(\gamma(a),\ \alpha(x)),\ \beta(\lambda_1(a,\ x)) = \\ = \lambda_2(\gamma(a)\ \alpha(x))\ (\text{для любых } a \in \mathfrak{A}_1 \text{ и } x \in \mathfrak{X}_1),$$

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

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

Поскольку на протяжении всей настоящей главы мы будем оперировать исключительно с абстрактными автоматами, термин «абстрактный» будет, как правило, опускаться всякий раз, когда это сокращение не вызывает недоразумений.

Следующим важным вопросом, который нам предстоит выяснить, является вопрос о взаимоотношении между автоматами первого и второго рода.

Пусть нам дан произвольный автомат A второго рода, заданный функцией переходов $\delta(a, x)$ и сдвинутой функцией выходов $\lambda(a, x)$. Построим новую функцию $\lambda_1(a, x) = \lambda(\delta(a,x),x)$. Из закона функционирования автоматов второго рода(2) мы получаем: $y(t)=\lambda(\delta(a(t-1), x(t)),x(t))= \lambda_1(a(t-1), x(t))$. Теперь ясно, что автомат B первого рода, заданный той же функцией переходов $\delta(a,x)$, что и автомат A, и функцией выходов $\lambda_1(a,x)$ (при условии сохранения множества состояний, начального состояния, а также входного и выходного алфавита автомата A), индуцирует то же самое отображение множества входных слов в множество выходных слов, что и автомат A.

Нами доказано следующее предложение.

1. 3. *Для каждого автомата* $A(\mathfrak{X},\mathfrak{Y}, \mathfrak{A},a_0,\delta(a, x), \lambda(a,x))$ *второго рода существует эквивалентный ему автомат* $B(\mathfrak{X}, \mathfrak{Y}, \mathfrak{A}, a_0, \delta(a,x),\lambda_1(a,x))$ *первого рода, функция выходов которого получается в результате подстановки функции переходов автомата A в его сдвинутую функцию выходов*:

$$\lambda_1(a, x) = \lambda(\delta(a, x), x).$$

Способ сведения автоматов второго рода к автоматам первого рода, указанный в предложении 1.3., мы условимся называть *интерпретацией автомата второго рода как автомата первого рода.*

Обратная интерпретация автоматов первого рода как автоматов второго рода при сохранении того же самого множества состояний оказывается, вообще говоря, невозможной. Более сложный способ (с изменением множества состояний) построения обратной интерпретации будет изложен ниже.

Произвольные автоматы первого рода мы будем, как отмечено в § 4 гл. I, называть в дальнейшем *автоматами Мили*, а частный случай автоматов второго рода, у которых сдвинутая функция выходов $\lambda(a,x)$ не зависит от второй переменной x — *автоматами Мура*. Автомат будет называться *конечным* или *бесконечным* в соответствии с тем, конечно или бесконечно множество его состояний.

Исходя из соображений практики, объектом дальнейшего изучения будут служить в основном конечные автоматы Мили и Мура. Мы остановимся поэтому несколько

подробнее на *способах задания* таких автоматов. Поскольку конечные множества могут быть заданы простым перечислением их элементов, вопрос о задании конечных автоматов сводится к заданию функций переходов и функций выходов (обычных или сдвинутых). Обыкновенно эти функции задаются таблицами с двумя входами, называемыми соответственно *таблицей переходов и таблицей выходов* (обычной или сдвинутой) данного конечного автомата.

Строки обеих этих таблиц обозначаются входными сигналами автомата, а столбцы — его состояниями. При этом условимся, что начальное состояние всегда будет обозначать первый слева столбец. На пересечении x-й строки и a-го столбца таблицы переходов ставится соответствующее значение $\delta(a,x)$ функции переходов, а в таблице выходов (обычной или сдвинутой) — соответствующее значение $\lambda(a,x)$ функции выходов (обычной или сдвинутой).

Задание таблиц переходов и выходов в том виде, как это было только что описано, полностью определяет соответствующий конечный автомат, поскольку при этом задаются не только функции переходов и выходов, но и множества состояний входных и выходных сигналов, а также начальное состояние. Весьма часто для обозначения состояний автомата употребляются целые положительные числа. В этом случае мы условимся, для определенности, начальное состояние обозначать единицей, а в случае, когда употребляется также и нуль — нулем.

Для случаев автомата Мура сдвинутая таблица выходов сводится по существу к одной строке. Помещая эту строку над таблицей перехода, мы придем к понятию *отмеченной таблицы переходов* автомата Мура. В отмеченной таблице переходов над каждым состоянием a_i автомата, обозначающим тот или иной столбец таблицы, стоит соответствующий этому состоянию выходной сигнал $\lambda(a_i,x)=\lambda_0(a_i)$ (сдвинутая функция выходов не зависит от входного сигнала x!). Условимся говорить, что в произвольном автомате Мура каждое состояние *а отмечено* соответствующим ему выходным сигналом $\lambda_0(a)$.

Второй способ задания абстрактных автоматов основан на использовании *направленных графов*. *Граф* произвольного абстрактного автомата A представляет собою

комбинацию вершин, изображаемых на рисунках кружочками, и соединяющих их стрелок — *ребер графа*. Вершины отождествляются с состояниями автомата, а стрелки — с входными сигналами. Если входной сигнал x_i вызывает переход автомата из состояния a_j в состояние a_k, то на графе автомата этому сигналу соответствует обозначенная буквой x_i стрелка, соединяющая вершину, соответствующую состоянию a_j, с вершиной, соответствующей состоянию a_k. При этом, разумеется, не исключается случай, когда вершина a_j и a_k совпадают.

Построенный таким образом граф автомата A задает только функцию переходов этого автомата. Для задания функции выходов (обычной или сдвинутой) ребра графа (стрелки) обозначаются не только входными, но и соответствующими им выходными сигналами. Если обозначенная входным сигналом x_i стрелка соединяет вершину a_j с вершиной a_k, то в случае автоматов первого рода ей приписывается выходной сигнал $\lambda_1(a_j, x_i)$, а в случае автоматов второго рода — выходной сигнал $\lambda_2(a_k, x_i)$, где λ_1 и λ_2 — соответственно обычная или сдвинутая функция выходов автомата.

В случае автоматов Мура все стрелки, входящие в одну и ту же вершину a_k, должны быть обозначены одним и тем же выходным сигналом. Поэтому принято обозначать входными сигналами не стрелки, а вершины, в которые эти ребра входят.

Таким образом, на графе автомата Мура каждая вершина имеет два обозначения — одно, обозначающее состояние автомата, и другое, обозначающее выходной сигнал, о т м е ч а ю щ и й состояние автомата на основании сдвинутой функции выходов.

Легко видеть, что граф автомата, благодаря введенным на нем (в соответствии со сделанным выше описанием) обозначениям вершин и ребер, полностью задает автомат,— при условии, что каким-то образом отмечена вершина графа, соответствующая начальному состоянию этого автомата.

Переход от задания конечного автомата с помощью таблиц к заданию с помощью графа и обратный переход выполняются вполне очевидным образом. Пусть, например, конечный автомат Мура задан отмеченной таблицей

переходов (табл. II. 1). Этой таблице соответствует граф, изображенный на рис. 2.

Таблица II 1

	a	a	c	c
	1	2	3	4
x	2	3	3	2
y	1	4	4	1

Задание автомата с помощью графа обладает большей наглядностью по сравнению с заданием его с помощью таблиц, однако при большом числе состояний такой способ задания становится достаточно громоздким. Мы будем поэтому в основном пользоваться заданием автоматов с помощью таблиц, используя графы лишь для иллюстрации.

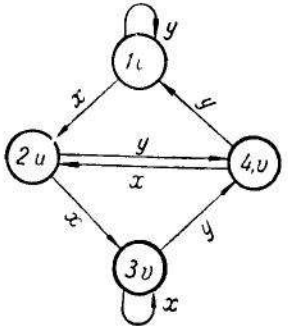

Рис 2

Следует упомянуть еще об одном упрощении, которым обычно пользуются при изображении графов автоматов: если несколько стрелок соединяют одну и ту же (упорядоченную) пару вершин, то все эти стрелки заменяются одной стрелкой того же направления с сохранением в ее обозначении всех букв, которые обозначали замененные ею стрелки. При этом, чтобы избежать путаницы в отношении выходных сигналов, каждый выходной сигнал пишется рядом с соответствующим ему входным сигналом и заключается обычно в скобки.

Отметим, что при обоих описанных способах задания автоматов особо просто осуществляется *интерпретация произвольного автомата Мура как автомата Мили*, описанная в более общем виде в предложении 1. 3.

В самом деле, в случае задания автомата Мура A отмеченной таблицей переходов достаточно подставить в эту таблицу вместо состояний отмечающие их выходные сиг-

налы, чтобы получить таблицу выходов автомата Мили B, эквивалентного автомату A. В случае задания автомата A графом для получения графа автомата B достаточно обозначить выходными сигналами не вершины графа, а стрелки, входящие в соответствующие вершины. Применение указанной процедуры к автомату Мура, граф которого изображен на рис. 2, приводит к эквивалентному ему автомату Мили, задаваемому графом, изображенным на рис. 3, и таблицами переходов и выходов II. 2 и II. 3.

Таблица II 2

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

Таблица II. 3

	1	2	3	4
x	u	v	t	u
y	u	t	v	u

В некоторых случаях оказывается удобным вместо таблицы переходов абстрактного автомата (любого рода) использовать так называемую *квадратную автоматную таблицу*. *Квадратной автоматной таблицей* мы будем называть таблицу, у которой как строки, так и столбцы обозначены различными состояниями автомата. Элементом таблицы, стоящим на пересечении a_i-й строки и a_j-го столбца, служит множество всех входных сигналов, вызывающих переход автомата из состояния a_i в состояние a_j (такое множество может быть, разумеется, и пустым).

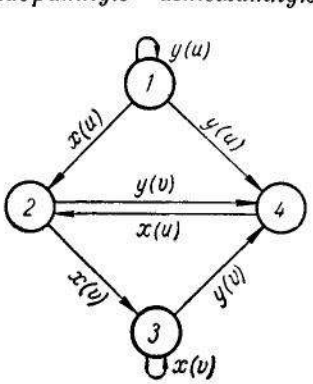

Рис 3

Если фиксировать какую-либо нумерацию состояний автомата натуральными числами 1, 2, ... и расположить строки и столбцы квадратной автоматной таблицы в соответствии с этой нумерацией, то квадратная автоматная таблица превращается в (*квадратную*) *автоматную матрицу* $\|\alpha_{ij}\|$.

В отличие от таблицы в матрице нет необходимости выписывать состояния, обозначающие строки и столбцы,— их заменяет номер строки или столбца, определяемый их положением в матрице. Разумеется, при изменении нумерации состояний изменяется и сама матрица, так что одна и та же функция переходов может задаваться различными квадратными автоматными матрицами. Напомним лишь, что, в соответствии с принятым выше соглашением, при любой нумерации начальному состоянию автомата приписывается номер, равный единице.

Как уже отмечалось выше, элементами квадратной автоматной матрицы служат множества входных сигналов $x_1, \ldots, x_n$ автомата. Для обозначения множества, состоящего из сигналов $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$, будем употреблять символы этих сигналов, соединенные знаком *дизъюнкции* (логического «или»): $x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}$. В частности, множество, состоящее из одной буквы, будем отождествлять с этой буквой. Для обозначения пустого множества будем употреблять специальный символ (звездочку или черточку), либо просто оставлять в соответствующих местах матрицы пустые места. Пользоваться символом нуля для этой цели неудобно, поскольку в ряде случаев естественно употреблять нуль для обозначения одного из входных сигналов.

Имея таблицу переходов или граф автомата и зафиксировав определенную нумерацию его состояний, легко построить квадратную автоматную матрицу этого автомата. Так, например, для автомата, граф которого изображен на рис. 1, при использовании фактически выполненной на ней нумерации состояний квадратная автоматная матрица будет иметь следующий вид:

$$\begin{Vmatrix} y & x & — & — \\ — & — & x & y \\ — & — & x & y \\ y & x & — & — \end{Vmatrix}.$$

Обратно, по квадратной автоматной матрице нетрудно восстановить таблицу переходов или его граф (без обозначения стрелок и вершин выходными сигналами).

Следует отметить одну особенность способов задания автоматов. В то время как произвольным образом заполненная таблица переходов или таблица выходов служит таб-

лицей переходов или соответственно таблицей выходов некоторого абстрактного автомата, не всякий граф и не всякая квадратная матрица описанного выше вида могут служить графом или соответственно квадратной автоматной матрицей некоторого автомата. Для того чтобы это имело место, необходимо соблюдение некоторых условий, связанных со свойствами функции переходов автомата.

Первое условие связано с однозначностью функций переходов и выходов и называется в соответствии с этим *условием однозначности*. Соблюдение этого условия требует, чтобы на графе автомата из любой вершины выходило бы не более одной стрелки, обозначенной любым данным входным сигналом. Применительно к квадратной автоматной матрице это условие означает, очевидно, что в любой ее строке любой данный входной сигнал должен встречаться не более одного раза.

Второе условие, называемое обычно *условием полной определенности*, определяется тем, что до настоящего времени мы рассматривали лишь такие автоматы, функции переходов и выходов которых (обычные или сдвинутые) всюду определены. Соблюдение этого условия требует, очевидно, чтобы для любого входного сигнала x из каждой вершины обязательно выходила бы стрелка, обозначенная этим входным сигналом, и чтобы этот входной сигнал обязательно присутствовал в каждой строке квадратной автоматной матрицы.

В настоящей книге мы не будем рассматривать такие обобщения абстрактных автоматов, для которых не выполняется условие однозначности. Вместе с тем, при синтезе цифровых автоматов весьма часто и с большой пользой употребляются такие обобщения понятия абстрактного автомата, для которых не выполняется условие полной определенности. Это — так называемые *частичные автоматы*.

Частичным автоматом называется абстрактный автомат, у которого функция переходов или функция выходов (обычная или сдвинутая), или обе эти функции, определены не для всех пар значений своих аргументов a *и* x.

В отличие от частичных автоматов, рассматривавшиеся ранее автоматы называются *вполне определенными*. В дальнейшем изложении будет удобно под термином *автомат*

понимать произвольный (частичный или вполне определенный) абстрактный автомат. Для частичных автоматов употребляются те же способы задания, что и для вполне определенных автоматов. В тех местах таблицы переходов или таблицы выходов (обычной или сдвинутой), в которых изображаемые ими функции не определены, мы будем ставить черточки. Граф частичного автомата может иметь вершины, из которых не выходят стрелки, обозначенные теми или иными входными сигналами.

Подобно вполне определенным автоматам частичные автоматы также делятся на автоматы первого и второго рода, автоматы Мили и автоматы Мура. *Изоморфизм* частичных автоматов предусматривает, чтобы соответствующее изоморфное отображение переводило множества значений, в которых не определены функции переходов или выходов (обычная или сдвинутая) одного автомата, в множества значений, в которых не определены соответствующие функции другого автомата.

Несколько сложнее обстоит дело с понятием *эквивалентности* частичных автоматов. Для того чтобы определить это понятие, необходимо прежде всего установить, что́ следует понимать под отображением, индуцируемым данным частичным автоматом A. Для этой цели, как и прежде, мы будем подавать на вход частичного автомата (приведенного предварительно в начальное состояние) различные входные слова. Подавая очередную букву $x(k)=x_{i_k}$ некоторого слова $p=x_{i_1}x_{i_2}\ldots x_{i_k}$ на вход автомата A, мы можем столкнуться с положением, когда соответствующий ей выходной сигнал не определен. Условимся в этом случае называть соответствующее слово p — *запрещенным* для данного частичного автомата A. Все слова, не являющиеся запрещенными, будем называть *допустимыми* (для данного автомата A).

Отображение φ множества входных слов в множество выходных слов, индуцируемое частичным автоматом A, является, таким образом, *частичным отображением*, областью определения которого служит множество всех допустимых слов данного автомата. Образ $\varphi(p)$ любого допустимого слова при этом отображении определяется точно так же, как и для вполне определенных автоматов. В дальнейшем нам будет удобно под термином «отображение» пони-

мать произвольное (вполне определенное или частичное) отображение.

Два частичных автомата с одинаковыми входным и выходным алфавитами естественно называть *эквивалентными*, если индуцируемые ими частичные отображения имеют одну и ту же область определения и совпадают на этой области.

Для случая частичных автоматов, однако, большее значение имеет не отношение эквивалентности, а отношение *эквивалентного продолжения* автоматов.

1. 4. *Говорят, что частичное отображение* φ *продолжает частичное отображение* ψ, *если область определения отображения* φ *включает в себя область определения* M *отображения* ψ, *а на области* M *оба отображения совпадают. Частичный автомат* A *называется эквивалентным продолжением частичного автомата* B, *если частичное отображение, индуцируемое автоматом* A, *продолжает частичное отображение, индуцируемое автоматом* B.

Ясно, что в случае, когда автомат A эквивалентно продолжает автомат B, входной алфавит автомата A должен содержать входной алфавит автомата B. То же самое будет иметь место и для выходных алфавитов, так как в дальнейшем мы будем всегда предполагать, что выходные слова, выдаваемые частичным автоматом в ответ на всевозможные входные слова, используют все буквы его выходного алфавита. В дальнейшем нам придется иметь дело в основном с такими эквивалентными продолжениями, при которых не происходит расширения входного и выходного алфавитов.

Из определения допустимых слов (для любого данного частичного автомата A) вытекает, что в случае, когда входное слово $p = x_{i_1} x_{i_2} \dots x_{i_k}$ допустимо, допустимыми будут также все слова x_{i_1}, $x_{i_1} x_{i_2}$, $\dots$, $x_{i_1} x_{i_2} \dots x_{i_{k-1}}$. Все эти слова называются *начальными отрезками* слова p. Обычно к числу начальных отрезков слова p причисляют само это слово, а также *пустое слово*, не содержащее ни одной буквы. Сделанное замечание позволяет сформулировать следующее предложение.

1. 5. *Область определения* M *частичного отображения, индуцируемого любым частичным автоматом, удовлетворяет следующему условию, называемому условием полноты:*

вместе с любым содержащимся в M словом область M содержит и все начальные отрезки этого слова.

Рассмотрим теперь некоторые соотношения между областями определений функций переходов и выходов в частичном автомате. Предположим, что мы имеем частичный автомат Мили A, и пусть для некоторого состояния a_i и входного сигнала x_j его функция выходов не определена. Это означает, очевидно, что переход автомата из состояния a_i в новое состояние под воздействием входного сигнала x_j реализуется лишь при подаче на вход автомата запрещенных входных слов. Поэтому мы можем не определять функцию переходов $\delta(a, x)$ автомата A в точке $a=a_i$, $x=x_j$ без опасения изменить реакцию автомата A на допустимые слова. Считая функцию $\delta(a, x)$ неопределенной в этой точке, мы не увеличим и не уменьшим множества допустимых слов автомата.

Проведенными рассуждениями нами доказано следующее предложение.

1. 6. *Если функцию переходов произвольного частичного автомата Мили A рассматривать не заданной во всех точках, в которых не задана его функция выходов, то возникший таким образом новый частичный автомат Мили B будет индуцировать то же самое частичное отображение, что и автомат A.*

При задании (частичного) автомата A таблицами переходов и выходов таблица переходов автомата B получается прочеркиванием всех мест таблицы переходов автомата A, которым соответствуют прочеркнутые места в его таблице выходов. Назовем эту операцию *распространением неопределенности таблицы выходов на таблицу переходов* автомата Мили.

Для частичных автоматов Мура естественно считать *запрещенными* все состояния, для которых сдвинутая функция выходов не определена. Легко понять, что перейти из начального состояния в запрещенное под воздействием непустого слова автомат Мура может лишь в том случае, когда это слово — запрещенное. Можно поэтому не определять функцию переходов автомата Мура всякий раз, когда функция принимает запрещенное значение, не изменяя при этом индуцируемого автоматом отображения (включая область определения этого отображения).

Нами доказано следующее предложение.

1. 7. *Не изменяя индуцируемого частичным автоматом Мура частичного отображения, можно считать функцию переходов этого автомата неопределенной всякий раз, когда она принимает запрещенное значение.*

Эквивалентное преобразование автоматов Мура, устанавливаемое предложением 1.7, сводится к тому, что в таблице переходов автомата Мура символы запрещенных состояний заменяются черточками. Как и в случае автоматов Мили, мы будем называть эту операцию *распространением неопределенности на таблицу переходов.*

Введем еще следующее определение.

1. 8. *Состояние a абстрактного (частичного или вполне определенного) автомата называется достижимым, если оно совпадает с начальным состоянием или если существует такое достижимое состояние b и такой входной сигнал x, что $a = \delta(b, x)$. В противном случае состояние a называется недостижимым. Автомат, все состояния которого достижимы, называется связным.*

Нетрудно понять, что автомат не может перейти в недостижимое состояние из начального состояния под воздействием допустимых входных слов. Можно поэтому вычеркнуть все столбцы таблиц переходов и выходов автомата, которые обозначены недостижимыми состояниями, не изменяя при этом индуцируемое автоматом частичное отображение. Условимся называть такую операцию *исключением недостижимых состояний.*

1. 9. *В результате исключения недостижимых состояний индуцируемое абстрактным автоматом (частичным или вполне определенным) отображение не изменяется.*

Предложение 1.9 показывает, что при изучении индуцируемых автоматами отображений можно ограничиться рассмотрением лишь *связных* автоматов.

Условимся еще относительно некоторых обозначений. Если A — произвольный абстрактный автомат, $\delta(a,x)$ — его функция переходов, а p — произвольное входное слово, то через $\delta(a, p)$ мы будем обозначать то состояние, в которое перейдет автомат A из состояния a, если на вход автомата последовательно, буква за буквой, подать все буквы слова p. Для обозначения состояния $\delta(a,p)$ мы будем иногда употреблять также сокращенное обозначение ap,

так что, в частности, функция переходов автомата может быть записана просто в виде выражения ax.

В заключение сделаем несколько замечаний, которые следует иметь в виду при практическом использовании введенных в настоящем параграфе понятий. Во-первых, нужно помнить, что всякий абстрактный автомат (в том числе и частичный) получает входные сигналы в о в с е моменты времени $t=1, 2, \ldots$ Это обстоятельство нисколько не сужает области применения абстрактных автоматов. В самом деле, если некоторый реальный автомат в какие-то моменты времени не получает никаких реальных сигналов, то можно ввести для такого случая специальный символ нулевого сигнала, так что рассматриваемый как абстрактный автомат наш автомат будет получать входные сигналы во все моменты времени.

Далее, необходимо помнить, что любая комбинация сигналов, передаваемых одновременно по реальным входным каналам, рассматривается в абстрактной теории как один входной сигнал. Если, например, реальный автомат имеет два входных канала, по которым могут передаваться три комбинации сигналов 00, 01 и 10, то, рассматривая этот автомат как абстрактный, мы должны ввести для него три (абстрактных) входных сигнала x_1, x_2, x_3, сопоставив их каким-нибудь образом заданным комбинациям реальных входных сигналов.

Еще одно замечание касается способов практического использования понятия частичного автомата. Обычно все реально существующие автоматы являются вполне определенными. Однако они могут быть поставлены в такие условия, что некоторые слова их входного алфавита никогда не подаются на их вход. Такие слова естественно называть запрещенными. Ясно, что с точки зрения правильности работы автомата совершенно безразлично, как он будет реагировать на запрещенные слова. Поэтому переходы и выходы автомата применительно к запрещенным словам могут быть определены совершенно произвольно.

Беря соответствующий частичный автомат, где эти переходы и выходы не определены вовсе, мы тем самым оставляем за собой право определить их впоследствии так, как это нам будет удобно. Таким образом, на практике частичный автомат A выступает как способ представления

целого класса вполне определенных автоматов, которые могут быть получены из автомата A в результате произвольного доопределения его функций переходов и выходов во всех точках, в которых они не были ранее определены.

§ 2. Автоматные отображения и события

Отображения (вообще говоря, частичные), индуцируемые абстрактными автоматами, мы будем называть *автоматными отображениями*. Из предложения 1.5 предыдущего параграфа и из способа определения автоматного отображения непосредственно вытекает, что всякое автоматное отображение удовлетворяет следующим четырем условиям:

1. *Автоматное отображение осуществляет однозначное отображение (вообще говоря, частичное) множества слов в некотором конечном алфавите $\mathfrak{X}$ (называемом входным алфавитом отображения) в множество слов в некотором конечном алфавите $\mathfrak{Y}$ (называемом выходным).*

2. *Область определения автоматного отображения удовлетворяет условию полноты, то есть, иначе говоря, вместе с любым содержащимся в ней словом содержит также все начальные отрезки этого слова. Пустое слово всегда входит в область определения отображения.*

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

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

Назовем эти условия условиями автоматности отображения.

Все слова входного алфавита разбиваются автоматным отображением на два класса: на класс *допустимых* и класс *запрещенных* слов, в зависимости от того, входят или не входят они в область определения этого отображения. Совокупность всех запрещенных для данного автоматного отображения слов будет называться его *областью запрета*.

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

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

Выбирая в качестве начального состояния автомата пустое слово e, мы получим, как легко видеть, автомат Мура, индуцирующий исходное отображение φ. В самом деле, пусть $q = x_i x_i \ldots x_{i_k}$ — любое допустимое слово отображения φ. Тогда все его начальные отрезки будут также допустимыми словами (в силу условия 2). Подавая на вход построенного автомата слово q, мы будем осуществлять последовательно его перевод в состояния $ex_{i_1} = x_{i_1}$, $x_{i_1} x_{i_2}$, ..., $x_{i_1} x_{i_2} \ldots x_{i_k}$. При этом автомат выдает некоторую последовательность выходных букв $y_{j_1} y_{j_2} \ldots y_{j_k} = p$. Из способа определения нашего автомата вытекает, что последняя буква слова p совпадает с последней буквой слова $\psi(q)$, предпоследняя буква слова p — с последней буквой слова $\varphi(x_{i_1} x_{i_2} \ldots x_{i_{k-1}})$, которая, в силу условия 4, совпадает с предпоследней буквой слова $\varphi(q)$ и т. д. Продолжая подобным образом и используя условие 3, мы установим совпадение всех букв слова p с соответствующими им буквами слова $\varphi(q)$. Следовательно, построенный нами автомат Мура A индуцирует исходное (частичное) отображение φ. Поскольку мы можем интерпретировать автомат A как автомат Мили (см. § 1), нами доказано следующее предложение.

2. 1. *Если алфавитное отображение* φ *удовлетворяет сформулированным выше четырем условиям автоматности, то можно построить автоматы Мили и Мура (вообще говоря, бесконечные), индуцирующие это отображение. В случае, когда область определения отображения* φ *конечна, эти автоматы также могут быть выбраны конечными.*

Доказанное предложение позволяет нам применять термин «автоматное отображение» ко всякому алфавитному отображению, удовлетворяющему условиям автоматности.

Из доказательства предложения 2.1 вытекает существование единого конструктивного приема, позволяющего по любому автоматному отображению с конечной областью определения (заданному на конечном множестве слов) строить индуцирующий это отображение конечный автомат Мили или Мура. Указанный прием восходит еще к работам Хафмена [1]) и Мили [2]). Нашей же целью является разработка более общего конструктивного приема, позволяющего строить конечные автоматы Мили и Мура, индуцирующие заданное автоматное отображение не только для случая конечной области определения, а в с я к и й р а з, к о г д а т а к и е а в т о м а т ы с у щ е с т в у ю т.

Из предложения 2.1 вытекает, что классы отображений, индуцируемых произвольными (не обязательно конечными) автоматами Мили и Мура, совпадают между собой: как один, так и другой класс состоят из всех автоматных отображений. Оказывается, что имеет место также совпадение классов отображений, индуцируемых конечными автоматами Мили и Мура. Поскольку в § 1 уже была установлена возможность интерпретации каждого автомата Мура как автомата Мили с тем же самым числом состояний, то для обоснования справедливости высказанного утверждения достаточно доказать следующую теорему.

2. 2. *Для всякого конечного автомата Мили существует эквивалентный ему (индуцирующий то же самое*

---

[1]) D. A. Huffman, The synthesis of sequential switching circuits. Journal of the Franklin Inst., v. 257, № 3, 4, 1954, p. 161—190, 275—303.

[2]) G. H. Mealy, A method for synthesizing sequential circuits. Bell System Tech J., v. 34, 1955, p. 1045—1079.

отображение) конечный автомат Мура. Существует единый конструктивный прием (универсальный алгоритм преобразования автоматов Мили в автоматы Мура), позволяющий по произвольному конечному автомату Мили, имеющему m входных сигналов и n состояний, построить эквивалентный ему автомат Мура, имеющий не более mn+1 состояний.

В самом деле, пусть A — конечный автомат Мили с множеством состояний $(a_1, \ldots, a_n)$, входным алфавитом $(x_1, \ldots, x_m)$, выходным алфавитом $(y_1, \ldots, y_r)$, начальным состоянием a_1, функцией переходов $\delta(a, x)=ax$ и функцией выходов $\lambda(a, x)$. Построим автомат Мура B, состояниями которого являются начальное состояние $b_1=a_1$ (совпадающее с начальным состоянием автомата A) и всевозможные пары $b_{ij}=(a_i, x_j)$ $(i=1, \ldots, n;\ j=1, \ldots, m)$.

Функцию переходов автомата B определим следующим образом: $b_1x_k=b_{1k}=(a_1, x_k)$, $b_{ij}x_k=(a_i, x_j)x_k=(a_ix_j,\ x_k)$ $(i=1,2,\ldots,\ n;\ j,\ k=1,2,\ \ldots,\ m)$. В случае, когда $a_i\,x_j$ не определено, неопределенным является также и $b_{ij}x_k$. Сдвинутую функцию выходов $\mu(b)$ автомата B определим следующим образом. значение $\mu(b_1)$не определено; значение $\mu(b_{ij})$ определено и равняется $\lambda(a_i, x_j)$ всякий раз, когда значение $\lambda(a_i, x_j)$ определено, и не определено — в противном случае.

Из самого способа построения автомата B непосредственно вытекает, что он индуцирует то же самое отображение, что и автомат A. В самом деле, пусть $p=x_{i_1}x_{i_2}\ldots x_{i_t}$ — произвольное допустимое входное слово автомата A. Построим последовательность состояний $a_1, a_{j_1}, a_{j_2}, \ldots, a_{j_{t-1}}$, в которые автомат A переходит из начального состояния a_1 под влиянием последовательных начальных отрезков слова p. Тогда соответствующая последовательность состояний автомата B будет иметь вид: a_1, (a_1, x_{i_1}), (a_{j_1}, x_{i_2}), ..., $(a_{j_{t-1}}, x_{i_t})$. Теперь, в силу определения сдвинутой функции выходов автомата B, очевидно, что автоматы A и B под влиянием слова p выдадут одно и то же выходное слово.

Пусть теперь слово p является запрещенным в автомате A. Это означает, что существует номер $s<t$ такой, что либо $\lambda(a_{j_s}, x_{i_{s+1}})$, либо $\delta(a_{j_{s-1}}, x_{i_s})$ не определены. В первом случае на состоянии $(a_{j_s}, x_{i_{s+1}})$ не будет определен выход-

ной сигнал автомата B, а во втором — не определен переход от состояния $b_{j_{s-1}i_s}=(a_{j_{s-1}}, x_{i_s})$ посредством входного сигнала $x_{i_{s+1}}$. В обоих случаях слово p оказывается запрещенным и для автомата B.

Итак, отображения, индуцируемые автоматами A и B, действительно совпадают. Автомат Мура B имеет при этом $mn+1$ состояние. После применения к нему операций распространения неопределенности на таблицу переходов и исключения недостижимых состояний (см. 1.7 и 1.8) число состояний может уменьшиться без изменения индуцируемого им соответствия. Тем самым предложение 2.2 полностью доказано.

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

Пусть φ — произвольное (вообще говоря,— частичное) отображение множества слов в (конечном) алфавите $\mathfrak{X}$ в множество слов в (конечном) алфавите $\mathfrak{Y}$. Обозначим через P область определения этого отображения. Будем применять к отображению φ две операции. Первая из них, которую мы назовем *операцией выравнивания длин слов* (или, кратко, *операцией выравнивания*), заключается в следующем. Во входной и выходной алфавиты отображения φ (то есть в $\mathfrak{X}$ и $\mathfrak{Y}$) добавляется по одной новой букве, которые мы будем называть *пустыми* и обозначать через α и β (иногда в качестве таких букв могут быть выбраны буквы алфавитов $\mathfrak{X}$ и $\mathfrak{Y}$; не исключено также совпадение букв α и β); к любому слову p из P приписывается с п р а в а m_p экземпляров букв α, а к его образу $q=\varphi(p)$ приписывается слева n_q экземпляров букв β так, чтобы длины полученных в результате приписывания новых букв слов p_1 и q_1 совпали.

Далее строится новое отображение φ_1 некоторого множества P_1 слов в алфавите $\mathfrak{X} \cup (\alpha)$ в множество слов в алфавите $\mathfrak{Y} \cup (\beta)$, которое переводит друг в друга слова p_1 и q_1,

полученные в результате выравнивания длин слов p и q соответственно: $\varphi_1(p_1)=q_1$ (p пробегает при этом все множество P).

Условимся говорить, что отображение φ_1 получается из отображения φ с помощью операции выравнивания.

Отображение φ однозначным образом восстанавливается по отображению φ_1, однако само отображение φ_1 может быть построено многими различными способами. Мы будем поэтому применять наряду с произвольной также и некоторую *стандартную операцию выравнивания*, при которой отображение φ_1 однозначно определяется отображением φ и присоединенными пустыми буквами α и β. Сущность этой операции заключается в том, что число m_p пустых букв α, приписываемых справа к произвольному слову p из области определения отображения φ, принимается равным длине слова $q=\varphi(p)$, а число n_q пустых букв β, приписываемых слева к слову q, принимается равным длине слова p. Ясно, что такое приписывание действительно приводит к выравниванию длин слов.

Вторая операция применяется только к выравненным алфавитным отображениям φ, то есть к таким отображениям, у которых длины входных и соответствующих им выходных слов равны между собой. Сущность этой операции, которую мы будем называть *операцией пополнения* отображения φ, заключается в распространении отображения φ на начальные отрезки слов. Операция пополнения имеет следующий вид: если s — произвольный начальный отрезок любого слова p из области определения отображения φ, то мы полагаем $\varphi(s)$ равным начальному отрезку слова $\varphi(p)$, имеющему равную с начальным отрезком s длину.

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

Если пополнение φ' отображения φ является однозначным, то оно удовлетворяет, очевидно, всем четырем условиям автоматности. К сожалению, однако, в большинстве случаев пополнение отображения φ неоднозначно, пос-

кольку одно и то же слово s может входить в качестве начального отрезка в несколько различных слов из области определения отображения φ. Вместе с тем справедливо следующее предложение.

2. 3. *В случае, когда отображение φ получено из некоторого однозначного алфавитного отображения в результате стандартной операции выравнивания длин слов, пополнение φ' этого отображения однозначно и является автоматным отображением.*

Действительно, пусть $p_1 = p\,\alpha\alpha\ldots\alpha$ — произвольное слово из области определения P отображения φ; $\varphi(p_1) = q_1 = \beta\beta\ldots\beta q$ — его образ при отображении φ; p — слово длины m, не содержащее пустой буквы α; q — слово длины n, не содержащее пустой буквы β; p_2 — произвольный начальный отрезок слова p_1.

Рассмотрим два случая. Пусть сначала слово p_2 содержит хотя бы одну пустую букву α. В этом случае слово p_2 имеет вид $p\alpha\alpha\ldots\alpha$. Но задание любого слова такого вида однозначно определяет слово p, а, следовательно, и слово p_1. Таким образом в этом случае слово p_2 совпадает с начальным отрезком одного лишь слова p_1, так что его образ $\varphi'(p_2)$ в пополненном отображении определяется однозначно.

Рассмотрим теперь второй возможный случай, то есть случай, когда слово p_2 не содержит ни одной буквы α. В этом случае, в силу определения стандартной операции выравнивания, соответствующий начальный отрезок слова q_1 будет состоять лишь из пустых букв, так что $\varphi'(p_2) = \beta\beta\ldots\beta$. Так будет, очевидно, не только для слова p_1, но и для любого слова в P, начальным отрезком которого является слово p_2. Следовательно, однозначность определения образа $\varphi'(p_2)$ будет иметь место и в этом случае.

Область определения P' отображения φ' содержит все начальные отрезки слов из P. Так как всякий начальный отрезок любого начального отрезка произвольного слова из P сам является начальным отрезком этого слова, то область P' удовлетворяет условию полноты. Таким образом, для отображения φ' выполняется второе условие автоматности. Выполнимость первого условия автоматности была доказана вместе с установлением однозначности отображения φ'. Выполнимость третьего условия очевидна.

Наконец, пусть p_3 — произвольный начальный отрезок любого слова p_2 из P'. Так как p_2, в свою очередь, является начальным отрезком некоторого слова p_1 из P, то p_3 также совпадает с каким-то начальным отрезком слова p_1. В силу способа построения отображения φ' слову p_2 относится в качестве образа некий начальный отрезок q_2 слова $\varphi(p_1)$, а слову p_3 — начальный отрезок q_3 того же слова, имеющий меньшую, чем отрезок q_2, длину. Так как q_3 может рассматриваться в качестве начального отрезка слова q_2, то выполнимость четвертого условия автоматности для отображения φ' доказана. Вместе с тем завершено доказательство предложения 2.3.

В качестве следствия предложения 2.3 можно сформулировать следующее предложение.

2. 4. *Любое однозначное алфавитное отображение можно превратить в автоматное отображение, применяя к этому отображению операцию выравнивания длин слов и операцию пополнения.*

Заметим, что в ряде случаев при превращении заданного отображения в автоматное отображение можно применять не стандартную операцию выравнивания, а какой-нибудь более экономный (с точки зрения числа дописываемых букв) вариант операции выравнивания. В частности, если само исходное отображение было автоматным, то можно считать, что применяется н у л е в а я операция выравнивания, при которой никакого дописывания пустых букв вообще не происходит.

Именно поэтому в предложении 2. 4 говорится о произвольной, а не о стандартной операции выравнивания.

В дальнейшем соответственно тому, применяется ли стандартная или нестандартная операция выравнивания, мы будем говорить о *стандартном* или *нестандартном* приеме сведения произвольного отображения к автоматному.

Обычно на практике поступают следующим образом. Сначала операцию выравнивания проводят наиболее экономным образом и, применяя затем операцию пополнения, проверяют (по признаку однозначности пополнения), получилось ли в результате автоматное отображение? Если нет, то производят новые дописывания пустых букв, **новую проверку пополнением и т. д. Предложение 2.3**

дает гарантию того, что в результате продолжения подобного процесса мы обязательно придем к автоматному отображению. Назовем этот метод приведения произвольного отображения к автоматному отображению *методом последовательного приведения* в отличие от *метода стандартного приведения*, описанного в предложении 2.3.

Остановимся теперь на способах задания автоматных отображений. Разумеется, можно задавать автоматное отображение индуцирующим его автоматом. Однако на практике при постановке той или иной задачи, относящейся к синтезу автомата, такой способ задания является конечной целью, а не исходным пунктом.

В случае, когда область определения автоматного отображения φ конечна, его чаще всего принято задавать с помощью так называемой *таблицы соответствия*. В левой половине этой таблицы выписывают в том или ином порядке входные слова, на которых данное отображение φ задано, а в правой части таблицы — соответствующие им выходные слова. Знаком раздела между каждым входным словом p и соответствующим ему выходным словом $\varphi(p)$ служит обычно стрелка, так что каждая строка таблицы соответствия имеет вид $p \to \varphi(p)$.

Используя четвертое условие автоматности, в таблицу соответствия обычно не включают входные слова, являющиеся начальными отрезками слов, уже включенных в таблицу. Таблицу соответствия, в которой никакое входное слово не является начальным отрезком какого-либо другого входного слова, мы будем называть *сокращенной таблицей соответствия*. При задании автоматного отображения сокращенной таблицей соответствия его продолжение на начальные отрезки входных слов, вошедших в таблицу, осуществляется на основании четвертого условия автоматности. Таким образом, при желании по сокращенной таблице соответствия всегда можно восстановить полную таблицу соответствия. В ряде случаев оказывается удобным рассматривать частично сокращенные таблицы соответствия (то есть таблицы соответствия, из которых исключена лишь часть входных слов, являющихся начальными отрезками других слов).

В ряде случаев удается задавать таблицей соответствия и такие автоматные отображения, область определения

которых бесконечна. Например, таблица соответствия,

$$xy \to uv$$
$$\underbrace{yxx\ldots x}_{n\text{ раз}} \to \underbrace{vvuu\ldots u}_{(n-1)\text{ раз}},$$

где n пробегает все целые положительные значения ($n=1,2,\ldots$), задает, очевидно, автоматное отображение с бесконечной областью определения.

Очень удобным способом задания произвольного автоматного отображения является задание его с помощью так называемого *автоматного множества событий*.

2. 5. *Событием в любом данном алфавите* $\mathfrak{X}$ *мы будем называть произвольное множество слов в этом алфавите, а сам алфавит* $\mathfrak{X}$ *— входным алфавитом этого события.*

2. 6. *Для любого алфавитного отображения* φ *с входным алфавитом* $\mathfrak{X}$ *и выходным алфавитом* $\mathfrak{Y}$, *и для любой выходной буквы (буквы выходного алфавита)* y *событие* S_y, *состоящее из всех слов входного алфавита, образы (при отображении* φ*) которых оканчиваются буквой* y, *будем называть событием, представленным в отображении* φ *выходной буквой* y.

Множество событий S_{y_i}, *где* y_i *пробегает все буквы выходного алфавита* $\mathfrak{Y}$, *называется каноническим множеством событий, соответствующим отображению* φ. *При этом говорят, что буква* y_i *выходного алфавита отмечает соответствующее ей событие* S_{y_i}.

Задание любого (однозначного) алфавитного отображения φ однозначно определяет соответствующее ему каноническое множество событий. Обратное, вообще говоря, неверно и имеет место лишь для автоматных отображений.

Действительно, пусть φ — произвольное автоматное отображение, а $p=x_{i_1}x_{i_2}\ldots x_{i_k}$ — произвольное входное слово, входящее в область определения этого отображения. Обозначим через $q=y_{j_1}y_{j_2}\ldots y_{j_k}$ образ $\varphi(p)$ слова p при отображении φ, и покажем, что этот образ однозначно определяется заданием канонического множества M событий $S_{y_1},\ldots,S_{y_m}$, соответствующего отображению φ. Для этой цели определим события множества M, в которые входят последовательные начальные отрезки слова p: x_{i_1}, $x_{i_1}x_{i_2}$, $\ldots$, $x_{i_1}x_{i_2}\ldots x_{i_k}$. В силу четвертого условия автоматности

этими событиями будут события $S_{y_{j_1}}, S_{y_{j_2}}, \ldots, S_{y_{j_t}}$, задание которых позволяет однозначно (по индексам y_{i_k} этих событий) восстановить выходное слово $\varphi(p)$. Ясно также, что отображение φ не определено на слове q всякий раз, когда слово (или даже некоторый его начальный отрезок) не входит в область определения отображения φ.

Нами доказано, таким образом, следующее предложение.

2. 7. *Всякое автоматное отображение однозначно определяется соответствующим ему каноническим множеством событий в предположении, что события этого множества отмечены соответствующими буквами выходного алфавита заданного отображения.*

Будем называть автоматное отображение и соответствующее ему множество событий *соответствующими друг другу*. Предложение 2.7 показывает, что автоматные отображения могут задаваться конечными множествами событий. Условимся называть *автоматными* множества событий, соответствующие автоматным отображениям. Из условий 1—4, которым удовлетворяет всякое автоматное отображение, и из определения автоматных множеств нетрудно вывести следующие условия, которым должно удовлетворять любое автоматное множество событий:

I. *Автоматное множество событий состоит из конечного числа событий в одном и том же конечном алфавите (называемом входным), которые попарно не пересекаются и не содержат пустого слова.*

II. *Если какое-либо слово принадлежит какому-нибудь событию из автоматного множества событий, то все начальные отрезки этого слова, за исключением пустого слова, принадлежат каким-либо событиям из того же самого множества.*

Назовем эти условия *условиями автоматности* данного множества событий. Справедливо следующее предложение.

2. 8. *Если некоторое множество M событий удовлетворяет условиям автоматности, то существует одно и, с точностью до обозначения букв выходного алфавита, только одно автоматное отображение ψ, отличающееся тем, что соответствующее ему каноническое множество событий совпадает с M.*

Действительно, пусть S_1, S_2, ..., S_m — события, составляющие множество M. Определим отображение φ множества слов во входном алфавите данного множества M в множество слов в алфавите $(S_1, S_2, \ldots, S_m)$ следующим образом. Если слово $p = x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ не содержится ни в одном из событий множества M, то значение $\varphi(p)$ считается неопределенным. Если же слово p содержится в каком-либо из данных событий, например в событии S_{j_k}, то найдется (в силу второго условия автоматности) ряд событий S_{j_1}, S_{j_2}, ..., $S_{j_{k-1}}$ множества M такой, что первое событие этого ряда содержит начальный отрезок x_{i_1} слова p, второе событие — начальный отрезок $x_{i_1} x_{i_2}$ и т. д. В этом случае полагаем $\varphi(p) = S_{j_1} S_{j_2} \ldots S_{j_k}$. Ясно, что для построенного таким образом отображения φ будут выполняться все четыре условия автоматности отображений, так что оно будет автоматным отображением. Из способа его построения непосредственно вытекает, что соответствующим ему каноническим множеством событий будет множество M. Единственность отображения φ вытекает из предложения 2.7. Тем самым предложение 2.8 полностью доказано.

Предложение 2.8 позволяет нам в дальнейшем называть автоматным всякое множество событий, удовлетворяющее условиям автоматности I и II.

§ 3. Алгебра событий

Как было установлено в предыдущем параграфе, любое автоматное отображение φ может быть задано конечным множеством M событий во входном алфавите этого отображения. Если область определения отображения φ конечна, то, как нетрудно видеть, все события множества M конечны (т. е. состоят из конечного множества слов). Конечное событие можно задать, перечислив все его элементы. Однако, как уже отмечалось выше, нашей основной целью является рассмотрение отображений с произвольными областями определения. Ясно, что в случае, когда область определения отображения φ бесконечна, хотя бы одно из событий множества M также будет бесконечным. Необходимо поэтому разработать специальный язык,

который позволял бы представлять те бесконечные события, которые нас будут в дальнейшем интересовать, конечными выражениями. Один из возможных путей для построения такого языка открывает так называемая *алгебра событий*.

Следуя терминологии современной абстрактной алгебры, мы будем называть *алгеброй* множество элементов произвольной природы, на котором определены некоторые *конечноместные операции*. Задать конечноместную (или, n-местную) операцию f в алгебре — значит указать способ, позволяющий отнести любому упорядоченному набору из n элементов $g_1, \ldots, g_n$ алгебры элемент $g=f(g_1, \ldots, g_n)$ той же алгебры — результат применения операции f к элементам $g_1, \ldots, g_n$. В так называемых *частичных* алгебрах те или иные алгебраические операции могут быть определены не для всех возможных наборов элементов.

3. 1. *Алгеброй событий в алфавите* $\mathfrak{X}$ *мы будем называть множество* $\mathfrak{A}(\mathfrak{X})$ *всех событий в этом алфавите, на котором заданы две двухместные операции, называемые дизъюнкцией и умножением, и одна одноместная операция, называемая итерацией. Дизъюнкцией* $S_1 \vee S_2$ *двух событий* S_1 *и* S_2 *называется теоретико-множественное объединение событий. Произведением* $S_1 S_2$ *событий* S_1 *и* S_2 *называется событие, состоящее из всех слов вида* pq, *где* p — *любое слово события* S_1, *а* q — *любое слово события* S_2. *Наконец, итерацией* $\{S\}$ *события* S *называется событие, состоящее из пустого слова и всевозможных слов вида* $p_1 p_2 \ldots p_n$, *где* $p_1, p_2, \ldots, p_n$ — *произвольные слова события* S, *а* n — *любое натуральное число* $n=1, 2, \ldots$

Для дальнейшего важно установить определенный порядок действий в алгебре событий. Условимся, что при отсутствии в выражении скобок, изменяющих обычный порядок действий, сначала должны выполняться итерации, затем умножения и, наконец, дизъюнкции. В случае необходимости изменить обычный порядок выполнения операций в выражение вводятся круглые скобки, которые мы будем называть о б ы ч н ы м и, в отличие от фигурных скобок, используемых для обозначения итерации и называемых поэтому и т е р а ц и о н н ы м и. Таким образом, с точки зрения порядка действий в алгебре событий имеется полная аналогия с обычной алгеброй, если считать, что

дизъюнкция соответствует сложению, умножение в алгебре событий — обычному умножению, а итерация — возведению в степень.

Операции в алгебре событий удовлетворяют целому ряду тождественных соотношений, которые можно использовать при преобразованиях и упрощениях формул в этой алгебре. Выпишем некоторые важнейшие соотношения, справедливость которых вытекает непосредственно из определения операций в алгебре событий. В этих формулах через P, Q и S обозначены произвольные события, а через e — событие, состоящее из одного пустого слова.

1. $P \vee Q = Q \vee P$ (коммутативность дизъюнкции).
2. $P \vee (Q \vee S) = (P \vee Q) \vee S$ (ассоциативность дизъюнкции).
3. $P \vee P = P$ (идемпотентность дизъюнкции).
4. $P(QS) = (PQ)S$ (ассоциативность умножения).
5. $P(Q \vee S) = PQ \vee PS$, $(P \vee Q)S = PS \vee QS$ (левая и правая дистрибутивность умножения по отношению к дизъюнкции).
6. $\{\{P\}\} = \{P\}$ (идемпотентность итерации).
7. $\{P\} = e \vee P\{P\}$ (правило развертывания итерации).
8. $P\{P\} = \{P\}P$ (закон коммутативности для итерации).
9. $\{P\}\{P\} = \{P\}$ (закон мультипликативного поглощения для итерации).
10. $\{P\} \vee P = \{P\}$ (закон дизъюнктивного поглощения для итерации).
11. $eS = Se = S$ (закон нейтральности пустого слова).
12. $\{e\} = e$.

Отметим вместе с тем, что умножение в алгебре событий, вообще говоря, некоммутативно: для произвольных событий R и Q справедливо $RQ \neq QR$. Заметим также, что законы ассоциативности дизъюнкции и умножения позволяют записывать без скобок дизъюнкцию и произведение любого конечного числа событий без опасения, что

после этого не вполне определенный порядок выполнения действий может привести к неоднозначности.

Подобно тому, как это имеет место в обычной алгебре, условимся называть *одночленами*, или *термами*, такие выражения в алгебре событий, в которых дизъюнкция не является последним из выполняемых действий. Дизъюнкцию более чем одного терма будем называть *многочленом* в алгебре событий.

Приведем примеры термов и многочленов: выражения $\{R\}(P \vee RS)Q(P \vee R)$, RS, $\{R\}$ будут, очевидно, термами, а выражения $R \vee Q$, $R(Q \vee S) \vee \{R\} \vee Q$ — многочленами. Первый из приведенных многочленов состоит из двух, а второй из трех термов.

Иногда нам придется вводить в алгебру событий и другие операции, которые, однако, мы не будем причислять к числу основных операций. В качестве таких неосновных операций отметим одноместную операцию *дополнения* события и двуместную операцию *пересечения* событий.

Дополнением $\overline{S}$ события S в некотором алфавите $\mathcal{X}$ называется множество всех слов в алфавите $\mathcal{X}$, не входящих в событие S. Пересечением $S_1 \wedge S_2$ событий S_1 и S_2 называется событие, состоящее из всех слов, входящих одновременно в оба события S_1 и S_2.

Перечислим некоторые замечательные события, с которыми придется иметь дело в дальнейшем. Это, во-первых, так называемое *всеобщее событие*, включающее в себя все слова в данном алфавите. *Невозможным*, или *пустым*, *событием*, напротив, называется событие, не содержащее ни одного слова (в том числе и пустого), или, другими словами, событие, состоящее из пустого множества слов. *Одноэлементными событиями* называются события, состоящие из одного слова. В дальнейшем мы условимся отождествлять каждое такое событие с единственным входящим в него словом. *Конечным* называется всякое событие, состоящее из конечного множества слов. Наконец, *полным* событием мы будем называть такое событие, которое, наряду со всяким словом, содержит все начальные отрезки этого слова, исключая лишь, быть может, пустое слово. Операция *пополнения события* заключается в добавлении к нему начальных отрезков всех входящих в него слов.

Особое значение для дальнейших построений имеют так называемые *элементарные* и *регулярные события*.

3.2. *Элементарными событиями в алфавите* $\mathfrak{X}=(x_1, \ldots, x_n)$ *называются* $n+1$ *одноэлементных событий* $x_1, x_2, \ldots, x_n, e$, *где* e — *пустое слово. Любое событие, которое можно получить из элементарных событий в результате применения любого конечного числа раз операций дизъюнкции, умножения и итерации, называется регулярным событием, а всякое его представление через элементарные события и три указанные операции — регулярным выражением.*

Легко видеть, что всякое одноэлементное и всякое конечное событие регулярно: всякое одноэлементное неэлементарное событие (слово) можно представить в виде произведения букв, а всякое конечное событие — в виде дизъюнкции одноэлементных событий. Регулярным будет и всеобщее событие в любом (конечном) алфавите $\mathfrak{X}=(x_1, \ldots, x_n)$. Регулярным выражением для этого события будет итерация дизъюнкции всех букв алфавита: $\{x_1 \vee x_2 \vee \ldots \vee x_n\}$. Невозможное событие мы также будем причислять к числу регулярных событий, считая, что его можно представить, например, в виде дизъюнкции пустого множества элементарных событий.

Заметим, что одно и то же регулярное выражение допускает много различных представлений. Поэтому одной из основных проблем в алгебре событий является проблема эквивалентных преобразований регулярных выражений, то есть таких преобразований, которые не меняют представляемых этими выражениями событий. Такие преобразования можно выполнить, в частности, с помощью выписанных выше тождественных соотношений.

Для регулярных выражений вводится естественное понятие *циклической глубины*. Циклической глубиной регулярного выражения мы будем называть максимальное число вложенных друг в друга пар итерационных скобок. Так, выражение $x\{y \vee \{x\}\}$ имеет циклическую глубину 2, выражение $\{\{\{x\}y\}z\}$ — циклическую глубину 3, выражение $(\{x\} \vee y\{xx\}\{y\})\{z\}$ — циклическую глубину 1, выражение $(x \vee y)y$ — циклическую глубину 0.

Циклическая глубина регулярного события определяется как наименьшая циклическая глубина представляющих это событие регулярных выражений.

Легко видеть, что справедливо следующее предложение.

3.3. *Конечные события, и только они, являются регулярными событиями, имеющими нулевую циклическую глубину.*

Предложение 3.3 свидетельствует о том, что лишь использование итерации позволяет нам строить регулярные выражения для бесконечных событий. Легко заметить, впрочем, что далеко не всякое бесконечное событие регулярно. Это обстоятельство становится ясным из простых теоретико-множественных соображений.

В самом деле, поскольку множество слов в непустом конечном алфавите счетно, то множество всех событий в любом таком алфавите имеет мощность континуума. В то же самое время множество всех регулярных выражений в любом конечном алфавите, как легко проверить, всего лишь счетно.

Впрочем, нетрудно [1]) построить конструктивный пример нерегулярного события. Таким примером является событие S, состоящее из всех слов в непустом конечном алфавите $\mathfrak{X}$, длины которых являются точными квадратами.

В самом деле, если бы указанное событие S было бы регулярным, оно допускало бы представление в виде регулярного выражения. Легко видеть, однако, что регулярное выражение, не содержащее итерации, представляет всегда конечное событие, то есть, иначе говоря, событие, состоящее из конечного числа слов. Поскольку интересующее нас событие S бесконечно, то в представляющем его регулярном выражении обязательно встречаются итерационные скобки. Выделим терм с парой таких скобок, не содержащихся внутри других итерационных скобок. Содержимое этих скобок, которое мы обозначим через P, в силу тождественного соотношения 12 (стр. 64) может считаться отличным от пустого слова e. Следовательно, в событии P содержится хотя бы одно слово h, имеющее ненулевую длину $p > 0$.

[1]) Следуя С. К. Клини (см. его статью «Представление событий в нервных сетях и конечных автоматах», опубликованную в сб «Автоматы», ИЛ, М., 1956, стр. 15—67).

Пусть теперь q есть длина какого-нибудь наугад выбранного слова из события, регулярное выражение для которого получается из выделенного терма заменой итерационной скобки $\{P\}$ пустым словом e. Так как $e \in P$, то это слово содержится в событии S. Согласно определению итерации в событии S будут содержаться события, получающиеся в результате замены скобки $\{P\}$ словом h, словом hh и т. д. до бесконечности. Отсюда уже непосредственно следует, что в событии S содержатся слова длины q, $q+p$, $q+2p$, ..., $q+np$, ...

В силу определения события S все числа $q+np$ (где $n=0, 1, 2, \ldots$) должны представлять собою точные квадраты. Ясно, однако, что это в действительности невозможно. В самом деле, если $q+np=m^2$, то $q+(n+1)p=(m+k)^2$, где $k \geqslant 1$. Вычитая эти неравенства почленно друг из друга, мы придем к соотношению $(m+k)^2-m^2=p$ или, что то же самое, к соотношению

$$2mk+k^2=p.$$

Это соотношение должно выполняться при любом сколь угодно большом n и, следовательно, при сколь угодно большом m (так как $m=\sqrt{q+pn}$). Тем самым мы придем к противоречию, поскольку в правой части соотношения $2km+k^2=p$ стоит фиксированное число, а левая часть может быть сделана сколь угодно большой. Таким образом доказано, что событие S не может быть регулярным.

Построенный пример может быть легко обобщен. Тот же самый прием, который уже был использован в этом примере, позволяет доказать нерегулярность события $S(N)$ в любом непустом алфавите, содержащего всевозможные слова, длины которых составляют произвольную последовательность $N=n_1, n_2, \ldots, n_k, \ldots$, удовлетворяющую единственному условию: разности $n_{k+1}-n_k (k=1, 2, \ldots)$ не ограничены в совокупности, то есть, иными словами, для любого сколь угодно большого числа M найдется такой номер k, что $n_{k+1}-n_k > M$.

Несмотря на то, что класс регулярных событий охватывает далеко не все бесконечные события, он тем не менее оказывается достаточным для описания всех автоматных отображений, индуцируемых конечными автоматами. Заметим также, что использование при построении

регулярных выражений наряду с операциями дизъюнкции, умножения и итерации также и операций дополнения, пересечения и пополнения не выводит за пределы класса регулярных событий.

§ 4. Представление событий в автоматах

Основной задачей абстрактной теории автоматов является установление связей, существующих между автоматами и индуцируемыми ими отображениями. Как было показано в § 2 настоящей главы, произвольное автоматное отображение можно задавать представляемыми этим отображением событиями. В связи с этим возникает задача установления связей между событиями и автоматами. Исходным пунктом для установления таких связей будет служить следующее определение.

4.1. *Для произвольного автомата A множество S_y всех входных слов, вызывающих появление выходных слов, которые оканчиваются одной и той же буквой (выходным сигналом) y, называется событием, представленным в автомате A выходным сигналом y. Множество, состоящее из событий S_y для всех букв выходного алфавита автомата A, называется каноническим множеством событий данного автомата A.*

Из приведенного определения и определения 2.6 непосредственно следует, что каноническое множество событий любого автомата A совпадает с каноническим множеством событий для индуцируемого этим автоматом отображения. Как и в § 2, мы будем предполагать, что события, составляющие каноническое множество событий автомата, о т м е ч е н ы соответствующими им выходными сигналами.

Далее, справедливо следующее очевидное предложение.

4.2. *Каноническое множество событий любого автомата является автоматным множеством событий. Обратно, для любого автоматного множества событий M существует автомат (в качестве которого можно выбрать как автомат Мили, так и автомат Мура), каноническое множество событий которого совпадает с множеством M.*

Справедливость сформулированного предложения вытекает из доказанных в § 2 настоящей главы предложений 2.1 и 2.8 и сделанного выше замечания о совпадении канони-

ческих множеств событий, соответствующих автомату и индуцируемому им автоматному отображению.

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

В связи со всем сказанным становится понятной необходимость решения следующих важнейших задач абстрактной теории автоматов. Прежде всего нужно научиться решать задачу нахождения по заданному абстрактному автомату A соответствующего ему канонического множества событий; эту задачу мы будем называть *канонической задачей анализа абстрактного автомата*. Далее необходимо научиться решать обратную задачу: по заданному автоматному множеству событий M находить абстрактный автомат, каноническое множество событий которого совпадает с M; эту задачу мы будем называть *канонической задачей синтеза абстрактного автомата*.

Для решения задачи синтеза удобно считать, что события заданного множества M заранее отмечены различными буквами выходного алфавита синтезируемого автомата. Условимся в дальнейшем называть каноническим всякое автоматное множество событий, для которого выполнены соответствующие отметки. Буквы, отмечающие события канонического множества, составляют так называемый *алфавит отметок*. После решения задачи синтеза этот алфавит превращается в выходной алфавит построенного автомата.

При использовании введенной терминологии описанные выше задачи канонического анализа и синтеза автоматов формулируются как обратные друг другу задачи: в первой по заданному автомату строится соответствующее ему каноническое множество событий, во второй — по заданному каноническому множеству событий строится соответствующий ему автомат.

Поскольку, как уже отмечалось выше, каноническая задача синтеза по самому своему существу неоднозначна, возникает задача синтеза (по заданному каноническому множеству событий) автомата, имеющего наименьшее возможное число состояний. Эта

задача обычно формулируется и решается как *задача минимизации* автомата, найденного при решении канонической задачи синтеза.

На практике оказывается удобным несколько обобщить канонические задачи анализа и синтеза абстрактных автоматов. Это обобщение основано на рассмотрении не только таких событий, которые представляются в автоматах одним выходным сигналом, но и таких событий, которые представляются произвольными множествами выходных сигналов.

4.3. *Событием, представленным в автомате A каким-либо множеством M выходных сигналов, называется объединение событий, представленных всеми выходными сигналами, составляющими множество M.*

В соответствии с этим определением мы будем считать, что невозможное событие представлено в любом автомате пустым множеством выходных сигналов. Нетрудно понять, что всеобщее событие в любом алфавите X представляется множеством всех состояний любого вполне определенного автомата, имеющего алфавит X в качестве своего входного алфавита.

Общая задача анализа абстрактного автомата ставится теперь как задача нахождения по заданному абстрактному автомату такого события, которое представлено любым множеством его состояний. Аналогично, *общая задача синтеза абстрактного автомата* ставится как задача построения по любому (а не только по автоматному) конечному множеству M событий такого автомата, который представляет каждое событие этого множества некоторым множеством своих выходных сигналов (ограничение лишь конечными множествами исходных событий связано с тем, что число различных множеств выходных сигналов в любом абстрактном автомате конечно).

Для упрощения языка условимся в дальнейшем говорить, что автомат представляет некоторое множество событий, если он представляет каждое событие этого множества.

Ясно, что общая и каноническая задачи анализа автомата весьма близки между собой: найдя каноническое множество событий автомата, нетрудно найти событие,

представленное любым множеством выходных сигналов автомата как объединение некоторых событий канонического множества. Обратно, умея решать общую задачу анализа, можно одно за другим найти все события, составляющие каноническое множество.

Гораздо менее ясен вопрос о близости канонической и общей задач синтеза автомата. Более того, заранее неясно, имеет ли вообще решение произвольная общая задача синтеза. Нетрудно видеть, что умение решить общую задачу синтеза позволяет решить и каноническую задачу. В самом деле, пусть нам дано автоматное множество событий M. Если применить к этому множеству метод решений общей задачи синтеза (в предположении, что такой метод существует), то мы найдем автомат, в котором каждое событие S_i множества M будет представлено некоторым множеством N_i выходных сигналов. Ввиду автоматности множества событий M различные события этого множества попарно не пересекаются. Но тогда не могут, очевидно, пересекаться и представляющие их множества выходных сигналов. Заменяя все выходные сигналы каждого такого множества одним выходным сигналом (особым для каждого множества), мы решаем тем самым каноническую задачу синтеза.

Покажем теперь, что имеет место и обратное: умение решать каноническую задачу синтеза позволяет решать также и общую задачу. Для этой цели опишем прием, который мы будем называть *стандартным приемом сведения общей задачи синтеза автомата к канонической задаче синтеза*.

Пусть $M=(Q_1, \ldots, Q_n)$ — произвольное конечное множество событий в некотором алфавите $\mathfrak{X}$. Обозначим через P_i множество всех непустых слов события Q_i, не содержащихся ни в одном из остальных событий Q_j ($j\neq i$; $i, j=1, \ldots, n$). Через P_{ij} обозначим множество всех непустых слов, каждое из которых содержится в событиях Q_i и Q_j и не содержится ни в одном из остальных событий Q_k ($k\neq i$, $k\neq j$, $i\neq j$; $k, i, j=1, 2, \ldots, n$). Вообще, для любого $k=1, 2, \ldots, n$ через $P_{i_1 i_2 \ldots i_k}$ мы будем обозначать событие, состоящее из всех непустых слов, входящих в события $Q_{i_1}, Q_{i_2}, \ldots, Q_{i_k}$ и не входящих ни в одно из остальных событий множества M ($i_1, i_2, \ldots, i_k=1, \ldots, n$; $i_r\neq i_s$ при $r\neq s$). Через P_0

обозначим множество всех непустых начальных отрезков слов событий $Q_1, \ldots, Q_n$, которые (рассматриваемые как отдельные слова) не содержатся ни в одном из событий $Q_1, \ldots, Q_n$.

Обозначим через $S_1, S_2, \ldots, S_m$ все непустые события из числа построенных событий $P_1, P_2, \ldots, P_n, P_{12}, \ldots, P_{12\ldots n}, P_0$ и назовем полученное множество N событий $(S_1, \ldots, S_m)$ *каноническим разбиением* исходного множества M.

Из самого способа построения канонического разбиения вытекает справедливость следующего предложения.

4.4. *Каноническое разбиение произвольного конечного множества событий в любом конечном алфавите представляет собою автоматное множество событий.*

Используя свойство автоматности построенного канонического разбиения N, можно, решая каноническую задачу синтеза, построить автомат A, для которого множество M будет каноническим множеством событий. Из способа построения канонического разбиения непосредственно следует, что любое из исходных событий Q_i представляется (с точностью до пустого слова) в виде объединения всех непустых событий вида

$$P_{i, i_1, \ldots, i_k} \quad (i_1 \ldots i_k = 1, \ldots, n;\ k = 0, 1, \ldots, n-1).$$

Так как каждое из таких событий представлено в автомате A каким-либо выходным сигналом, то событие Q_i (за вычетом, быть может, лишь пустого слова) для любого $i = 1, \ldots, n$ представляется в этом автомате некоторым множеством выходных сигналов.

Следовательно, описанный прием действительно сводит решение общей задачи синтеза автомата к решению канонической задачи синтеза. При этом исходные для общей задачи события представляются в синтезированном автомате с точностью до пустого слова. Особую роль пустого слова в задачах синтеза нетрудно понять, если вспомнить, что в принятом нами законе функционирования автоматов пустому слову не соответствует никакого выходного сигнала.

Поэтому задача представления событий выходными сигналами для случая событий, содержащих пустое слово, строго говоря, не имеет смысла.

Мы условимся в дальнейшем рассматривать все задачи представления событий с точностью до пустого слова. Иначе, будем говорить, что некоторое событие S представлено в автомате множеством M выходных сигналов, если этим множеством представлено событие, составленное из всех непустых слов события S.

Описанный выше стандартный прием действительно позволяет (с учетом сделанных замечаний относительно пустого слова) свести общую задачу синтеза автоматов к канонической задаче. Что же касается последней задачи, то она всегда имеет решение в силу предложения 4.2. Однако такое решение (основанное на результатах § 2) не может нас удовлетворить, поскольку оно приводит к построению, вообще говоря, бесконечного автомата даже в тех случаях, когда существует дающий решение задачи конечный автомат.

Поэтому сформулированные выше задачи синтеза (а также и задачи анализа) автоматов нуждаются в дальнейшем уточнении. При таком уточнении нужно, с одной стороны, фиксировать некоторый язык, приспособленный для конструктивного описания событий (как множеств слов), с которыми мы будем оперировать, а с другой стороны,— ограничиться некоторым вполне определенным классом автоматов, допускающих какой-либо конструктивный способ задания. Ясно, что при этом уже нельзя будет оперировать с произвольными событиями и автоматами, поскольку, как хорошо известно в математике, «произвольные» множества и «произвольные» функции не допускают конструктивного описания. Поэтому мы ограничимся в дальнейшем лишь конечными автоматами и регулярными событиями. Оба эти класса объектов допускают конструктивное описание; первый — на языке таблиц переходов и выходов, а второй — на языке регулярных выражений алгебры событий. Такое ограничение вполне достаточно для практических целей, потому что цифровые автоматы, с которыми приходится иметь дело на практике, всегда конечны.

Все остальное содержание настоящей главы будет посвящено разработке методов решения общих задач анализа, синтеза и минимизации для случая конечных автоматов. Уточненные постановки

этих задач приводятся в специально посвященных этим задачам параграфах.

Заметим теперь же, что справедливо следующее предложение.

4.5. *Для любого конечного множества M событий в любом конечном алфавите тогда и только тогда существует конечный автомат, представляющий каждое из событий множества M некоторым множеством своих выходных сигналов, когда все события множества M регулярны*[1]).

Истоки этой фундаментальной для теории конечных автоматов теоремы восходят еще к работе С. К. Клини[2]), где был впервые доказан один ее важный частный случай. Одним из следствий этой теоремы является то, что язык регулярных выражений оказывается достаточным для описания отображений, индуцируемых произвольными конечными автоматами.

Отметим еще, что на основании предложения 2.8 и построенного выше стандартного приема сведения общей задачи синтеза автомата к канонической задаче синтеза из доказанной в § 2 теоремы 2.2 может быть выведено следующее следствие.

4.6. *Если некоторое (конечное) множество событий в m-буквенном алфавите допускает представление (множествами выходных сигналов) в конечном автомате Мили с n состояниями, то оно допускает представление (также множествами выходных сигналов) в конечном автомате Мура, число состояний которого не превосходит $mn+1$.*

Для случая автоматов Мура иногда удобно бывает считать, что представленное в автомате A некоторым множеством N выходных сигналов событие S представляется множеством M всех тех состояний автомата, которые отмечены выходными сигналами множества N. Ясно, что все слова события S переводят автомат A из начального состояния в состояния из множества M.

Условимся говорить, что событие S представлено в автомате Мура множеством M (конечных) состояний, если

---

[1]) Строгое доказательство этого предложения будет дано в двух последующих параграфах.

[2]) С. К. Клини, Представление событий в нервных сетях и конечных автоматах. Сб. «Автоматы», ИЛ, М., 1956, стр. 15—67.

событие S состоит из всех тех и только тех входных слов, которые переводят автомат из начального состояния в одно из состояний множества M.

Отмечая все состояния множества M выходными сигналами, отличными от выходных сигналов, отмечающих не вошедшие в M состояния, мы можем событие S, представленное ранее множеством состояний M, представить множеством N всех выходных сигналов, которыми отмечены состояния множества M.

Легко видеть, что при этом множеством N выходных сигналов представляются все непустые слова события S и только такие слова. Что же касается пустого слова, то мы уже условились выше не принимать его во внимание при представлении событий. Поэтому способы представления событий в автоматах Мура множествами выходных сигналов и множествами состояний оказываются эквивалентными (с точностью до пустого слова) между собой.

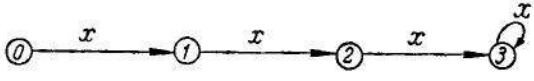

Рис. 4.

Изучая проблему представления событий множествами состояний автомата, естественно поставить вопрос о возможности сведения этой проблемы к проблеме о представлении событий одним состоянием. Иначе говоря, возникает вопрос, можно ли для всякого события, представленного в автомате Мура множеством состояний, построить новый автомат Мура, в котором это событие было бы представлено одним единственным состоянием?

Легко видеть, что ответ на поставленный вопрос — отрицателен. В самом деле, достаточно рассмотреть событие $S = x \vee xx$, состоящее из двух слов x и xx в однобуквенном алфавите $\mathfrak{X}(x)$. Это событие представимо в автомате, граф которого изображен на рис. 4, множеством двух состояний 1 и 2.

Предположим, что существует автомат A, в котором рассматриваемое событие S представлено каким-либо одним состоянием a. Ясно, что под действием входа x автомат

из начального состояния a_0 должен перейти в состояние a, ибо слово x содержится в S : $a_0x=a$. Поскольку слово xx также содержится в S, то оно должно переводить автомат из состояния a_0 в состояние a, то есть $a_0(xx)=a$. Но это означает, очевидно, что входной сигнал x переводит состояние $a=a_0x$ в состояние (a_0x) $x=a_0(xx)=a$. В таком случае не только слова x и xx, но также слова xxx, $xxxx$ и т. д. должны переводить автомат из состояния a_0 в состояние a и, следовательно, входить в событие S, что на самом деле не так. Полученное противоречие и доказывает невозможность представления события S одним состоянием ни в каком автомате.

Еще более важное значение имеет вопрос о существовании непредставимых в автоматах событий. Если не исключать из рассмотрения автоматов с бесконечным числом внутренних состояний, то все события оказываются представимыми. Однако в случае конечных автоматов дело обстоит иначе. Первый пример события, не представимого ни в каком конечном автомате, был указан К л и н и. Таким событием является рассмотренное в предыдущем параграфе нерегулярное событие S, состоящее из всех слов, длины которых являются точными квадратами. Непредставимость этого события вытекает из его нерегулярности, поскольку, как будет показано в следующем параграфе, все представимые в конечных автоматах события регулярны.

В заключение настоящего параграфа рассмотрим вопрос о связи проблемы представления событий в конечных автоматах со степенями квадратной автоматной матрицы.

Рассмотрим произвольный автомат A с входным алфавитом $(x_1, \ldots, x_m)$ и множеством внутренних состояний $(a_1, a_2, \ldots)$. Состояние a_1 будем считать начальным. Как явствует из определения квадратной автоматной матрицы $\mathfrak{A}=\|\alpha_{ij}\|$ автомата A (см. § 2 настоящей главы) для любых i и j, элемент α_{ij} представляет собою не что иное, как множество слов входного алфавита, имеющих длину 1 и переводящих автомат из состояния a_i в состояние a_j.

В соответствии с введенными определениями мы можем, следовательно, рассматривать элементы матрицы $\mathfrak{A}=\|\alpha_{ij}\|$ как события в алфавите $(x_1, \ldots, x_m)$.

Используя операции алгебры событий, мы можем построить алгебру матриц, элементами которых служат события.

Дизъюнкцией $\mathfrak{A} \vee \mathfrak{B}$ двух квадратных матриц $\mathfrak{A} = \| \alpha_{ij} \|$ и $\mathfrak{B} = \| \beta_{ij} \|$ одного и того же порядка n называется матрица $\| \alpha_{ij} \vee \beta_{ij} \|$.

Произведением матриц $\mathfrak{A}$ и $\mathfrak{B}$ мы будем называть матрицу $\| \gamma_{ij} \|$, элементы которой определяются по формулам:

$$\gamma_{ij} = \alpha_{i1}\beta_{1j} \vee \alpha_{i2}\beta_{2j} \vee \ldots \vee \alpha_{in}\beta_{nj} \qquad (i, j = 1, 2, \ldots, n). \quad (1)$$

Произведение и дизъюнкция понимается в этих формулах в смысле алгебры событий.

Теперь мы можем определить последовательные степени квадратной автоматной матрицы $\mathfrak{A}$.

$$\mathfrak{A}^1 = \mathfrak{A} \quad \mathfrak{A}^2 = \mathfrak{A} \cdot \mathfrak{A}, \quad \mathfrak{A}^3 = \mathfrak{A}^2 \cdot \mathfrak{A} \text{ и т. д.}$$

Условимся через S_{ij} обозначать событие, состоящее из всех слов входного алфавита, переводящих рассматриваемый автомат A из состояния a_i в состояние a_j. Через $S_{ij}(k)$ мы будем обозначать событие, состоящее из всех тех слов события S_{ij}, которые имеют длину k. Введем еще обозначение $\alpha_{ij}(k)$ для элемента матрицы $\mathfrak{A}^k$, стоящего на пересечении i-й строки и j-го столбца.

Справедливо следующее предложение.

Для любого $k = 1, 2, \ldots$ *имеет место соотношение*

$$S_{ij}(k) = \alpha_{ij}(k) \qquad (2)$$

Переходя к доказательству этого предложения, заметим, что справедливость соотношения (2) при $k = 1$ вытекает непосредственно из определения квадратной автоматной матрицы, поскольку, очевидно, $\alpha_{ij}(1) = \alpha_{ij}$. Предположим теперь соотношение (2) доказанным для всех $k \leqslant p$ и будем доказывать его для $k = p + 1$.

По определению степеней матрицы $\mathfrak{A}$ мы имеем $\mathfrak{A}^{p+1} = \mathfrak{A}^p \mathfrak{A}$ или, переходя к элементам:

$$\alpha_{ij}(p+1) = \alpha_{i1}(p)\alpha_{1j}(1) \vee \ldots \vee \alpha_{in}(p)\alpha_{nj}(1) \qquad (3)$$
$$(i, j = 1, \ldots, n).$$

Любое слово h из $\alpha_{ij}(p+1)$ имеет вид $h = gx$, где $g \in \alpha_{ik}(p)$, $x \in \alpha_{kj}(1)$ при каком-либо k $(1 \leqslant k \leqslant n)$. Слово g, согласно индуктивному предположению, переводит авто-

мат A из состояния a_i в состояние a_k и имеет длину p. Что же касается слова x, то оно имеет длину 1 и переводит автомат из состояния a_k в состояние a_j. Но тогда, очевидно, слово h имеет длину $p+1$ и переводит автомат из состояния a_i в состояние a_j.

Обратно, пусть h — произвольное слово длины $p+1$, переводящее автомат из состояния a_i в состояние a_j. Слово h можно записать в виде произведения gx слова g длины p и слова x длины 1. Пусть слово g переводит автомат из состояния a_i в некоторое состояние a_k; тогда, очевидно, слово x должно переводить автомат из состояния a_k в состояние a_j. Следовательно, согласно индуктивному предположению, слово g содержится в $\alpha_{ik}(p)$, а слово x—в α_{kj} (1). В силу формулы (3) отсюда следует, что слово h содержится в α_{ij} $(p+1)$. Тем самым индукция проведена, и формула (2) доказана для всех значений $k=1, 2,\ldots$

Из доказанного предложения вытекает следствие:

Если образовать дизъюнкцию всех степеней квадратной автоматной матрицы $\mathfrak{A}$ *произвольного автомата* A, то *в полученной матрице* $\mathfrak{B}=\mathfrak{A}^0 \vee \mathfrak{A}^1 \vee \mathfrak{A}^2 \vee \ldots$. *элемент* p_{ij}, *стоящий на пересечении* i*-й строки и* j*-го столбца, представляет собою множество всех слов, переводящих автомат из состояния* a_i *в состояние* a_j. *Под нулевой степенью* $\mathfrak{A}^0$ *матрицы* $\mathfrak{A}$ *здесь понимается матрица, на главной диагонали которой стоят символы пустого слова* e, *а на всех остальных местах — невозможное событие* Φ.

Дизъюнкция любого множества элементов первой строки матрицы $\mathfrak{B}$: $\beta_{1j_1} \vee \beta_{1j_2} \vee \ldots \vee \beta_{1j_r}$ является не чем иным, как событием, представляемым в автомате A начальным состоянием a_1 и множеством конечных состояний $a_{j_1}, a_{j_2}, \ldots, a_{j_r}$.

Тем самым установлена требуемая связь между задачей представления событий в абстрактном автомате и степенями квадратной матрицы.

§ 5. Анализ конечных автоматов

В настоящем параграфе мы будем решать общую проблему анализа для случая конечных автоматов.

Точная формулировка этой проблемы такова.

Требуется построить алгоритм, который позволял бы для любого конечного автомата Мили или Мура A, *задан-*

ного своими таблицами переходов и выходов или, соответственно, отмеченной таблицей переходов, и для любого множества N выходных сигналов автомата A находить регулярное выражение (одно из возможных) для события, представленного в автомате A множеством N выходных сигналов.

Как следует из результатов предыдущего параграфа, при решении проблемы анализа для автоматов Мура можно предполагать, что событие S, регулярное выражение для которого требуется найти, представлено в автомате не множеством N выходных сигналов, а множеством состояний автомата.

С целью подчеркнуть, что речь идет не о выходных сигналах автомата, а о его состояниях, мы будем называть состояния автомата его *внутренними состояниями*. Соответствующим образом переформулированную общую проблему анализа мы будем называть *проблемой анализа конечного автомата Мура*. Эта проблема решается теми же методами, что и общая проблема анализа автоматов Мили или Мура, однако имеет по сравнению с последней то преимущество, что при ее решении можно ограничиться использованием лишь таблицы переходов или графа автомата, не прибегая к таблице выходов.

Примем поэтому такой план изложений: сначала изложим метод решения проблемы анализа конечных автоматов Мура при условии представления событий множеством внутренних состояний, а затем укажем на те изменения, которые необходимо сделать в описанном алгоритме для решения общей проблемы анализа конечных автоматов Мили.

Итак, переходим к решению проблемы анализа конечных автоматов Мура, для чего сформулируем предварительно эту проблему в точной постановке.

Проблема анализа конечного автомата Мура заключается в следующем. Дана таблица переходов или граф конечного автомата Мура A, дано начальное состояние a_1 автомата A и произвольное множество M его внутренних состояний. Требуется найти регулярное выражение для события, представленного в этом автомате множеством M.

Для построения *алгоритма анализа*, решающего поставленную проблему, введем предварительно несколько новых понятий.

Путем в автомате A с входным алфавитом $(x_1, \ldots, x_n)$ и внутренними состояниями $a_1, \ldots, a_m$ называется всякая конечная последовательность попарно чередующихся букв a_j и x_l, а именно: $l = a_{j_0} x_{i_1} a_{j_1} x_{i_2} a_{j_2} \ldots a_{j_{k-1}} x_{i_k} a_{j_k}$ $(k \geqslant 0)$ при условии, что для любого $p = 1, \ldots, k$ состояние $a_{j_{p-1}}$ переводится входом x_{i_p} в состояние a_{j_p}. Путь называется *простым*, если все входящие в него состояния a_j, за исключением, быть может, лишь первого a_{j_0} и последнего a_{j_k}, попарно различны. Если $a_{j_0} = a_{j_k}$, то путь называется *замкнутым*.

Слово $h = x_{i_1} x_{i_2} \ldots x_{i_k}$, получающееся в результате удаления из пути l символов внутренних состояний, называется (входным) словом, соответствующим этому пути. Легко видеть, что путь l однозначно определяется соответствующим ему входным словом h и начальным состоянием a_{j_0}. Используя это обстоятельство, мы будем иногда обозначать путь соответствующей ему парой (a_{j_0}, h). Условимся также говорить, что путь l входит в то или иное событие S, если соответствующее этому пути входное слово входит в S.

Введем еще понятие о *типе путей*. Всякий путь в автомате A, имеющем внутренние состояния $a_1, \ldots, a_m$, начинающийся состоянием a_i и кончающийся любым состоянием из любого множества внутренних состояний автомата M, будем называть путем типа $a_i \to M$; пути типа $a_i \to a_i$ мы будем называть просто путями типа a_i.

Наконец, если N — любое множество внутренних состояний, не содержащее состояния a_i, то условимся называть путем типа $a_i | N |$ всякий путь типа a_i, не содержащий в своем составе ни одного состояния из множества N. Заметим, что пути типа a_i можно при желании называть путями типа $a_i | N |$, выбирая в качестве множества N пустое множество внутренних состояний. Это обстоятельство позволяет в дальнейшем изложении рассматривать лишь два сорта типов путей: тип $a_i \to M$, который мы будем называть *начальным*, и тип $a_i | N |$, который естественно назвать *циклическим*. Разумеется, если множество M состоит из единственного элемента a_l, то начальный тип

$a_i \to a_i$ будет вместе с тем и циклическим. Ясно также, что всякий путь типа $a_i|N|$ будет вместе с тем и путем типа $a_i|N_1|$, где N_1 — любое подмножество множества N. Для упрощения формул условимся в случае необходимости заменять в обозначениях типов путей символы состояний индексами этих состояний (для чего, разумеется, необходимо фиксировать некоторую нумерацию состояний), так что, например, пути типа a_i будут называться также путями типа i, и т. п.

Множество всех слов входного алфавита, соответствующих путям любого данного типа P, мы будем называть *событием типа* P.

Введем, наконец, понятие *комплексов* различных типов. Пусть S — некоторое событие, представленное в автомате A с начальным состоянием a_1 множеством M внутренних состояний. Тогда *начальным комплексом события* S или просто *начальным комплексом* называется формальная дизъюнкция всех простых незамкнутых путей типа $a_1 \to M$. Этот комплекс мы будем обозначать через K_S или просто через K, если рассматривается лишь одно событие S. Если $a_1 \in M$, то в начальный комплекс мы будем включать путь a_1, состоящий лишь из одной буквы, который условимся считать незамкнутым. Комплексом типа $a_i|N|$ условимся называть формальное выражение, получаемое в результате вычеркивания в каждом из простых путей типа $a_i|N|$ символов a_i, стоящих в начале и в конце пути, объединения всех полученных в результате такого вычеркивания «урезанных» путей знаками дизъюнкции и заключения полученного выражения в итерационные (фигурные) скобки. В комплексы типов a_i и $a_i|N|$ не включаются пути, состоящие лишь из одной буквы. Если пути какого-либо типа P в автомате отсутствуют, то соответствующий комплекс типа P мы будем называть *пустым* и отождествлять с пустым словом e.

Пустота начального комплекса события S означает, очевидно, что событие S пусто. Поскольку в этом случае решение проблемы анализа тривиально (формально регулярным выражением для S может служить, например, дизъюнкция пустого множества букв входного алфавита), то в дальнейшем этот случай мы исключим из рассмотрения.

Подчеркнем, что комплексы составляются не из произвольных путей данного типа, а лишь из простых путей этого типа. Поскольку число всех простых путей в конечном автомате очевидным образом конечно, то конечными являются и все определенные выше комплексы. Каждый из таких комплексов может рассматриваться поэтому как некоторое регулярное выражение в расширенном алфавите $(x_1, \ldots, x_n, a_1, \ldots, a_m)$. Для любого данного конечного автомата конечным будет, разумеется, и число комплексов.

Нетрудно видеть, что при фактическом построении комплексов все трудности оказываются преодоленными после того, как найдены выражения для начального комплекса и всех комплексов типа a_i $(i=1, \ldots, m)$. Действительно, в силу определения всякий комплекс типа $a_i[N]$ получается из комплекса типа a_i вычеркиванием всех тех дизъюнктивных членов, в составе которых встречаются символы состояний, входящих в множество N. Что же касается начального комплекса и комплексов типа a_i, то их приходится находить с помощью фактического перебора соответствующих простых путей на графе автомата или в его таблице переходов.

Поясним теперь все сказанное о построении комплексов на примере. Предположим, что нам необходимо найти комплексы всех типов для события S, представленного в автомате A^* с начальным состоянием 1 множеством M, состоящим из состояний 1 и 2; автомат A^* задан таблицей переходов II.4.

Таблица II. 4

	1	2	3
x	2	2	2
y	1	3	1

Непосредственным перебором по табл. II.4 мы находим все простые незамкнутые пути типа $1 \to (1,2)$. Таких путей будет, очевидно, два: путь 1, содержащий единственное состояние 1, и путь $_1x_2$ (для удобства чтения, символы

состояний в пути условимся записывать в виде нижних индексов при соответствующих буквах). Следовательно, начальный комплекс K события S будет иметь вид $1 \vee_1 x_2$.

Аналогичным путем находим, что имеются два простых пути типа 1: ${}_1y_1$, ${}_1x_2y_2y_1$, три простых пути типа 2: ${}_2x_2$, ${}_2y_2x_2$, ${}_2y_2y_1x_2$ и два простых пути типа 3: ${}_3x_2y_2$, ${}_3y_1x_2y_2$. Комплексы типов 1, 2 и 3 имеют вид:

$$K_1 = \{y \vee x_2y_2y\},$$
$$K_2 = \{x \vee y_2x \vee y_2y_1x\},$$
$$K_3 = \{x_2y \vee y_1x_2y\}.$$

Все остальные комплексы находятся с помощью вычеркивания в комплексах K_1, K_2, K_3 лишних членов:

$$K_{1[2]} = K_{1[3]} = K_{1[2,3]} = \{y\},$$
$$K_{2[1]} = \{x \vee y_2x\},$$
$$K_{2[3]} = K_{2[1,3]} = \{x\}$$
$$K_{3[1]} = \{x_2y\},$$
$$K_{3[2]} = K_{3[1,3]} = e.$$

Для дальнейшего будет иметь значение еще одно понятие, а именно понятие о *ранге комплексов*. Условимся приписывать всем комплексам типа $a_i|N|$, для которых множество N состоит из k элементов, ранг $k+1$. Комплексы типа a_i при этом будут иметь ранг 1. Что же касается начального комплекса, то ему, по определению, присваивается нулевой ранг.

Алгоритм анализа конечных автоматов состоит в последовательном применении одной единственной операции, которую мы назовем *операцией расширения*. Суть этой операции состоит в замене всех входящих в то или иное выражение символов внутренних состояний автомата символами комплексов; операция выполняется в соответствии с правилом, называемым *правилом замены*.

Правило замены. Если путь $a_1x_{i_1}\ a_{j_1}\dots\ x_{i_k}a_{j_k}$ входит в начальный комплекс, то буква a_1 заменяется символом комплекса типа a_1, буква a_{j_1} — символом комплекса типа $a_{j_1}|a_1|$, буква a_{j_2} — символом комплекса типа $a_{j_2}|a_1,\ a_{j_1}|$ и т. д. Если же урезанный путь (т. е. путь без начального и конечного состояний) $x_{i_1}a_{j_1}x_{i_2}a_{j_2}\dots x_{i_r}$

входит в комплекс типа $a_j[N]$, то буква a_{j_1} заменяется символом комплекса типа $a_j\,[N,\ a_j]$, буква a_{j_2} — символом комплекса типа $a_j\,]N,\ a_j,\ a_{j_1}[$ и т. д.

Операция расширения может применяться к любому регулярному выражению (и даже к произвольной формуле) над алфавитом, содержащим все буквы входного алфавита автомата и символы всех его комплексов. Если $f\,(x_1,\ \ldots,\ x_n,\ K,\ K_{P_1},\ \ldots,\ K_{P_s})$ — произвольная формула указанного вида, то операция расширения состоит, во-первых, в замене всех символов комплексов их явными выражениями через пути, а, во-вторых, в подстановке вместо символов внутренних состояний a_i в каждом из возникших таким образом путей символов комплексов высших рангов в соответствии со сформулированным выше правилом замены.

При применении операции расширения из регулярного выражения получается, как нетрудно видеть, снова регулярное выражение; при этом, однако, минимальный ранг входящих в него комплексов повышается по сравнению с исходным выражением не менее, чем на единицу. Если в исходном автомате A имеется m внутренних состояний, то после $(m+1)$-кратного применения операции расширения к любому регулярному выражению $f\,(x_1,\ldots\ \ldots,\ x_n,\ K,\ K_{P_1}\ \ldots,\ K_{P_s})$ в алфавите, состоящем из входных сигналов и символов комплексов этого автомата, получится, очевидно, регулярное выражение $R(x_1,\ \ldots,\ x_n)$, уже не содержащее символов комплексов автомата.

Поясним операцию расширения комплексов на примере автомата A^*, уже рассмотренного в настоящем параграфе с целью пояснения методов построения комплексов (см. стр. 83). Пусть исходное выражение f совпадает с символом K начального комплекса представленного в автомате A^* события S. Применим к нему четыре раза операцию расширения:

1-е расширение:

$$K = 1 \vee_1 x_2 \longrightarrow K_1 \vee K_1 x K_{a_{[1]}} = K_1\,(e \vee x K_{a_{[1]}}).$$

2-е расширение:

$$K_1\,(e \vee x K_{a_{[1]}}) = \{y \vee x_2 y_2 y\}\,(e \vee x\,\{x \vee y_2 x\}) \longrightarrow$$
$$\longrightarrow \{y \vee x K_{a_{[1]}} y K_{a_{[1,\ a_1]}} y\}\,(e \vee x\,\{x \vee y K_{a_{[1,\ a_1]}} x\}).$$

3-е расширение:

$$\{y\vee x K_{s[1,2]} y K_{s[1,2]} y\}\,(e\vee x\{x\vee y K_{s[1,2]} x\}) =$$
$$= \{y\vee x\{x\vee y_3 x\}\, yey\}\,(e\vee x\{x\vee yex\}) =$$
$$= \{y\vee x\{x\vee y_3 x\}\, yy\}\,(e\vee x\{x\vee yx\}) \rightarrow$$
$$\rightarrow \{y\vee x\{x\vee y K_{s[1,2]} x\}\, yy\}\,(e\vee x\{x\vee yx\}).$$

4-е расширение:

$$\{y\vee x\{x\vee y K_{s[1,2]} x\}\, yy\}\,(e\vee x\{x\vee yx\}) =$$
$$= \{y\vee x\{x\vee yex\}\, yy\}\,(e\vee x\{x\vee yx\}) =$$
$$= \{y\vee x\{x\vee yx\}\, yy\}\,(e\vee x\{x\vee yx\}).$$

На 4-м расширении применять правило замены уже не требуется, ввиду отсутствия символов внутренних состояний. Ясно, что все последующие расширения уже не будут менять полученного выражения. Заметим еще, что при первом расширении мы применяли операции алгебры событий с целью упрощения полученного выражения; введенный здесь символ e означает пустое слово и в исходный алфавит не входит. Разумеется, применять упрощения выражений в алгебре событий при выполнении операции расширения вовсе необязательно,— это делается лишь с целью упрощения записи.

Основным результатом, обосновывающим алгоритм анализа конечных автоматов, является следующее предложение.

(*U*) *Если событие S представлено множеством M внутренних состояний конечного автомата A, имеющего m внутренних состояний, то событие S регулярно, а регулярное выражение для этого события может быть получено в результате $(m+1)$-кратного применения операции расширения к символу начального комплекса события S.*

Мы докажем более общее предложение (*V*), из которого справедливость предложения (*U*) усматривается непосредственно.

(*V*) *$(m-k+1)$-кратное расширение любого комплекса типа P, имеющего ранг k, приводит к регулярному выражению, представляющему событие Q_P типа P.*

Действительно, начальный комплекс K, о котором идет речь в предложении (*U*), имеет тип $a_1 \rightarrow M$ ранга

нуль, а событие типа $a_1 \to M$ есть не что иное, как событие S. Таким образом, предложение (U) действительно является простым следствием предложения (V) и, следовательно, нам достаточно доказать это последнее предложение.

Прежде чем перейти к доказательству предложения (V), введем одно новое понятие. Если $f(x_1, \ldots, x_n, K, K_{P_1}, \ldots, K_{P_t})$ — любое регулярное выражение в алфавите, содержащем входные сигналы $x_1, \ldots, x_n$ автомата и символы всех его комплексов, то *событием, соответствующим* этому выражению, мы будем называть событие во входном алфавите $(x_1, \ldots, x_n)$, представляемое регулярным выражением, которое получается из выражения f в результате замены символов комплексов их значениями и вычеркивания (замены пустыми словами) в полученной формуле всех символов внутренних состояний автомата.

Так, например, для рассмотренного выше автомата A^* выражению $K_1 \vee K_{s[1]}$ будет соответствовать событие $\{y \vee xyy\} \vee x\ \{xy\}$, выражению $\{K \vee K_s y\}$ — событие $\{e \vee x \vee \{xy\} \vee yxy\}y\} = \{x \vee \{xy \vee yxy\}y\}$, выражению K — событие $e \vee x$ и т. п.

Условимся через $K_P(j)$ обозначать j раз расширенный комплекс K_P типа P, а через $R_P(j)$ — событие, соответствующее выражению $K_P(j)$ $(j = 0, 1, \ldots, m+1)$. Выражение $K_P(0)$ совпадает с исходным комплексом K_P.

В терминах введенных понятий предложение (V) можно перефразировать следующим образом:

Если комплекс K_P имеет ранг k, то событие $R_P(m-k+1)$ совпадает с событием Q_P типа P.

Приступая к доказательству предложения (V), заметим, что выполнение каждой операции расширения, в соответствии с формулировкой предложения (V), обогащает исходное выражение: в событии $R_P(j+1)$ появляются, вообще говоря, новые слова по сравнению с событием $R_P(j)$. Однако из способа расширения непосредственно вытекает, что путь l, соответствующий каждому такому новому слову h, получается из пути l_0, соответствующего некоторому слову события $R_P(j)$, с помощью вставок циклических путей, начинающихся и кончающихся одним и тем же состоянием пути l_0. Начальное и конечное состояния пути при этом не меняются. Поэтому слово h всегда

остается принадлежащим событию типа P. Тем самым доказано, что при любом j, в том числе и при $j=k$, имеет место включение $R_p\,(m-j+1)\subset Q_p$.

Теперь, для того чтобы доказать предложение (V), достаточно установить, что

$$Q_p\subset R_p\,(m-k+1). \qquad (1)$$

Справедливость этого включения будет доказана индукцией по рангу k. Для $k=m$ справедливость (1) очевидна, поскольку для всякого комплекса $K_a[N_m]$ ранга m соответствующее ему событие $R_{a_i}[N_m]$ имеет вид $\{x_{i_1}\vee x_{i_2}\vee\ldots\vee x_{i_s}\}$ и, в силу определения комплекса типа $a_i[N_m]$, включает все слова, переводящие автомат из состояния a_i в состояние a_l, минуя все остальные состояния.

Таким образом, справедливость включения (1) для всех типов P ранга m может считаться установленной.

Предположим теперь, что включение (1) доказано для всех значений ранга k, больших чем p; докажем справедливость (1) для $k=p$. Пусть фиксированы произвольный тип P ранга p и произвольное слово q из события Q_p типа P. Требуется доказать, что слово q принадлежит событию $R_p(m-p+1)$, которое мы для краткости будем обозначать просто через R.

Рассмотрим отдельно два случая.

1-й случай. Ранг p больше нуля; тип имеет вид $a_i[N]$. Путь $l=(a_i,q)$, определяемый состоянием a_i и словом q, будет в этом случае начинаться и кончаться состоянием a_i. Это обстоятельство позволяет представить путь l в виде $l=a_il_1a_il_2\ldots a_il_ca_i$, где урезанные (с вычеркнутыми начальным и конечным состояниями) пути $l_1,l_2,\ldots,\ l_c$ не содержат состояния a_i. Поскольку выражение $R=R_p(m-p+1)$ в рассматриваемом случае заключено в итерационные скобки, то очевидно, что для доказательства требуемого включения слова q в событие R достаточно показать, что все слова, соответствующие путям $a_il_1a_i,\ldots,\ a_il_ca_i$, содержатся в событии R. Следовательно, не нарушая общности, можно предполагать, что слово l содержит символ a_i лишь в начале и в конце.

Представим теперь слово l в виде

$$l=a_ix_{n_0}(a_{m_1}h_1a_{m_1})\,x_{n_1}(a_{m_2}h_2a_{m_2})\,x_{n_2}\ldots(a_{m_r}h_ra_{m_r})\,x_{n_r}a_i. \qquad (2)$$

Здесь путь $a_{m_1}h_1a_{m_1}=q_1$ не содержит символа a_l, путь $g_2=a_{m_2}h_2a_{m_2}$ не содержит символов a_l и a_{m_1} и т. д. Одним словом, каждый из выделенных в скобки путей не содержит ни одного из символов внутренних состояний, встречающихся в явном виде в выражении (2) слева от этого пути. Добиться такого представления слова l можно следующим очевидным способом: нужно выделить самый левый и самый правый из символов a_{m_1}; тогда в оставшейся (правой) части пути l символ a_{m_1} уже встречаться не будет. В этой оставшейся части слова находим самый левый символ a_{m_2} и выделяем новый участок пути: от самого левого символа a_{m_2} до самого правого. Продолжая подобным образом, мы получим требуемое разбиение пути l.

Поскольку в пути l по условию не могут встречаться символы состояний из множества N, то мы приходим к заключению, что путь $g_1=a_{m_1}h_1a_{m_1}$ имеет тип $a_{m_1}[N, a_l]$ и содержится, следовательно, в событии $Q_{a_{m_1}[N,a_l]}$, путь g_2 содержится в событии $Q_{a_{m_2}[N,a_{l_1},a_{m_1}]}$ и т. д.

Легко заметить также, что путь l получается из простого пути $l_0=a_lx_{n_0}a_{m_1}x_{n_1}a_{m_2}x_{n_2}\ldots a_{m_r}x_{n_r}a_l$ в результате подстановки вместо состояний $a_{m_1}, a_{m_2},\ldots, a_{m_r}$ соответственно путей $g_1, g_2,\ldots, g_r$. Но путь l_0 после выбрасывания начального и конечного состояний войдет, очевидно, в комплекс K_p типа $P=a_l[N]$. Подстановка путей g_i в путь l_0 соответствует подстановке при образовании расширения $K_p\to K_p$ (1): каждый из путей g_i подставляется в то же самое место пути l_0, куда при расширении подставляется комплекс того же типа, что и путь g_i. Но все подставляемые комплексы имеют ранги, большие, чем p, и потому, согласно предположению индукции, событие, соответствующее любому из этих комплексов, содержит слово, соответствующее тому из путей $g_1,\ldots,g_r$, который имеет одинаковый с данным комплексом тип.

При переходе от расширения K_p к расширению K_p $(m-p+1)$ будет совершено $m-p$ шагов, что, по предложению индукции, оказывается достаточным, чтобы все подставленные при первом расширении комплексы (они имеют ранг $p+1$ и выше) превратились в соответствующие им события.

Теперь ясно, что слово q, соответствующее пути l, может быть получено в результате раскрытия регулярного

выражения $K_p(m-p+1)$. Следовательно, $q \in R_p\,(m-p+1)$ и, ввиду произвольности q из Q_p, мы получаем требуемое включение:

$$Q_p \subset R_p\,(m-p+1).$$

2-й случай. Индекс P равен нулю; тип P имеет вид $a_1 \to M$. Путь $l=(a_1, q)$, определяемый состоянием a_1 и словом q, будет начинаться состоянием a_1 и кончаться одним из состояний a_{m_i} множества M. В этом случае доказательство ведется аналогично первому случаю. Отличие заключается лишь в том, что первый этап рассуждений опускается, и сразу строится разбиение пути l, аналогичное разбиению (2), но все же немного от него отличающееся:

$$l = (a_1 h_0 a_1)\, x_{n_1}\, (a_{m_1} h_1 a_{m_1}) \ldots\ x_{n_r}\, (a_{m_r} h_r a_{m_r}). \qquad (3)$$

Здесь, как и в первом случае, в первой скобке g_0 выделено самое левое и самое правое вхождение символа a_1 в путь l, во второй — самое левое и самое правое вхождение символа a_m в оставшуюся (правую) часть пути. В дальнейшем рассуждения проводятся точно так же, как и в первом случае, и поэтому мы их проводить не будем.

Предложение (V), а значит, и предложение (U) доказано. Тем самым полностью обоснован алгоритм анализа для случая конечных автоматов Мура.

Алгоритм анализа для случая конечных автоматов Мили отличается от построенного алгоритма лишь одной деталью. А именно, поскольку рассматриваемое событие S представляется в данном автомате Мили A с начальным состоянием a_1 не множеством M внутренних состояний, а множеством N выходных сигналов, необходимо иначе строить начальный комплекс. Вместо комплекса типа $a_1 \to M$ в этом алгоритме строится комплекс K типа $a_1 \to N$. Комплекс K представляет собою дизъюнкцию всех простых путей с отброшенной последней буквой, которые оканчиваются всевозможными парами $a_{j_{k-1}} x_{i_k}$, дающими при подстановке их в функцию выходов автомата A выходные сигналы, принадлежащие представляющему множеству N. В остальном алгоритм анализа конечных автоматов Мили ничем не отличается от алгоритма анализа конечных автоматов Мура. Нетрудно видеть, что для него оказы-

ваются применимыми все приведенные выше рассуждения и доказательства.

Из рассмотрения построенного в настоящем параграфе алгоритма анализа непосредственно вытекает справедливость следующего предложения.

5.1. *Если событие S представлено в конечном автомате Мили или Мура с m состояниями множеством выходных сигналов (или множеством внутренних состояний), то это событие регулярно и допускает регулярное выражение, циклическая глубина которого не превосходит m.*

Чтобы убедиться в справедливости этого предложения, достаточно заметить, что регулярное выражение для события S, получаемое в результате применения алгоритма анализа, конструируется с помощью $m+1$ расширения регулярного выражения K, циклическая глубина которого равна нулю. При этом после первого расширения циклическая глубина выражения остается равной нулю, а после каждого следующего увеличивается не более чем на единицу. В заключение настоящего параграфа рассмотрим несколько примеров анализа конечных автоматов.

Пример 1. Конечный автомат A с тремя внутренними состояниями 1, 2, 3 и двумя входными сигналами x и y задан таблицей переходов II.5.

Таблица II.5

	1	2	3
x	2	3	1
y	1	2	3

Требуется найти регулярное выражение для события S, представленного в этом автомате множеством состояний (1, 2) при начальном состоянии 1.

Решение. Непосредственно по табл. II.5 находим начальный комплекс $K=1\vee_1 x_2$ и комплексы типов 1, 2 и 3:

$$K_1=\{y\vee x_2x_3x\},$$
$$K_2=\{y\vee x_3x_1x\},$$
$$K_3=\{y\vee x_1x_2x\}.$$

Остальные комплексы мы выписывать не будем, поскольку они легко находятся на основе комплексов K_1, K_2, K. Применяем четыре раза операцию расширения:

1-е расширение:

$$K = 1 \vee_1 x_2 \longrightarrow K_1 \vee K_1 x K_{2[1]} = K_1 (e \vee x K_{2[1]}).$$

2-е расширение:

$$K_1 (e \vee x K_{2[1]}) = \{y \vee x_2 x_3 x\} (e \vee x \{y\}) \longrightarrow$$
$$\longrightarrow \{y \vee x K_{2[1]} x K_{3[1,2]} x\} (e \vee x \{y\}).$$

3-е расширение:

$$\{y \vee x K_{2[1]} x K_{3[1,2]} x\} (e \vee x \{y\}) = \{y \vee x \{y\} x \{y\} x\} (e \vee x \{y\}) = R.$$

Поскольку операция расширения, примененная к выражению R уже не может его изменить, дальнейшие расширения становятся излишними, а выражение

$$R = \{y \vee x \{y\} x \{y\} x\} (e \vee x \{y\})$$

представляет собою искомое регулярное выражение для события S. Заметим, что в этом случае не потребовалось четырех расширений, предусмотренных в алгоритме синтеза. Таким образом, число $m+1$ представляет собою лишь верхнюю грань числа необходимых расширений; в отдельных случаях эта грань может и не достигаться.

Пример 2. Конечный автомат A с четным числом состояний 1, 2,..., $2n$ и с двумя входными сигналами x и y задан таблицей переходов II.6.

Таблица II. 6

	1	2	3	4	5	6	...	$(2n-1)$	$2n$
x	2	1	4	3	6	5		$2n$	$2n-1$
y	1	3	2	5	4	7		$2n-2$	$2n$

Требуется найти регулярное выражение для события S, представленного в автомате A состоянием 1 при начальном состоянии 1.

Решение. Начальный комплекс K состоит из единственного пути $K=1$. Произведя первое расширение,

получим выражение $K(1)=K_1$. Из табл. II.6 видно, что

$$K_1=\{x_2 x \vee y\},\quad K_{2[1]}=\{y_2 y\},$$
$$K_{3[1,\,2]}=\{x_4 x\},\ K_{4[1,\,2,\,3]}=\{y_3 y\},\ \ldots,\ K_{(2n-1)[1,\,2,\,\ldots,\,2n-2]}=$$
$$=\{x_{2n} x\},\ K_{2n[1,\,2,\,\ldots,\,2n-1]}=\{y\}.$$

Произведя второе расширение, мы придем к выражению:

$$K(2)=\{y \vee x K_{2[1]} x\}.$$

После третьего расширения получим выражение

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

Нетрудно видеть, что, продолжая подобным образом, мы на $2n$-м расширении придем к выражению

$$K(2n)=\{y \vee x \{y \{x \{\ldots \{x K_{2n[1,\,2,\,\ldots,\,2n-1]} x\} \ldots\} x\} y\} x\}$$

и на $(2n+1)$-м расширении — к выражению

$$R=K(2n+1)=\{y \vee x \{y \{x \{\ldots \{x \{y\}\, x\} \ldots\} x\} y\} x\},$$

имеющему циклическую глубину $2n$. Это выражение и будет искомым регулярным выражением для события S.

Заметим, что в рассмотренном примере достигается верхняя граница циклической глубины регулярного выражения для события, представленного в автомате, которая была найдена в сформулированном выше предложении 5.1.

Пример 3. Автомат Мили A с двумя внутренними состояниями 1,2, двумя входными сигналами x, y и двумя выходными сигналами z и v задан таблицей переходов II.7 и таблицей выходов II.8.

Таблица II.7

	1	2
x	2	2
y	1	1

Таблица II.8

	1	2
x	z	v
y	z	v

Найти регулярное выражение для события S, представленного в автомате S выходным сигналом v (при начальном состоянии 1).

Решение. Находим начальный комплекс $K = {}_1x_2x \vee {}_1x_2y$, а также комплексы типов 1 и 2:

$$K_1 = \{y \vee x_2y\}, \quad K_2 = \{x \vee y_1x\}.$$

Производим необходимые расширения:

$$K = {}_1x_2x \vee {}_1x_2y \to K_1xK_{2[1]}x \vee K_1xK_{2[1]}y =$$
$$= K_1xK_{2[1]}(x \vee y) = \{y \vee x_2y\}\, x\, \{x\}\, \{x \vee y\} \to$$
$$\to \{y \vee xK_{2[1]}y\}\, x\, \{x\}\, (x \vee y) = \{y \vee x\, \{x\}\, y\}\, x\, \{x\}\, (x \vee y).$$

Последнее выражение и дает искомое выражение для события S:

$$S = \{y \vee x\, \{x\}\, y\}\, x\, \{x\}\, (x \vee y).$$

§ 6. Основной алгоритм синтеза конечных автоматов

Основной задачей настоящего параграфа является построение алгоритма синтеза вполне определенных автоматов по представляемым ими событиям. Сначала мы будем решать так называемую *общую проблему синтеза* вполне определенных конечных автоматов. Приведем точную формулировку этой проблемы.

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

Описываемый ниже алгоритм, решающий указанную проблему, мы будем называть *основным алгоритмом синтеза конечных автоматов*. Напомним, что согласно принятому ранее соглашению представление событий будет рассматриваться с точностью до пустого слова, которое, таким образом, будет играть лишь служебную роль.

Для построения общего алгоритма синтеза достаточно научиться строить по представляемым ими событиям

лишь конечные автоматы Мура. В самом деле, построив автомат Мура B, представляющий заданное множество регулярных событий M, мы можем интерпретировать его как автомат Мили A способом, описанным в § 1 настоящей главы. Как было показано в § 1, автоматы A и B индуцируют одно и то же автоматное отображение, и следовательно, представляют все события исходного множества M одинаковыми множествами своих выходных сигналов.

В свою очередь, решая задачу синтеза автомата Мура, мы можем, в силу результатов § 4, считать, что заданные нам регулярные события требуется представить не множествами выходных сигналов, а множествами (внутренних) состояний автомата.

Для того чтобы перейти от представления событий $R_1,\ldots,R_p$ в автомате Мура множествами состояний к представлению их множествами выходных сигналов, достаточно в качестве выходных сигналов взять различные подмножества заданного множества событий $(R_1,\ldots,R_p)$ и отмечать каждое состояние a автомата множеством всех таких событий, которые представляются этим состоянием (то есть таких событий, которые содержат слова, переводящие автомат из начального состояния в рассматриваемое состояние a). В частности, состояния, не представляющие ни одного из заданных событий, отмечаются пустым множеством событий. Описанный способ отметок состояний автомата Мура B мы назовем *каноническим способом отметок*.

Применяя канонический способ отметок, мы получим, как нетрудно видеть, представление любого заданного события R_i множеством всех тех выходных сигналов, которые в своем составе содержат это событие $(i=1,\ldots, p)$. Исключение может составить лишь пустое слово (если оно входило в заданное событие), поскольку пустое слово не может быть представлено никаким выходным сигналом. Таким образом, при использовании канонического способа отметок заданные события представляются множествами выходных сигналов (отметок) с точностью до пустого слова.

Переходя к построению основного алгоритма синтеза конечных автоматов, условимся, что все те из исходных для алгоритма регулярных выражений $R_1, \ldots, R_p$

(представляющих заданные регулярные события), которые являются многочленами, будут всегда заключены в обычные (неитерационные) скобки. Это условие мы будем называть *условием правильности записи* регулярных выражений. Ясно, что необходимость выполнения этого условия нисколько не ограничивает общности построения.

Местами в правильно записанном регулярном выражении R над алфавитом $\mathcal{X}=(x_1, \ldots, x_n)$ мы условимся называть специально вводимые *знаки раздела* (вертикальные линии), ставящиеся между любыми двумя знаками этого выражения (знаками в выражении R являются буквы алфавита $\mathcal{X}$, символ пустого слова e, знак дизъюнкции, итерационные и обычные скобки). Кроме этих так называемых *разделяющих мест* вводятся еще два места — *начальное* и *конечное*. Первое из них располагается слева от са́мого левого знака выражения R, а второе — справа от са́мого правого знака.

В качестве примера рассмотрим разметку мест в регулярном выражении $R=z\vee x\{y\vee z\}$. Приведенное к правильной форме, это выражение приобретает вид $R=(z\vee x\{y\vee z\})$. После приведения можно произвести разметку мест:

$$\underset{1}{|}\,(\,\underset{2}{|}\,z\,\underset{3}{|}\,\vee\,\underset{4}{|}\,x\,\underset{5}{|}\,\{\,\underset{6}{|}\,y\,\underset{7}{|}\,\vee\,\underset{8}{|}\,z\,\underset{9}{|}\,\}\,\underset{10}{|}\,)\,\underset{11}{|}\,. \qquad (1)$$

Таким образом, данное регулярное выражение имеет всего 11 мест.

Начнем теперь *развертывать* данное регулярное выражение в слово, то есть последовательно, буква за буквой, выписывать какое-либо слово из представляемого им события. При таком развертывании, производимом в соответствии с порядком действий, указанным в выражении, мы будем осуществлять переходы от одного места данного регулярного выражения к другому, отправляясь от начального места и заканчивая конечным местом. Естественно различать два вида таких переходов — *непосредственный переход* и *переход через букву основного алфавита* $\mathcal{X}$, в котором задается представляемое выражением событие.

Рассмотрим в качестве примера процессы развертывания размеченного выше выражения R в принадлежащие представляемому им событию слова z и xyz. При развертыв-

вании выражения в первое слово мы должны осуществить непосредственный переход от начального места 1 к месту 2, переход через букву z от места 2 к месту 3 и, наконец, непосредственный переход от места 3 к конечному месту 11. При развертывании выражения в слово xyz порядок переходов будет следующим: непосредственный переход — от места 1 к месту 4, переход через букву x — от места 4 к месту 5, непосредственный переход — от места 5 к месту 6, переход через букву y — от места 6 к месту 7, непосредственный переход — от места 7 к месту 10, непосредственный переход — от места 10 к месту 8, переход через букву z — от места 8 к месту 9, непосредственный переход — от места 9 к месту 10 и непосредственный переход — от места 10 к месту 11.

Пусть R — регулярное выражение, $q=x_{i_1}x_{i_2}\ldots x_{i_k}$ — произвольное слово во входном алфавите события, представляемого выражением R. Условимся говорить, что место α в выражении R *связано* словом q с местом β в том же выражении, если от места α к месту β можно перейти с помощью чередования любого числа непосредственных переходов и переходов через буквы $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ слова q, взятых по одному разу в том порядке, в каком они входят в слово. Мы будем употреблять также другую терминологию, говоря, что место β *q-следует* за местом α всякий раз, когда место α связано с местом β словом q.

Условимся говорить, что место β *подчинено месту* α, если от места α к месту β можно перейти с помощью одних лишь непосредственных переходов, то есть, иначе говоря, если место α связано с местом β пустым словом e.

Из определения операций и порядка их выполнения в алгебре событий непосредственно вытекают следующие *правила подчинения мест в регулярных выражениях*:

1. *Начальные места всех термов многочлена, помещенного в обычные или итерационные скобки, подчинены месту, находящемуся непосредственно слева от открывающей скобки из пары скобок, объемлющих данный многочлен (многочлен в этом правиле может вырождаться в одночлен, важно лишь, чтобы он был заключен в скобки).*

2. *Место, расположенное непосредственно справа от закрывающей скобки (итерационной или обыкновенной) подчинено конечным местам всех термов многочлена*

(в частном случае, одночлена), заключенного в соответствующие скобки, а в случае итерационных скобок — также месту, расположенному непосредственно слева от соответствующей открывающей скобки.

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

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

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

6. *Каждое место подчинено самому себе.*

7. *Других случаев подчинения мест, кроме тех, которые определяются правилами* 1, 2, 3, 4 *и* 5, *нет.*

В качестве примера использования приведенной системы правил установим отношение подчиненности мест для рассмотренного выше регулярного выражения (1).

Место 1 в этом выражении, кроме себя самого, очевидно, не подчинено никаким другим местам. Тем же самым свойством обладают также места 3, 5, 7 и 9. Места 2 и 4, кроме самих себя, подчинены месту 1 (по правилу 1), места 6 и 8 — месту 5 (по правилу 1) и месту 10 (по правилу 3). Место 10 подчинено (кроме самого себя) местам 5, 7 и 9 (по правилу 2). Наконец, место 11 подчинено (кроме самого себя) местам 3 и 10 (по правилу 2) и местам 5, 7 и 9 (по правилу 5).

Важное значение для построения алгоритма синтеза имеет понятие *основного места* в правильно записанном регулярном выражении. Основными местами в таких выражениях считаются по определению все места, непосредственно с л е в а от которых стоит буква основного алфавита, а также начальное место.

Условимся также называть все места, непосредственно с п р а в а от которых стоит буква основного алфавита, *предосновными.*

Рассмотрим теперь произвольное множество правильно записанных регулярных выражений $R_1, R_2, \ldots, R_p$. Отождествим между собой все начальные места этих выражений и будем рассматривать совокупность различных

множеств их основных мест, включая пустое множество основных мест (не содержащее ни одного места).

Если k — общее число экземпляров букв основного алфавита, входящих в данные выражения $R_1,\ldots, R_p$, то, как следует из элементарной теории множеств и определения основного места, число различных множеств основных мест будет равняться $q=2^{k+1}$.

Построим теперь автомат Мура A, входной алфавит которого совпадает с основным алфавитом $\mathfrak{X}=(x_1,\ldots,x_n)$ выражений $R_1, \ldots, R_p$, а внутренними состояниями служат всевозможные множества $a_1, a_2, \ldots, a_q$ основных мест этих выражений. Состояние $a_i x_j = a_l$, в которое автомат A переходит из состояния a_i под действием входа x_j, определяется как множество a_l всех тех основных мест выражений $R_1,\ldots, R_p$, которые связаны однобуквенным словом x_i хотя бы с одним из основных мест, входящих в множество a_i. Ясно, что таким образом однозначно определяется некоторая функция переходов в автомате A. В частности, в силу этого определения, состояние автомата A, соответствующее пустому множеству основных мест, обладает тем свойством, что оно переводится в самое себя любой входной буквой x_j.

В качестве начального состояния a_1 в построенном автомате выберем множество, состоящее из начального места (которое считается общим для всех выражений $R_1, \ldots, R_p$). Для того чтобы завершить построение автомата, остается лишь отметить его состояния. При этом мы будем пользоваться подмножествами $(R_{i_1}, \ldots, R_{i_r})$ данного нам множества $M=(R_1,\ldots,R_p)$ регулярных выражений (не исключая из рассмотрения пустого множества и самого множества M).

Примем следующее *правило отметок*: *состояние a_i автомата A (множество основных мест выражений $R_1,\ldots,R_p$) отмечается множеством, содержащим все те и только те выражения $R_1, \ldots, R_p$, конечные места которых подчинены хотя бы одному (основному) месту из числа мест, входящих в множество a_i.*

Из определения функции переходов в построенном нами автомате A и из способа отметки его состояний непосредственно вытекает следующее предложение.

Слово q в основном алфавите $\mathfrak{X}=(x_1, \ldots, x_n)$ тогда и только тогда переводит автомат A из начального состоя-

ния a_1 в состояние a_j, отмеченное произвольным множеством, содержащим любое заданное регулярное выражение R_i, когда начальное место выражения R_i связано с конечным местом этого выражения словом q.

Нетрудно видеть также, что слово q тогда и только тогда связывает начальное и конечное места выражения R_i, когда оно принадлежит событию, представляемому этим выражением. Тем самым становится очевидным следующее предложение.

6.1. *Событие S_i, представляемое регулярным выражением R_i, в построенном выше автомате Мура A представляется множеством состояний, содержащим все те и только те состояния, в отмечающие множества которых входит символ R_i $(i=1,2, \ldots, p)$.*

Поскольку в процессе построения автомата A мы использовали канонический способ отметки его состояний, то, как было показано в начале параграфа, можно считать, что автомат A представляет исходные события не только множествами своих состояний, но и множествами выходных сигналов. Но тогда построенный автомат можно интерпретировать как автомат Мили. Кроме того, в процессе построения нами было выяснено, что число состояний построенного автомата равняется 2^{k+1}, где k — общее число букв входного алфавита (считая повторения) в исходных регулярных выражениях. Некоторые из этих состояний могут оказаться недостижимыми и могут, следовательно, быть исключены без изменения индуцируемого автоматом отображения (см. § 1), а, значит, и без изменения представляемых им событий. Таким образом, величину 2^{k+1} следует рассматривать как верхнюю оценку для числа состояний синтезируемого автомата.

Все сказанное можно резюмировать в виде следующей теоремы существования.

6.2. *Существует единый конструктивный прием (основной алгоритм синтеза конечных автоматов), позволяющий для любого конечного множества регулярных событий, заданных своими регулярными выражениями $R_1, \ldots, R_p$, построить конечный автомат Мура A и конечный автомат Мили B, в которых все события множества M представляются (с точностью до пустого слова) некоторыми множествами их выходных сигналов. При этом число различ-*

ных выходных сигналов в автоматах A и B не превышает 2^p, *а число состояний в каждом из них не более чем* 2^{k+1}, *где k — общее число букв входного алфавита (считая повторения), которые входят в заданные выражения* $R_1, \ldots, R_p$.

Из данной теоремы и предложения 5.1 вытекает справедливость сформулированной (но не доказанной) в § 4 теоремы 4.5.

Рассуждения, с помощью которых была доказана теорема 6.2, позволяют оформить основной алгоритм синтеза конечных автоматов в виде ряда четко формулируемых правил. Эти правила мы сформулируем таким образом, чтобы включить в алгоритм также и упрощение синтезируемого автомата за счет исключения недостижимых состояний. Такое исключение можно, очевидно, выполнить, если при построении переходов в автомате, отправляясь от начального состояния, учитывать (и вносить в таблицу) лишь те состояния, которые будут фактически появляться при рассмотрении переходов. Новые состояния перестанут появляться через конечное число шагов, поскольку общее число возможных состояний конечно.

Правила основного алгоритма синтеза конечных автоматов

Правило 1. *Заданные регулярные события (число которых предполагается конечным) представляются правильно записанными регулярными выражениями* $R_1, \ldots R_p$. *Все места (как основные, так и неосновные) в этих выражениях отмечаются вертикальными черточками.*

Правило 2. *Каждому основному месту в выражениях* $R_1, \ldots, R_p$ *приписывается в качестве индекса неотрицательное целое число. При этом всем начальным местам приписывается один и тот же индекс, а именно индекс нуль. Что же касается остальных основных мест, то они нумеруются в произвольном порядке натуральными числами 1, 2, ... Все введенные здесь индексы называются основными. Основные индексы подписываются под вертикальными чертами соответствующих им (основных) мест и подчеркиваются снизу общей для каждого из выражений* $R_1, \ldots, R_p$ *горизонтальной разделительной чертой.*

Правило 3. *Индекс (основной) каждого основного места* α *распространяется в качестве неосновного индекса на все места (как основные, так и неосновные), подчиненные месту* α, *но отличные от него самого (правила, определяющие подчинение мест, были сформулированы в настоящем параграфе). При этом каждое место* β *получает, вообще говоря, некоторое множество неосновных индексов. Все индексы этого множества подписываются в произвольном порядке под вертикальной чертой, соответствующей месту* β, *ниже разделительной горизонтальной черты. Все индексы (как основные, так и неосновные), относящиеся к любому предосновному месту, заключаются в общую рамку.*

Правило 4. *Строится таблица переходов некоторого автомата Мура A. В качестве состояний этого автомата берутся подмножества множества всех основных индексов. При этом подмножество, состоящее из основных индексов* $i_1,\ldots,i_k (k \geqslant 1)$, *мы будем обозначать через* $i_1 \vee i_2 \vee \ldots \vee i_k$, *а пустое множество основных индексов — звездочкой (соответствующее ему состояние автомата A называется пустым). В качестве начального состояния выбирается состояние* 0 *и со столбца, соответствующего этому состоянию начинается построение таблицы переходов. Столбцы, соответствующие остальным состояниям, выписываются в произвольном порядке, но лишь после того, как обозначающие их состояния уже появились в выписанных ранее столбцах таблицы. Строки таблицы обозначаются (в произвольном порядке) различными буквами входного алфавита заданного множества событий. На пересечении произвольной* (x_i-*й*) *строки и произвольного* (a_j-*го*) *столбца таблицы выписывается состояние (множество основных индексов), состоящее из основных индексов всех тех и только тех основных мест, которые* x_i-*следуют за предосновными местами, в числе индексов которых (как основных, так и неосновных) находится хотя бы один индекс, принадлежащий состоянию* a_j *(в случае несуществования основных мест с требуемыми свойствами на соответствующем месте таблицы выписывается пустое состояние).*

Правило 5. *Каждое из состояний* $i_1 \vee i_2 \vee \ldots \vee i_k$ $(k \geqslant 1)$, *которые обозначают столбцы таблицы переходов, отмечается множеством* $(R_{j_1}, \ldots, R_{j_m})$ *всех символов тех*

и только тех регулярных выражений $R_1, \ldots, R_p$, конечные места которых содержат в числе своих индексов (как основных, так и неосновных) хотя бы один из индексов $i_1, \ldots, i_{j_k}$. Пустое состояние отмечается пустым множеством регулярных выражений $R_1, \ldots, R_p$; его мы будем обозначать через ( ). С помощью введенных отметок, принимаемых за выходной алфавит, строится отмеченная таблица переходов искомого конечного автомата Мура A.

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

Построенные автоматы A и B представляют заданные события множествами своих состояний и (с точностью до пустого слова) множествами своих выходных сигналов. Событие, заданное регулярным выражением R_i, представляется множеством всех тех и только тех состояний, которые отмечены множествами, содержащими в качестве своих элементов выражение R_i. Это же событие (за вычетом лишь пустого слова, если оно содержится в событии) представляется в автоматах A и B множеством всех тех и только тех выходных сигналов (множеств, состоящих из выражений $R_1, \ldots, R_p$), которые содержат в своем составе выражение $R_i (i=1, \ldots, p)$. Множество состояний, отмеченных пустым множеством ( ), или выходной сигнал ( ) представляет событие, состоящее из всех слов входного алфавита, не вошедших в заданные события.

Обоснование правильности описанного алгоритма содержится в доказательстве предложения 6.1, в свойствах канонического способа отметок, описанных в начале настоящего параграфа, и в методе интерпретации автоматов Мура как автоматов Мили, развитом в § 1.

Необходимо отметить еще, что в процессе синтеза автомата или по окончании этого процесса производят обычно переобозначение состояний и выходных сигналов с целью упрощения записи таблиц переходов и выходов. Обычно при переобозначении состояний мы

будем просто нумеровать их натуральными числами 1,2, ..., используя для обозначения начального состояния единицу. В некоторых случаях, впрочем, оказывается целесообразным обозначать состояния числами 0, 1, 2, ...; при этом начальное состояние обозначается всегда нулем.

Рассмотрим теперь пример синтеза конечного автомата в соответствии с описанным алгоритмом. Требуется построить конечный автомат, в котором для входного алфавита $\mathfrak{X}$, состоящего из двух букв x и y, были бы представлены два события: событие R_1, состоящее из всех слов в алфавите $\mathfrak{X}$, в которых все буквы x предшествуют всем буквам y, и событие R_2, состоящее из всех слов в алфавите $\mathfrak{X}$, которые кончаются буквой x.

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

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

После разметки мест эти выражения приобретут вид

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

Применяем правило 2, в результате чего возникают следующие выражения:

$$R_1'' = \underset{0}{|}\{\underset{1}{|}x|\}\underset{2}{|}\{|y|\}|, \quad R_2'' = \underset{0}{|}\{\underset{3}{|}x|\vee\underset{4}{|}y|\}\underset{5}{|}x|.$$

В результате применения правила 3 мы получаем:

$$R_1''' = \underset{0}{|}\{\underset{1}{|}x|\}\underset{2}{|}\{|y|\}|, \quad R_2''' = \underset{0}{|}\{\underset{3}{|}x|\vee\underset{4}{|}y|\}\underset{5}{|}x|.$$

$$\begin{array}{|c|}\hline 0 \\ 1 \\ \hline\end{array} \quad \begin{array}{c} 0 \\ 1 \end{array} \quad \begin{array}{|c|}\hline 0 \\ 1 \\ 2 \\ \hline\end{array} \quad \begin{array}{c} 0 \\ 1 \\ 2 \end{array} \qquad \begin{array}{|c|}\hline 0 \\ 3 \\ 4 \\ \hline\end{array} \quad \begin{array}{|c|}\hline 0 \\ 3 \\ 4 \\ \hline\end{array} \quad \begin{array}{|c|}\hline 0 \\ 3 \\ 4 \\ \hline\end{array}$$

Применение правила 4 приводит к построению таблицы переходов II.9.

Применение правила 5 дает нам отмеченную таблицу переходов II.10.

Обозначая выходные сигналы ( ), (R_1), (R_2), и (R_1, R_2) соответственно через z, u, v, w и нумеруя состояния, мы приходим к отмеченной таблице переходов II.11.

Таблица II. 9

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

Таблица II. 10

	(R_1)	(R_1, R_2)	(R_1)	(R_2)	( )
	0	1 ∨ 3 ∨ 5	2 ∨ 4	3 ∨ 5	
x y	1 ∨ 3 ∨ 5 2 ∨ 4	1 ∨ 3 ∨ 5 2 ∨ 4	3 ∨ 5 2 ∨ 4	3 ∨ 5 4	3 ∨ 5 4

При интерпретации построенного автомата как автомата Мили в соответствии с правилом 6 мы получим таблицу его выходов 11.12.

Таблица II. 11

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

Таблица II. 12

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

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

Легко видеть, что при представлении любого множества событий во вполне определенном автомате оказываются автоматически представленными всевозможные

пересечения и объединения событий этого множества. Нетрудно видеть также, что будут представлены и все дополнения этих событий (см. § 3).

В случае же применения описанного выше алгоритма синтеза окажутся представленными также и пополнения (см. § 3) исходных событий.

Действительно, нетрудно видеть, что пополнение события, задаваемого любым из исходных регулярных выражений R_i, будет представлено в получаемых в результате синтеза автоматах A и B множеством всех тех и только тех состояний, которые содержат в своем составе индекс хотя бы одного из основных мест выражения R_i.

Поскольку выражения $R_1, \ldots, R_p$ в описанном алгоритме синтеза могут представлять любые регулярные события и поскольку события, представленные в конечных автоматах, непременно регулярны (ср. 5.1), то тем самым доказано следующее предложение.

6.3. *Дополнение и пополнение любого регулярного события, а также пересечение любого конечного множества регулярных событий являются регулярными событиями.*

§ 7. Усовершенствования основного алгоритма синтеза

Описанный выше алгоритм синтеза конечных автоматов допускает ряд уточнений и изменений, позволяющих упрощать синтезируемый автомат. Первое уточнение преследует цель сократить количество используемых основных индексов и уменьшить, по возможности, число состояний в синтезируемом автомате. Уточнение это опирается на использование понятия комплекса регулярных выражений.

Комплексом регулярных выражений K называется любое конечное множество регулярных выражений $R_1, \ldots, R_p$ в одном и том же конечном алфавите (называемом входным алфавитом комплекса), у которых размечены и снабжены индексами все основные места. При этом различные основные места в выражениях $R_1, \ldots, R_p$ могут иметь один и тот же основной индекс. Всякое множество, состоящее из всех основных мест в выражениях $R_1, \ldots, R_p$, которые имеют один и тот же основной индекс k, называется *основ-*

ным местом комплекса K; индекс k называется *основным индексом этого места*. Начальные места всех выражений $R_1, \ldots, R_p$ всегда снабжаются одинаковым индексом 0 и составляют *начальное место комплекса* K.

Если некоторое основное место α комплекса K состоит из основных мест $\alpha_1, \alpha_2, \ldots, \alpha_k$ в выражениях $R_1, \ldots, R_p$, составляющих комплекс, то местами, п о д ч и н е н н ы м и м е с т у α в к о м п л е к с е K, считаются все места в в ы р а ж е н и я х $R_1, \ldots, R_p$, подчиненные хотя бы одному из мест $\alpha_1, \ldots, \alpha_k$ в этих выражениях. Конечными и предосновными местами комплекса считаются, соответственно, конечные и предосновные места составляющих его выражений. Будем говорить, что основное место α комплекса K x_i-*следует* за предосновным местом β, если хотя бы одно из составляющих место α основных мест в выражениях $R_1, \ldots, R_p$ x_i-*следует* за местом β.

Говорят, что слово q входного алфавита *связывает* основное место α комплекса K с основным местом β того же комплекса, если оно связывает некоторое основное место в одном из выражений $R_1, \ldots, R_p$, входящее в место α, с каким-либо основным местом того же самого выражения, входящим в место β. Мы в этом случае будем также говорить, что место β q-*следует* за местом α.

Основные места в данном комплексе K регулярных выражений называются *соответственными*, если множества слов входного алфавита, связывающих начальное место комплекса с каждым из этих мест, одинаковы. Основные места $\alpha_1, \ldots, \alpha_k$ в комплексе K называются *подобными*, если в комплексе K им подчинены одинаковые множества конечных мест и если для любой буквы x входного алфавита комплекса множества $M_1, \ldots, M_k$ основных мест комплекса, x-следующих за местами $\alpha_1, \ldots, \alpha_k$, одинаковы, либо становятся одинаковыми при замене входящих в них мест $\alpha_2, \ldots, \alpha_k$ местом α_1.

Мы будем применять к комплексам регулярных выражений так называемую *операцию отождествления мест*. Сущность этой операции заключается в том, что некоторому множеству основных мест данного комплекса K, имевших различные основные индексы, приписывается один и тот же индекс k и, следовательно, все основные места в регулярных выражениях комплекса, входившие в

отождествляемые места комплекса K, составят новое основное место комплекса, имеющее основной индекс k.

Правила 1 и 2 основного алгоритма синтеза конечных автоматов представляют собою не что иное, как правила написания некоторого комплекса K регулярных выражений для заданного множества регулярных событий. В этом комплексе каждое основное место, за исключением начального места, состоит из единственного основного места в выражениях $R_1, \ldots, R_p$, составляющих комплекс. Таким образом, в комплексе K_0 не проведено никакого отождествления мест, за исключением предусмотренного в определении всякого комплекса отождествления начальных мест входящих в комплекс выражений.

Все последующие правила (то есть правила 3—6) основного алгоритма синтеза, оперируют с местами и с их индексами в этом комплексе K_0, который мы будем называть *нулевым*. На практике оказывается возможным, однако, прежде чем переходить к правилам 3—6 алгоритма синтеза, применить к нулевому комплексу K_0 операцию отождествления мест так, чтобы последующее применение правил 3—6 алгоритма синтеза к новому комплексу привело к построению автоматов, представляющих исходные события $R_1, \ldots, R_p$.

Нетрудно показать, что основной алгоритм синтеза конечных автоматов может быть дополнен следующим правилом.

Правило 2а. *К комплексу K_0 заданных регулярных выражений $R_1, \ldots, R_p$, полученному по правилам 1—2 основного алгоритма синтеза конечных автоматов, можно применять последовательно, шаг за шагом, операцию отождествления соответственных мест и операцию отождествления подобных мест. Последующие правила* (правила 3—6) *алгоритма синтеза применяются не к нулевому комплексу K_0, а к комплексу, в который он превратится после выполнения любого числа таких отождествлений.*

Следует подчеркнуть, что операцию отождествления мест следует осуществлять именно последовательно, не допуская одновременного (на одном шаге) отождествления и по признаку подобия и по признаку соответствия, ибо после отождествления по одному признаку (например, по признаку соответствия) места, бывшие ранее подоб-

ными, могут перестать быть таковыми, и наоборот. Что же касается порядка отождествления, то он безразличен: можно начинать отождествление как с соответственных, так и с подобных мест.

Покажем теперь, что применение правила 2а не влияет на правильность представления событий в синтезируемом автомате. Действительно, из определения способа построения перехода в синтезируемом автомате (правило 3) и определения соответственных мест непосредственно вытекает, что в каждом состоянии автомата, содержащем индекс какого-либо из основных мест, будут одновременно присутствовать индексы всех соответственных ему мест. Следовательно, операция отождествления соответственных мест сводится к тому, что некоторое множество $i_1 \vee i_2 \vee \ldots \vee i_k$ основных индексов, одновременно входящих или одновременно не входящих в любое из используемых состояний автомата, заменяется одним индексом i. Такая операция не меняет, очевидно, ни числа состояний автомата, ни переходов между ними. Она не влияет также и на отметку состояний, в силу принятого нами определения подчинения мест в комплексе; основному месту с индексом i будет подчинено то же самое множество конечных мест выражений $R_1, \ldots, R_p$, что и совокупности основных мест $i_1, i_2, \ldots, i_k$ до отождествления. Следовательно, операция отождествления соответственных мест, приводя к экономии основных индексов, не изменяет (с точностью до обозначений состояний) самого синтезируемого автомата.

Возможность отождествления подобных мест основана на несколько иных соображениях.

Выясним прежде всего, как отражается операция отождествления мест в комплексе на применение к этому комплексу правил 3—5 основного алгоритма синтеза автомата Мура. Пусть K — некоторый комплекс регулярных выражений $R_1, \ldots, R_p$; $\alpha_1, \alpha_2, \ldots, \alpha_k$ — основные индексы каких-либо его основных мест; γ — основной индекс соответствующего им места в комплексе K_1, полученном в результате отождествления мест $\alpha_1, \ldots, \alpha_k$ в комплексе K. Обозначим через A и A_1 автоматы Мура, полученные в результате применения правил 3—5 основного алгоритма синтеза, соответственно к комплексам K и K_1

Из определения операции отождествления и законов подчинения мест в комплексе легко усмотреть способ, которым отмеченная таблица переходов автомата A_1 может быть получена из соответствующей таблицы автомата A. Для этой цели достаточно, как это вытекает из правил 4 и 5 алгоритма синтеза, в отмеченной таблице переходов автомата A индексы $\alpha_1, \ldots, \alpha_k$ заменить всюду индексом γ и произвести слияние тех столбцов, которые после такой замены становятся обозначенными одинаковыми состояниями. Слияние столбцов происходит путем образования дизъюнкции соответствующих друг другу (расположенных в одной строке) элементов этих столбцов, а отметки вновь полученного таким образом столбца производятся объединением множеств, отмечавших слитые столбцы состояния (общего для всех слитых столбцов).

Вообще говоря, подобная операция изменяет представляемые автоматом события. Однако в случае, когда отождествляемые места (с основными индексами $\alpha_1, \ldots, \alpha_k$) подобны друг другу, из определения подобия мест непосредственно вытекает, что сливающиеся в один столбец таблицы переходов автомата A будут одинаковыми и одинаково отмеченными.

Как будет показано в § 10, посвященном минимизации абстрактных автоматов, слияние столбцов при таких условиях приводит к появлению автомата, эквивалентного исходному, а следовательно, и представляющего те же самые события.

Таким образом, операция отождествления подобных мест в алгоритме синтеза автоматов оказывается действительно возможной. Следует подчеркнуть, что в отличие от операции отождествления соответственных мест эта операция, как правило, приводит к уменьшению числа состояний в синтезируемом автомате.

При практическом применении правила 2а необязательно, разумеется, проводить все возможные отождествления; можно ограничиться лишь некоторыми из них. Возникающие таким образом алгоритмы (с тем или иным числом предварительных отождествлений мест) мы будем называть *усовершенствованными алгоритмами синтеза конечных (вполне определенных) автоматов.*

В случае, когда стоит задача синтеза лишь (конечного) автомата Мили (а не автоматов Мура и Мили), то в общий алгоритм синтеза могут быть внесены еще некоторые изменения, приводящие, вообще говоря, к новому уменьшению числа состояний синтезируемого автомата.

Прежде всего в случае синтеза автоматов Мили нет необходимости отмечать состояния автомата в соответствии с правилом 5 и затем применять правило 6; последовательное применение правил 5 и 6 можно заменить применением одного правила 5а, относящегося к построению таблицы выходов синтезируемого автомата Мили.

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

Легко видеть, что применение правила 5а действительно эквивалентно последовательному применению двух правил 5 и 6 основного алгоритма синтеза. Алгоритм, состоящий из правил 1, 2, 3, 4 и 5а, мы будем называть *особым алгоритмом синтеза конечных автоматов Мили.*

Нетрудно понять, что этот алгоритм допускает отождествление соответственных мест в нулевом комплексе исходных регулярных выражений, полученном по правилам 1 и 2 основного алгоритма. Что же касается подобных мест, то,— ввиду отсутствия в рассматриваемом случае отметок состояний,— их необходимо заменить так называемыми *квазиподобными местами*, определяемыми следующим образом:

7.1. *Основные места $\alpha_1, \ldots, \alpha_k$ любого заданного комплекса регулярных выражений называются квазиподобными, если для любой буквы x входного алфавита комплекса K множества конечных мест комплекса, x-следующих за основными местами $\alpha_1, \ldots, \alpha_k$, совпадают между собой,*

а множества $M_1, \ldots, M_k$ *основных мест комплекса, х-следующих за местами* $\alpha_1, \ldots, \alpha_k$, *либо одинаковы, либо становятся одинаковыми после замены входящих в множества* $M_1, \ldots, M_k$ *мест* $\alpha_2, \ldots, \alpha_k$ *местом* α_1.

Рассмотрим комплекс K регулярных выражений и комплекс K_1, полученный из него в результате отождествления подобных мест с индексами $\alpha_1, \ldots, \alpha_k$ и присваивания получившемуся в результате отождествления мест основному месту индекса α_1. Пусть A и A_1 — автоматы Мили, построенные в результате применения правил 3, 4, 5а к комплексам K и K_1. Из правил 3,4 и 5а вытекает, что таблица переходов автомата K_1 получается в результате дизъюнкции столбцов таблицы переходов автомата K, обозначающихся одинаковыми (после замены индексов $\alpha_2, \ldots, \alpha_k$ индексом α_1) состояниями. Аналогичное явление происходит и в таблице выходов. Из определения же квазиподобных мест вытекает, что объединяемые таким образом столбцы как в таблице переходов, так и в таблице выходов оказываются одинаковыми. Как показано в § 10, посвященном минимизации автоматов, слияние столбцов при таких условиях не меняет отображения, индуцируемого автоматом, а следовательно, не меняет и представляемых им событий. Следовательно, можно сформулировать следующее правило отождествления мест в случае синтеза автоматов Мили.

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

Заметим, что согласно приведенному выше определению, квазиподобными являются все т у п и к о в ы е места комплекса, т. е. такие его основные места, которые не связаны ни одной буквой входного алфавита ни с одним основным местом комплекса. Если в синтезированном автомате Мили A имеются состояния, образуемые основными индексами лишь одних тупиковых мест, то в таблице переходов автомата A столбцы, обозначенные всеми

такими состояниями, будут состоять из одних лишь символов пустого состояния *, а соответствующие столбцы в таблице выходов — из одних лишь символов пустого выходного сигнала. Как уже отмечалось, такие столбцы могут быть отождествлены без нарушения правильности представления событий. Ясно также, что при вычеркивании индексов тупиковых мест из других состояний автомата состояниям, полученным в результате такого вычеркивания, будут соответствовать, в силу правил 4 и 5а, те же самые столбцы в таблице переходов и в таблице выходов, что и до вычеркивания индексов.

Тем самым доказано, что при синтезе автоматов Мили наряду с правилом 2б может применяться также следующее правило.

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

Алгоритмы синтеза, использующие правила 1, 2, 2б, 2в, 3, 4, 5а, мы будем называть *усовершенствованными особыми алгоритмами синтеза конечных автоматов Мили.* Мы имеем дело здесь не с одним алгоритмом, а с целым комплексом алгоритмов, поскольку правила 2б и 2в сформулированы так, что соответствующие им упрощения могут и не проводиться или проводиться лишь частично.

Отметим также, что усовершенствованные алгоритмы синтеза автоматов Мили, в отличие от основного алгоритма, уже не приводят к автоматам, в которых заданные события представлены множествами состояний, а лишь к таким автоматам, в которых эти события представлены множествами выходных сигналов.

Продемонстрируем теперь работу усовершенствованных алгоритмов синтеза на примере.

Пример 1. Найти конечные автоматы Мура и Милп, представляющие регулярные события в алфавите (x, y), заданные следующими регулярными выражениями:

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

Решение. 1) Выпишем нулевой комплекс K_0 заданных регулярных выражений:

$$\underset{0}{R} = \underset{0}{|}\underset{1}{x}\underset{2}{|}\{\underset{2}{y}|\}, \quad \underset{0}{P} = \underset{0}{|}\underset{3}{x}\underset{4}{|}\underset{4}{x}|.$$

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

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

$$R = \underset{0}{|}x\underset{1}{|}\{y\underset{2}{|}\}, \quad P = \underset{0}{|}x\underset{1}{|}x\underset{4}{|}.$$

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

Применяя теперь правило 3, мы получим размеченный комплекс:

$$R = \underset{\substack{\boxed{0}\\ \boxed{1}\\ \boxed{2}}}{|}x\underset{\substack{1\\ \boxed{2}}}{|}\{y\underset{\substack{2\\ \boxed{2}}}{|}\}, \quad P = \underset{\substack{\boxed{0}\\ \boxed{1}}}{|}x\underset{\substack{\boxed{1}\\ \boxed{2}}}{|}x\underset{4}{|}.$$

По правилам 4 и 5 получаем отмеченную таблицу переходов автомата Мура II.13.

Таблица II. 13

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

Таблица II. 14

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

После переобозначений:

$$0 \to 1\ ^{1}) \quad 1 \to 2, \quad 2 \to 3, \quad 4 \to 4, \quad * \to 5,$$
$$(R) \to u, \quad (P) \to v, \quad (\,) \to w,$$

[1]) Стрелка употребляется в качестве знака переобозначения. Например, $0 \to 1$ означает, что вместо обозначения 0 в отмеченной таблице переходов II. 13 берется обозначение 1.

мы получаем отмеченную таблицу переходов автомата Мура II.14. Событие R представлено в этом автомате выходным сигналом u, а событие P — выходным сигналом v. Применяя правило 6, мы получаем таблицы переходов и выходов автомата Мили II.15 и II.16.

Таблица II. 15

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

Таблица II. 16

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

2) Если бы мы вначале отождествили не соответственные, а подобные места, то пришли бы к комплексу:

$$R = \underset{0}{|}x\underset{1}{|}\{y\underset{1}{|}\}, \quad P = \underset{0}{|}x\underset{3}{|}x_4.$$

После такого отождествления места 1 и 3 перестают быть соответственными, поскольку начальное место связывается с местом 3 лишь однобуквенным словом x, а с местом 1 — множеством слов вида $x\{y\}$. Отмеченная таблица автомата Мура C, построенная по этому комплексу, будет иметь вид (см. табл. II.17):

Таблица II. 17

	( )	(R)	(R)	(P)	( )
	0	1	1 v 3	4	*
x	1 v 3	*	4	*	*
y	*	1	1	*	*

Нетрудно показать, что автоматы A и C изоморфны, так что в рассматриваемом случае отождествление подобных мест не привело к уменьшению числа состояний.

3) Рассмотрим, наконец, как в этом примере работает упрощенный особый алгоритм синтеза автоматов Мили.

Применяя правила 2б и 2в, мы после отождествления соответственных мест получим следующий комплекс:

$$R = \underset{0}{|}x\underset{1}{|}\{y\underset{2}{|}\}, \quad P = \underset{0}{|}x\underset{1}{|}x.$$

Применяя правила 3 и 4, мы находим таблицу переходов автомата Мили D (табл. II.18). Применение правила 5а дает таблицу выходов II.19.

Таблица II. 18

	0	1	2	*
x	1	*	*	*
y	*	2	2	*

Таблица II. 19

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

У автомата D — лишь четыре состояния, в то время как у ранее синтезированных автоматов было пять состояний.

При построении всех описанных до сих пор алгоритмов мы ставили своей задачей получать вполне определенные автоматы. Можно, однако, достичь дальнейшего упрощения синтезируемых автоматов, если считать их не вполне определенными, а частичными автоматами. Для этой цели необходимо условиться относительно того, какие слова при синтезе автоматов следует считать допустимыми, а какие запрещенными.

Будем сначала рассматривать произвольный автомат Мура A. Естественно назвать *неопределенными* все те состояния автомата A, в которые автомат может перейти (отправляясь от начального состояния) только под действием запрещенных слов. Это название связано с тем, что автомат A не изменит, как легко видеть, своих реакций на допустимые слова, если его неопределенные состояния отождествить с любыми другими состояниями автомата. Можно поэтому не определять переходов в автомате,

переводящих его в неопределенные состояния. Поскольку после этого указанные состояния делаются недостижимыми, их можно просто напросто исключить из автомата (см. § 1). Если после такого исключения возникнут новые недостижимые состояния, их также можно исключить.

В силу принятого нами соглашения, мы можем всегда относить пустое слово к числу запрещенных слов. Тогда начальное состояние автомата также может оказаться неопределенным (это, в частности, будет во всех случаях, когда начальное состояние не встречается среди элементов таблицы переходов). В этом случае все переходы в автомате, переводящие его в начальное состояние, можно, как и в вышеупомянутом случае, считать неопределенными, однако исключать начальное состояние из числа состояний автомата, разумеется, нельзя.

Все сказанное можно сформулировать в виде следующего правила.

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

Это правило может сочетаться с любым из описанных алгоритмов, включающих в себя в качестве заключительного или промежуточного этапа синтез конечного автомата Мура. Таким образом, могут быть получены алгоритмы, заключающиеся в последовательном применении правил 1,2 (2а), 3, 4, 5 (5б), (6) [1]). Если требуется синтезировать только автомат Мура, то можно исключить правило 6, если ставится задача синтеза какого-нибудь (а не обя-

[1]) Скобки служат для выделения тех правил, которые могут как применяться, так и не применяться в зависимости от цели, которая ставится перед алгоритмом.

зательно самого простого) автомата, то можно исключить правила 2а и 5б.

Алгоритмы, получаемые добавлением правила 5б в основной и упрощенный алгоритмы синтеза конечных автоматов, мы также будем называть основным и упрощенным; при этом в случае необходимости мы будем добавлять к названию алгоритма указание о виде синтезируемого автомата: если будет указан вполне определенный автомат — это будет означать алгоритм без правила 5б, если же будет упомянут частичный автомат — это будет означать алгоритм с правилом 5б.

Автомат Мили — в случае, когда он строится обычным путем (как интерпретация предварительно построенного автомата Мура) — наследует те же неопределенности, которые вносятся правилом 5б в порождающий его (в соответствии с правилом 6) автомат Мура. Если же автомат Мили возникает прямым путем — в результате применения особых алгоритмов синтеза автоматов Мили,— то введение неопределенности в его таблицы переходов и выходов требует дальнейших рассмотрений.

Может случиться, во-первых, что некоторые выходные сигналы в автомате Мили B могут возникать лишь под действием запрещенных слов. Такие выходные сигналы естественно назвать *неопределенными*, так как замена их любым другим сигналом не изменит, очевидно, реакцию автомата B на допустимые слова. Это означает, что в таблице выходов автомата B все неопределенные выходные сигналы можно заменить черточками. Применяя предложение 1.6, можно распространить возникшую таким образом неопределенность таблицы выходов автомата B на его таблицу переходов.

Предположим далее, что имеется допустимое входное слово p такое, что добавление к нему любой буквы входного алфавита превращает его в запрещенное слово. Всякое слово, обладающее таким свойством, естественно назвать *предзапрещенным*. Все состояния автомата Мили B, в которые этот автомат может переходить из начального состояния a_0 только под действием запрещенных или предзапрещенных слов, мы назовем *неопределенными*. Переход автомата в неопределенное состояние a означает, что все последующие переходы автомата — и появление

соответствующих им выходных сигналов — будут совершаться под действием запрещенных слов и поэтому могут быть определены как угодно. Определяя эти переходы и выходные сигналы так, как они определены, скажем, для начального состояния, мы можем отождествить все неопределенные состояния с начальным состоянием, не изменив реакции автомата на допустимые слова. Это означает, очевидно, возможность замены всех вхождений неопределенных состояний в таблицу переходов автомата черточками и вычеркивания всех столбцов (за исключением начального), обозначенных неопределенными состояниями, из таблиц переходов и выходов автомата.

Легко видеть, что справедливо следующее предложение.

7.2. *Все состояния автомата Мили, для которых функция выходов не определена при всех входных сигналах, и все состояния автомата Мура, для которых не определена сдвинутая функция выходов, являются неопределенными.*

Это предложение позволяет находить неопределенные состояния по неопределенным выходным сигналам.

Все рассуждения относительно неопределенных выходных сигналов и неопределенных состояний для автоматов Мили могут быть резюмированы в виде следующего правила.

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

Это правило может сочетаться с любым из описанных алгоритмов синтеза (дающим в результате автомат Мили), выступая в качестве последнего из выполняемых правил. Обычно, однако, оно применяется лишь в сочетании с осо-

бы ми алгоритмами синтеза автоматов Мили. Возникающие таким образом алгоритмы состоят из правил 1, 2, (2б), (2в), 3, 4, 5а (6а) (выполнение правил, номера которых заключены в скобки, не обязательно). Условимся, что добавление правила 6а в описанные выше особые алгоритмы синтеза автоматов Мили (обычный и упрощенные) не меняет их названий. В случае необходимости уточняют, синтез какого автомата (частичного или вполне определенного) имеется в виду.

Рассмотрим теперь основные способы задания множеств допустимых слов и областей запрета, которые встречаются на практике. Для каждого из этих способов применительно к автоматам, полученным с помощью любого из описанных выше алгоритмов синтеза вполне определенных автоматов, укажем те выходные сигналы и состояния, которые будут при этом неопределенными. Правильность этих указаний легко проверить, анализируя соответствующие правила описанных алгоритмов синтеза.

1-й способ. *Допустимыми являются все слова, входящие хотя бы в одно из заданных событий* $R_1, \ldots, R_p$, *и произвольные начальные отрезки таких слов. Все остальные слова составляют область запрета.*

Неопределенным в этом случае (как для автоматов Мура, так и для автоматов Мили) всегда является пустое состояние (то есть состоящее из пустого множества основных индексов). Начальное состояние является неопределенным в случае, когда оно не встречается среди элементов таблицы переходов автомата.

2-й способ. *Допустимым является всякое слово, содержащееся в одном и только в одном (зависящем от выбора слова) из заданных регулярных событий* $R_1, \ldots, R_p$, *причем такое, что всякий его непустой начальный отрезок также содержится точно в одном (зависящем от выбора этого отрезка) из заданных событий* $R_1, \ldots, R_p$ *(необязательно, чтобы слово и его начальные отрезки входили в одно и то же событие). Все остальные слова составляют область запрета.*

Неопределенными в этом случае являются все выходные сигналы, состоящие из двух и более событий $R_1, \ldots, R_p$, а также пустой выходной сигнал. Неопределенными состояниями являются для автоматов Мура все состояния,

отмеченные неопределенными выходными сигналами (в частности, пустое состояние), и начальное состояние, если оно не входит в качестве одного из элементов в таблицу переходов. В случае автоматов Мили неопределенным будет пустое состояние. Остальные неопределенные состояния могут быть получены с помощью предложения 7.2.

3-й способ. *Допустимым является всякое слово, обладающее тем свойством, что как оно само, так и любой из его непустых начальных отрезков содержатся не более чем в одном из заданных регулярных событий* $R_1, \ldots, R_p$. *Все остальные слова составляют область запрета.*

Неопределенными в этом случае будут те же выходные сигналы и состояния, что и в предыдущем случае, за исключением пустого состояния и пустого выходного сигнала.

4-й способ. *Область запрета задана явно — в виде одного из регулярных выражений* S, *входящих в заданное множество регулярных выражений.*

Неопределенными в этом случае будут все выходные сигналы, в состав которых входит символ области запрета S. Неопределенные состояния характеризуются с помощью предложения 7.2. В частности, для автоматов Мура неопределенными будут все состояния, отмеченные неопределенными выходными сигналами. Начальное состояние является неопределенным, если оно не входит в число элементов таблицы переходов автомата.

Заметим, что неопределенность начального состояния в случае, когда оно не входит в число элементов таблицы переходов автомата, имеет место при любой области запрета.

В заключение параграфа рассмотрим примеры применения упрощенных алгоритмов синтеза для случая синтеза частичных автоматов.

Пример 2. События заданы регулярными выражениями $P=\{x\}\{y\}$, $Q=\{x\}(x \vee y)$. Построить представляющие эти события частичные автоматы Мура и Мили при условии, что область запрета задана первым из указанных выше способов.

Решение. Отождествляя сначала соответственные, а потом подобные места, мы приходим к следующему (размеченному) комплексу. (Заметим, что отождествление

подобных мест 0 и 1 здесь не производится.)

$$P = \underset{0}{|\{|x|\}|}\underset{1}{|\{|y|\}|}\underset{2}{|}, \qquad Q = \underset{0}{|\{|x|\}|}\underset{1}{(|x|\vee|y|)|}\underset{3}{|}.$$

$$\begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ & & 2 & 2 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ \hline \end{array}\ 3$$

Правила 4 и 5 приводят к отмеченной таблице переходов автомата Мура (табл. II.20).

Таблица II. 20

	P	(P, Q)	(P, Q)	P	$( )$
	0	1 v 3	2 v 3	2	*
x	1 v 3	1 v 3	*	*	*
y	2 v 3	2 v 3	2	2	*

Неопределенными являются здесь пустое и начальное состояние. Применяя правило 5б и вводя переобозначения $0 \to 1$, $1 \vee 3 \to 2$, $2 \vee 3 \to 3$, $2 \to 4$, $P \to u$, $(P, Q) \to v$, мы приходим к следующей отмеченной таблице переходов искомого частичного автомата Мура (табл. II.21):

Таблица II. 21

	—	v	v	u
	1	2	3	4
x	2	2	—	—
y	3	3	4	4

Применяя правило 6 основного алгоритма, получим автомат Мили C, заданный своими таблицами переходов и выходов (табл. II.22 и II.23).

Таблица II. 22

	1	2	3	4
x	2	2	—	—
y	3	3	4	4

Таблица II. 23

	1	2	3	4
x	v	v	—	—
y	v	v	u	u

Пример 3. Найти автомат Мили, представляющий событие $R=x\vee\{y\}\,y$ при условии, что область запрета задана регулярным выражением $S=\{xy\}$.

Решение. Размечая комплекс в соответствии с правилами 2, 2в и 3, мы получим:

$$R=\begin{array}{|c|c|c|c|c|c|c|}\hline (&x\vee&\{&y&\}&y&)\\ 0&&&1&&&\\ \hline\end{array}\,,\qquad \begin{array}{|c|c|c|c|c|}\hline 0&0&\boxed{0}&\boxed{0}\\ &&\boxed{1}&\boxed{1}\\ \hline\end{array}$$

$$S=\begin{array}{|c|c|c|c|c|}\hline \{&x&y&\}\\ 0&&2&3\\ \hline\end{array}\,,\qquad \begin{array}{|c|c|}\hline \boxed{0}&0\\ \boxed{3}&3\\ \hline\end{array}$$

По правилам 4 и 5а построим таблицы переходов и выходов II.24 и II.25.

Таблица II. 24

	0	1	2	3	*
x	2	*	*	2	*
y	1	1	3	*	*

Таблица II. 25

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

Запрещенным в рассматриваемом случае будет только выходной сигнал S. Применим правило 6а. Заменяя черточкой сигнал S в таблице выходов и ставя черточку в соответствующем месте таблицы переходов, приходим к табл. II.26 и II.27.

Теперь, как легко видеть, состояние 3 сделалось недостижимым, и его можно, следовательно, исключить. Производя переобозначения $0\to 1$, $1\to 2$, $2\to 3$, $*\to 4$, $R\to u$, $(\ )\to v$, мы получаем окончательные таблицы переходов (II.28) и выходов (II.29) искомого автомата Мили D.

Таблица II. 26

	0	1	2	3	*
x	2	*	*	2	*
y	1	1	—	*	*

Таблица II. 27

	0	1	2	3	*
x	R	( )	( )	( )	( )
y	R	R	—	( )	( )

Таблица II. 28

	1	2	3	4
x	3	4	4	4
y	2	2	—	4

Таблица II. 29

	1	2	3	4
x	u	v	v	v
y	u	u	—	v

§ 8. Синтез автоматов по индуцируемым ими отображениям

Одной из наиболее важных задач теории абстрактных автоматов является задача синтеза конечных автоматов по реализуемым ими алфавитным отображениям. Используя полученные в предыдущих параграфах результаты, нетрудно указать общий метод решения этой задачи.

8. 1. Первый этап решения указанной задачи состоит в том, что исходное (вообще говоря, частичное) алфавитное отображение φ записывается в виде таблицы соответствия [1]) и приводится к автоматному виду с помощью применения операции выравнивания длин слов (см. § 2). При этом обычно не прибегают сразу к стандартной операции выравнивания длин слов, а производят ряд последовательных попыток приведения отображения к автоматному виду путем приписывания по возможности меньшего числа пустых букв, прибегая после каждой попытки к про-

[1]) Общие методы конструктивного задания бесконечных таблиц соответствия в книге специально не разбираются. Простой пример такого задания рассмотрен в конце этого параграфа.

верке выполнимости условий автоматности (см. § 2). Результатом первого этапа является сокращенная таблица соответствия автоматного отображения ψ, в которое превратилось исходное отображение φ в результате применения операции выравнивания длин слов.

Второй этап решения состоит в нахождении канонического множества событий, соответствующего отображению ψ (см. § 2). Число событий канонического множества равно, как известно, числу букв выходного алфавита $\mathfrak{B}=(y_1, \ldots, y_m)$ отображения ψ. Событие $S(y_i)$ этого множества, соответствующее выходной букве y_i, строится следующим образом. Просматривая сокращенную таблицу соответствия отображения ψ, выделяют все начальные отрезки выходных слов, кончающиеся буквой y_i. Начальные отрезки входных слов отображения, соответствующие выделенным отрезкам, и составят событие $S(y_i)$ $(i=1, \ldots, m)$.

Результатом второго этапа является каноническое множество событий $S(y_i)$, ..., $S(y_m)$ отображения ψ. Попутно на этом этапе производится контроль правильности выполнения операций первого этапа: легко видеть (см. § 2), что *отображение ψ тогда и только тогда является автоматным, когда найденные события попарно не пересекаются.*

Третий этап решения состоит в нахождении возможно более простых регулярных выражений для найденных на предыдущем этапе событий $S(y_1)$, ..., $S(y_m)$. При этом пользуются тождественными преобразованиями в алгебре событий, а также следующим приемом расширения событий: *при подыскании для события $S(y_i)$ наиболее простого регулярного выражения можно добавлять к нему произвольное множество запрещенных слов, т. е. таких слов, которые не содержатся ни в одном из исходных событий $S(y_1)$, ..., $S(y_m)$.* Для упрощения последующих рассмотрений будем обозначать найденное таким образом регулярное выражение для события $S(y_i)$ просто через y_i $(i=1, \ldots, m)$. Заметим, что благодаря применению операции расширения событий, события, представляемые регулярными выражениями y_1, ..., y_m, могут пересекаться между собой.

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

Четвертый этап решения состоит в синтезе конечного автомата Мили или Мура, представляющего события $R_1, \ldots, R_m$, задаваемые регулярными выражениями $y_1, \ldots, y_m$, посредством множеств своих выходных сигналов.

Процесс синтеза может производиться по двум основным вариантам.

Первый вариант (с естественной областью запрета). В процессе синтеза считаются запрещенными, во-первых, все те слова, у которых хотя бы один непустой начальный отрезок (включая само слово) не содержится ни в одном из событий $R_1, \ldots, R_m$, и, во-вторых, все те слова, у которых хотя бы один непустой начальный отрезок (включая само слово) содержится одновременно в нескольких (более, чем в одном) событиях $R_1, \ldots, R_m$. Легко видеть, что этот способ задания области запрета совпадает со вторым из способов, рассмотренных в предыдущем параграфе. Производится синтез конечного частичного автомата Мура или Мили, представляющего события $R_1, \ldots, R_m$ с учетом введенной области запрета. При этом можно применять любой из описанных в двух предыдущих параграфах алгоритмов синтеза. Выходные сигналы (после исключения неопределенных выходных сигналов) в построенных автоматах будут совпадать с буквами $y_1, \ldots, y_m$ выходного алфавита отображения ψ, а сами автоматы будут индуцировать частичное отображение ψ_1, продолжающее частичное отображение ψ с сохранением входного и выходного алфавитов отображения ψ.

Второй вариант (с исключением одного из событий). Одно из заданных регулярных выражений (обычно наиболее сложное) исключается из заданного множества $y_1, \ldots, y_m$ регулярных выражений. Пусть этим выражением будет y_1. К оставшимся выражениям $y_2, \ldots, y_m$ применяется любой из описанных в § 6 и § 7 алгоритмов синтеза частичных автоматов Мили или Мура. При этом запрещенными считаются все те слова, у которых хотя бы один непустой начальный отрезок (включая само слово) содержится одновременно в нескольких (более, чем в одном) событиях $R_2, \ldots, R_m$. Выходные сигналы, содержащие более одного символа $y_2, \ldots, y_m$, будут при этом неопределенными. Из остающихся выход,

ных сигналов $y_2, \ldots, y_m$, ( ) пустой выходной сигнал ( ) заменяется выходным сигналом y_1, после чего синтезированные автоматы будут индуцировать частичное отображение ψ_2, продолжающее исходное частичное отображение ψ с сохранением его входного и выходного алфавита.

Обоснование описанного метода решения задачи синтеза автомата по индуцируемому им отображению вытекает из результатов второго параграфа настоящей главы и из сущности описанных в предыдущих параграфах алгоритмов синтеза автоматов по представляемым ими событиям. Некоторых пояснений заслуживает лишь способ образования областей запрета.

Из способа построения операции расширения событий на третьем этапе решения ясно, что любые нетривиальные пересечения расширенных событий $R_1, \ldots, R_m$, а также дополнение их объединения состоят из запрещенных слов. Из условия автоматности множества событий $S(y_1), \ldots, S(y_m)$ вытекает, что всякое слово, хотя бы один непустой начальный отрезок которого запрещен, само будет запрещенным. Теперь видно, что образование естественной области запрета в первом варианте синтеза не нарушает индуцируемого синтезируемым автоматом отображения на множестве допустимых слов отображения ψ. Расширение же области допустимых слов в автомате по сравнению с областью допустимых слов в исходном отображении ψ может произойти за счет запрещенных слов, добавленных в процессе расширения событий $S(y_1), \ldots, S(y_m)$ только к одному из этих событий.

Из способа образования расширений событий на третьем этапе решения непосредственно вытекает, что событие $S(y_1)$ содержится в дополнении объединения расширенных событий $R_1, \ldots, R_m$. Таким образом, во вполне определенном автомате A, представляющем события $R_2, \ldots, R_m$, событие $S(y_1)$, быть может расширенное за счет некоторого множества запрещенных слов, представляется выходным сигналом, состоящим из пустого множества выражений $y_2, \ldots, y_m$. Заменяя этот пустой выходной сигнал буквой y_1, мы придем, очевидно, к автомату, индуцирующему отображение, совпадающее с отображением ψ на множестве допустимых слов отображения ψ. Точно таким же образом, как и в первом варианте, нетрудно показать, что образо-

вание области запрета, описанной во втором варианте синтеза, не нарушает этого совпадения (способ задания области запрета совпадает в этом случае с третьим способом § 7, см. стр. 121).

Тем самым предложенный метод решения задачи синтеза автомата по индуцируемому им отображению полностью обоснован. Попутно доказано следующее предложение.

Если в описанном методе пользоваться первым вариантом синтеза и не прибегать к расширению исходных событий $S(y_1), \ldots, S(y_m)$, *то области определения исходного отображения и отображения, индуцируемого синтезированным автоматом, совпадают при условии, что возможности введения неопределенности в функции переходов и выходов, предоставляемые правилами* 5б *и* 6а *предыдущего параграфа, использованы полностью.*

Заметим, однако, что разумное использование возможностей, заключающихся в операции расширения событий (с целью получения наипростейших регулярных выражений), существенно упрощает процесс синтеза и особенно последующую минимизацию автоматов. Это обстоятельство обусловливает большие преимущества описанного метода синтеза даже в том случае, когда существуют более прямые методы синтеза, то есть в случае конечности области определения исходного автоматного отображения φ (см. § 2).

8.2. Сделаем еще одно замечание о дальнейших возможностях упрощения синтезируемого автомата в соответствии со вторым вариантом синтеза. В ряде случаев при использовании этого варианта можно относительно просто произвести дифференциацию пустых выходных сигналов, отождествляя только некоторую часть из них в таблице выходов автомата (обычной или сдвинутой) с выходным сигналом y_1 и производя замену остальных таких сигналов черточками. Так бывает, например, в том случае, когда все слова исключаемого события $S(y_1)$ совпадают с начальными отрезками слов остающихся событий $S(y_2), \ldots, S(y_m)$. Ясно, что в этом случае пустые выходные сигналы, относящиеся к пустому состоянию автомата, не будут представлять слов события $S(y_1)$, и их можно поэтому считать неопределенными. Неопределенным будет в этом случае и пустое состояние.

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

Рассмотрим теперь примеры синтеза автоматов по индуцируемым ими отображениям.

Пример 1. Найти конечный автомат Мили, индуцирующий отображение φ, заданное сокращенной таблицей соответствия:

$$yx \longrightarrow uv$$
$$\underbrace{xx\ldots x}_{n\text{ раз}} \longrightarrow v\,\underbrace{uu\ldots u}_{(n-1)\text{ раз}} \qquad (n=1,2,\ldots)$$

Решение. Легко проверить, что заданное отображение — автоматное. Поэтому можно начать решение сразу со второго этапа. Каноническое множество событий отображения φ состоит из двух событий:

$$S(u)=y\vee\{x\}xx \quad \text{и} \quad S(v)=x\vee yx.$$

Приведенные регулярные выражения являются достаточно простыми, в силу чего их минимизацию (третий этап решения) можно опустить. На четвертом этапе применим первый вариант синтеза.

Разметка выражений (в соответствии с правилами 2, 2б, 2в) дает:

$$u=\underset{0}{|}(|y\vee|\{|\underset{1}{x}|\}|\underset{2}{x}|x), \qquad v=\underset{0}{|}(|x\vee|\underset{4}{y}|x).$$

$$\boxed{0}\ 0\ \boxed{\begin{matrix}0\\1\end{matrix}}\ \boxed{\begin{matrix}0\\1\end{matrix}} \qquad \boxed{0}\ \boxed{0}$$

По правилам 4 и 5а находим таблицы переходов и выходов II.30 и II.31.

Таблица II 30

	0	1 ∨ 2	4	*
x	1 ∨ 2	1 ∨ 2	*	*
y	4	*	*	*

Таблица II.31

	0	1 ∨ 2	4	*
x	v	u	v	()
y	u	()	()	()

Учитывая, что область запрета задается вторым из способов, описанных в предыдущем параграфе, находим, что неопределенным выходным сигналом является пустой сигнал, а неопределенным состоянием — пустое состояние. Применяя правило 6а и вводя переобозначения $0 \to 1$, $1 \vee 2 \to 2$, $4 \to 3$, получаем таблицы переходов и выходов искомого (частичного) автомата Мили II.32 и II.33.

Таблица II.32

	1	2	3
x	2	2	—
y	3	—	—

Таблица II.33

	1	2	3
x	v	u	v
y	u	—	—

Применяя же второй вариант синтеза к выражению v (исключая u), мы получаем таблицу переходов II.34 и таблицу выходов II.35.

Таблица II.34

	0	4	*
x	*	*	*
y	4	*	*

Таблица II.35

	0	4	*
x	v	v	( )
y	( )	( )	( )

Неопределенных выходных сигналов и неопределенных состояний здесь нет. Вводя переобозначения $0 \to 1$, $* \to 2$, $4 \to 3$, мы получаем таблицы переходов и выходов искомого автомата Мили (табл. II.36 и II.37).

Таблица II.36

	1	2	3
x	2	2	2
y	3	2	2

Таблица II.37

	1	2	3
x	v	u	v
y	u	u	u

Нетрудно видеть, что автомат B получается из автомата A в результате ликвидации неопределенностей в его таблицах переходов и выходов.

Пример 2. Найти конечный автомат Мура, осуществляющий возведение в квадрат двузначных двоичных чисел. Числа подаются на вход автомата последовательно, разряд за разрядом, младшими разрядами вперед. Так же осуществляется выдача результата.

Решение. Поскольку результат возведения в квадрат двузначного числа может быть четырехзначным, то в первоначально заданной таблице соответствия входные и выходные слова имеют различную длину. Используя прием выравнивания длин слов, добавим к входным словам справа по две пустых буквы. В качестве пустой буквы можно в рассматриваемом случае выбрать цифру 0, поскольку добавление нуля в старшие разряды входных слов позволяет однозначно восстановить их первоначальный вид. После этого добавления мы получаем автоматное отображение φ, заданное следующей сокращенной таблицей соответствия:

$$0000 \longrightarrow 0000$$
$$1000 \longrightarrow 1000$$
$$0100 \longrightarrow 0010.$$
$$1100 \longrightarrow 1001.$$

Каноническое множество событий для отображения φ имеет вид

$$S(0) = 0 \vee 00 \vee 000 \vee 0000 \vee 10 \vee 100 \vee 1000 \vee 01 \vee 0100 \vee 11 \vee 110,$$

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

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

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

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

$$\begin{array}{l} 1=|(|1|\vee|0|1|0|\vee|1|1|0|0|)|. \\ \quad 0\ |\ 1\ |\ 2\ 3\ 6\ |\ 1\ 4\ 5\ 6\ | \\ \hline \quad \boxed{0}\ \ \boxed{0}\ \ \ \boxed{0}\ \ \ \ 1 \\ \qquad\qquad\qquad\qquad\ \ 6 \end{array}$$

По полученному выражению находим отмеченную таблицу переходов автомата Мура A (табл. II.38).

Таблица II.38

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

В соответствии с правилами второго варианта синтеза в этой таблице следует заменить пустой выходной сигнал ( ) выходным сигналом 0. Производя затем переобозначения пустого состояния ($* \to 7$), мы приходим к окончательному виду отмеченной таблицы переходов искомого автомата Мура (табл. II.39).

Таблица II.39

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

Пример 3. Построить автомат Мили, который преобразовывал бы подаваемые на его вход (старшими

разрядами вперед) числа от единицы до девяти, записанные в четверичной системе счисления, в приближенные (целочисленные) значения квадратного корня из них, выдаваемые на вход также в четверичной системе счисления (старшими разрядами вперед).

Р е ш е н и е. Приближенные значения квадратного корня из чисел от единицы до девяти будут равны соответственно 1, 1, 2, 2, 2, 2, 3, 3, 3. Будучи записаны в четверичной системе счисления, они приводят к следующей таблице соответствия отображения φ, которое требуется реализовать:

$$\begin{array}{ll} 01 \to 01 & 12 \to 02 \\ 02 \to 01 & 13 \to 03 \\ 03 \to 02 & 20 \to 03 \\ 10 \to 02 & 21 \to 03. \\ 11 \to 02 & \end{array}$$

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

Находим соответствующее ему каноническое множество событий:

$$\begin{aligned} S(0) &= 0 \vee 1 \vee 2, \\ S(1) &= 01 \vee 02, \\ S(2) &= 03 \vee 10 \vee 11 \vee 12, \\ S(3) &= 13 \vee 20 \vee 21. \end{aligned}$$

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

$$\begin{aligned} 1 &= 0(1 \vee 2), \\ 2 &= (03 \vee 10 \vee 11 \vee 12), \\ 3 &= (13 \vee 20 \vee 21). \end{aligned}$$

Разметка этих выражений по правилам 2, 2б, 2в, 3 приводит к следующему размеченному комплексу:

$$1 = \underset{\boxed{0}}{0}\,(\,\underset{\boxed{1}}{1}\ \vee\ \underset{\boxed{1}}{2}\,),\qquad 2 = (\,\underset{\boxed{0}}{0}\ 3 \vee \underset{\boxed{0}}{1}\ 0 \vee \underset{\boxed{0}}{1}\ 1 \vee \underset{\boxed{0}}{1}\ 2),$$

$$3 = (\,\underset{\boxed{0}}{1}\ 3 \vee \underset{\boxed{0}}{2}\ 0 \vee \underset{\boxed{0}}{2}\ 1).$$

По правилам 4 и 5а находим таблицы переходов и выходов автомата Мили II.40 и II.41.

Таблица II.40

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

Таблица II.41

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

В соответствии с правилами второго варианта синтеза пустой выходной сигнал ( ) должен быть заменен здесь выходным сигналом 0. Однако нетрудно видеть, что мы находимся в условиях применимости замечания 8.2. Поэтому мы можем считать неопределенными пустое состояние автомата и соответствующие ему выходные сигналы. По правилу 6а мы приходим к таблицам переходов и выходов искомого автомата II.42 и II.43.

Таблица II.42

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

Таблица II.43

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

Нетрудно заметить, что, не нарушая индуцируемого автоматом A отображения на множестве допустимых слов исходного отображения φ, можно заменить черточками все вхождения нулевого выходного сигнала во все столбцы таблицы выходов, отличные от начального столбца.

§ 9. Минимизация абстрактных автоматов

Основной задачей настоящего параграфа является разработка таких конструктивных приемов, которые позволяли бы по любому данному конечному автомату Мили или Мура A находить *эквивалентный* ему (или, в случае частичных автоматов, *эквивалентно продолжающий* его) автомат Мили или соответственно автомат Мура, которые имели бы наименьшее возможное число состояний. Эта задача носит название *задачи минимизации конечных автоматов*. Напомним, что вполне определенные автоматы называются эквивалентными, если они индуцируют одно и то же отображение, а частичный автомат A называется эквивалентным продолжением частичного автомата B, если индуцируемое им отображение совпадает с отображением, индуцируемым автоматом B на всех допустимых для автомата B словах.

Частично задача минимизации конечных автоматов уже решалась в предыдущих параграфах. Так, в § 1 был описан метод исключения недостижимых состояний. В параграфах, посвященных синтезу конечных автоматов, много внимания уделялось возможности получения автоматов с возможно меньшим числом состояний. Следует подчеркнуть, что разработанные выше методы синтеза выгодно отличаются от других возможных методов (например, от прямого метода синтеза, описанного в теореме 2.1 из § 2) как раз тем, что они дают возможность в большинстве случаев получать автоматы, достаточно близкие к минимальным.

Однако существуют и такие случаи, когда автоматы, синтезированные в соответствии с описанными выше алгоритмами, будут заметно отличаться от минимальных. Необходимо поэтому разработать методы, позволяющие минимизировать уже построенные абстрактные автоматы

с целью получения автоматов с минимальным числом состояний.

Первым (предварительным) этапом всякой минимизации является выделение неопределенных выходных сигналов и состояний и внесение соответствующей неопределенности в таблицы переходов и выходов автомата. Внесение неопределенности не должно изменять исходного отображения, которое должен индуцировать рассматриваемый автомат. Вместе с тем желательно, чтобы оно было, по возможности, полным, т. е. не оставляло бы определенным ни одного из тех мест таблицы переходов и выходов автомата, которые могут быть сделаны неопределенными. Степень полноты внесенной неопределенности определяет в значительной мере и возможности последующей минимизации.

Вторым этапом минимизации является исключение недостижимых состояний, то есть переход к связному автомату (см. § 1). Возможность уменьшения числа состояний на этом этапе, вообще говоря, увеличивается при увеличении степени неопределенности, внесенной в автомат на первом этапе.

Третьим этапом, заключающим процесс минимизации, является этап объединения в одно состояние множеств так называемых *совместимых состояний*. К подробному рассмотрению этого этапа мы сейчас и перейдем.

Прежде всего необходимо ввести ряд новых определений. Пусть A — произвольный абстрактный автомат, a — любое его состояние. Будем говорить, что слово p во входном алфавите автомата A *применимо* к состоянию a, если, подавая это слово на вход автомата A, установленного предварительно в состояние a, мы получим на его выходе вполне определенное слово q в выходном алфавите, имеющее ту же самую длину, что и слово p. Иначе говоря, подавая последовательно на вход автомата буквы слова p, мы ни разу не должны столкнуться со случаем неопределенности соответствующего очередной подаваемой букве выходного сигнала. Слово q будет называться *результатом применения слова p к состоянию a*.

В случае, когда слово p неприменимо к состоянию a, мы будем говорить, что результат применения слова p

к состоянию a является *неопределенным* (при этом безразлично, дают ли некоторые непустые начальные отрезки слова p в применении к состоянию a о п р е д е л е н н ы е результаты или нет).

Используя понятие о неопределенном результате применения слова p к состоянию автомата, мы можем не заботиться о том, чтобы подаваемые на вход автомата слова были обязательно словами во входном алфавите автомата. Мы можем подавать на вход автомата, установленного в то или иное начальное состояние a, слова в любом алфавите, считая, что результат применения слова p к состоянию a не определен всякий раз, когда слово p содержит буквы, не включенные во входной алфавит автомата.

Теперь можно дать определение *совместимых состояний*.

9.1. *Состояния $a_{i_1}, \ldots, a_{i_n}$, входящие в один и тот же или в несколько различных автоматов Мили, называются совместимыми, если все определенные (не считая неопределенных результатов) результаты применения любого слова p к состояниям $a_{i_1}, \ldots, a_{i_n}$ будут одними и теми же (зависящими только от слова p, но не от выбора состояния a_{i_k} из данного множества состояний). В случае автоматов Мура кроме этого условия для совместимости данных состояний требуется еще, чтобы, не считая неопределенных отметок, все совместимые состояния имели бы одинаковые отметки. Совместимые состояния во вполне определенных автоматах называются также эквивалентными.*

Для случая конечных автоматов существует эффективный конструктивный прием для нахождения совместимых между собой состояний в любом конечном множестве автоматов одного и того же типа (Мура или Мили). Этот прием основан на использовании понятия *i-совместимости состояний*.

Состояния $a_{i_1}, \ldots, a_{i_n}$ а в т о м а т о в М и л и называются *i-совместимыми* для любого данного $i=1,2,\ldots$, если, с точностью до неопределенных результатов, результат применения любого слова длины i к состояниям $a_{i_1}, \ldots, a_{i_n}$ будет одним и тем же, находясь в зависимости лишь от выбора слова, но не от выбора состояния. Состояния а в т о м а т о в М у р а называются *0-совместимыми*, если, не считая неопределенных отметок, они одинаково отмечены;

они называются *i-совместимыми* для любого $i=1, 2, \ldots$, если они 0-совместимы и если, с точностью до неопределенных результатов, результат применения любого данного слова длины i ко всем рассматриваемым состояниям одинаков.

Легко видеть, что справедливо следующее предложение.

9.2. *i-совместимые состояния будут также и j-совместимыми для любого $j<i$. Состояния тогда и только тогда совместимы, когда они i-совместимы для всех $i=1, 2, \ldots$*

Введем еще следующее важное определение.

9.3. *Для любого множества M автоматов одного и того же типа (Мура или Мили) и любого $i=0, 1, 2, \ldots$ i-классом данного множества M называется всякое максимальное множество i-совместимых между собой состояний автоматов из M, т. е. такое множество, к которому нельзя добавить ни одного нового состояния без нарушения свойства i-совместимости. Всякое максимальное множество совместимых между собой состояний автоматов из M будет называться ∞-классом или классом совместимости множества M.*

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

Уже на этом примере видно, что для частичных автоматов i-классы, вообще говоря, пересекаются между собой. То же самое имеет место и для ∞-классов. В то же самое время нетрудно показать, что для вполне определенных автоматов i-классы не могут пересекаться между собой, ибо отношение i-совместимости состояний обладает в этом случае свойством транзитивности (если состояние a_1 i-совместимо с состоянием a_2, а состояние a_2 i-совместимо с состоянием a_3, то состояние a_1 i-совместимо с состоянием

a_s). Для случая же частичных автоматов свойство транзитивности для отношения i-совместимости, вообще говоря, не выполняется.

Определим теперь так называемую *операцию расщепления i-классов*. Пусть M — множество автоматов одного и того же типа (Мура или Мили), $(x_1, x_2, \ldots, x_n)$ — множество всех букв их входных алфавитов, $K_1(i), \ldots, K_p(i)$ — совокупность всех i-классов рассматриваемого множества. Будем говорить, что множество N состояний $a_{j_1}, \ldots, a_{j_k}$, целиком содержащееся в одном из i-классов $K_r(i)$, *выдерживает* умножение на входную букву x_m, если все состояния $a_{j_1}x_m, \ldots, a_{j_k}x_m$ (не считая тех, которые не определены) содержатся в одном и том же i-классе $K_t(i)$, зависящем от выбора N и x_m.

9.4. *Операция расщепления i-классов состоит в нахождении максимальных подмножеств состояний каждого i-класса, выдерживающих умножение на все буквы $x_1, \ldots, x_n$ входных алфавитов рассматриваемого множества автоматов.*

Операция расщепления i-классов выполняется очевидным образом с помощью использования таблиц переходов рассматриваемого множества автоматов. Справедливо следующее важное предложение.

9.5. *В результате применения операции расщепления к i-классам произвольного множества M автоматов одного и того же типа (Мура или Мили) возникают все $(i+1)$-классы этого множества ($i=0, 1, \ldots$): ими являются все максимальные множества, возникшие в результате расщепления.*

В самом деле, пусть N — произвольное множество, возникшее в результате применения операции расщепления к одному из i-классов $K_r(i)$ данного множества автоматов M, не входящее ни в какое другое множество того же вида, а a_m и a_n — два произвольных состояния из N. Рассмотрим произвольное слово $p_1 = xp$, имеющее длину $i+1$, и будем считать сначала, что рассматриваемое множество M состоит из автоматов Мили. В таком случае, в силу предложения 9.2, состояния a_m и a_n будут 1-совместимыми. Следовательно, выходные сигналы, определяемые парами (a_m, x) и (a_n, x), одинаковы в случае, если оба они определены. С другой стороны, по определению

множества N, состояния $a_m x$ и $a_n x$, если они оба определены, входят в один и тот же i-класс, так что применение к ним слова p дает одинаковые результаты. Теперь ясно, что, в случае, когда оба результата применения слова p_1 к состояниям a_m и a_n не будут неопределенными, они обязательно совпадут. Ввиду произвольности выбора слова p_1 это означает, что состояния a_m и a_n $(i+1)$-совместимы.

Предположим теперь, что найдется состояние a_k, которое $(i+1)$-совместимо со всеми состояниями множества N. В силу предложения 9.2 оно будет также i-совместимо с ними и, следовательно, по определению i-классов, найдется i-класс $K_t(i)$, содержащий все состояния множества N и состояние a_k. Пусть a_j — произвольное состояние, содержащееся в N. Ввиду $(i+1)$-совместимости состояний a_j и a_k для любой входной буквы x состояния $a_j x$ и $a_k x$,— в случае, если они оба определены,— будут, очевидно, i-совместимы. Это, в свою очередь, означает, что в результате применения операции расщепления к i-классу $K_t(i)$ возникнет множество N_1, содержащее все состояния множества N и состояние a_k. Но включение $N \subset N_1$ противоречит определению множества N. Следовательно, к множеству N нельзя добавить ни одного нового состояния, $(i+1)$-эквивалентного со всеми состояниями множества N. Тем самым доказано, что множество N является $(i+1)$-классом.

Пусть теперь $K_p(i+1)$ — произвольный $(i+1)$-класс множества автоматов M. В силу теоремы 9.2 он содержится в некотором i-классе $K_s(i)$ того же множества. Из определения $(i+1)$-класса следует, что для любой входной буквы x все определенные состояния вида ax (где a — любое состояние из $K_p(i+1)$), являются i-совместимыми и входят, следовательно, в один и тот же i-класс. Таким образом, множество $K_p(i+1)$ выдерживает умножение на все входные буквы. Обозначим через P одно из максимальных подмножеств i-класса $K_s(i)$, выдерживающих умножение на все входные буквы и содержащее класс $K_p(i+1)$. Используя первую часть данного доказательства, получим, что все состояния множества P $(i+1)$-совместимы. В силу определения $(i+1)$-класса, это означает, что $P = K_p(i+1)$. Таким образом, $(i+1)$-класс $K_p(i+1)$ возникает в ре

зультате применения операции расщепления к i-классам множества M.

Ясно, что $K_p(i+1)$ является максимальным среди множеств, возникающих в результате расщепления i-классов, поскольку все такие множества будут $(i+1)$-классами (как было только что показано), а $(i+1)$-класс $K_p(i+1)$ не может, по определению, содержаться ни в каком бо́льшем $(i+1)$-классе.

Тем самым для случая автоматов Мили предложение 9.5 полностью доказано.

Доказательство для случая автоматов Мура почти дословно повторяет уже проведенное доказательство. Некоторые изменения должны быть сделаны лишь в самом начале построения. Рассмотренные выше состояния $a_m x$ и $a_n x$ — в случае, если они оба определены,— попадают, в силу определения множества N, в один и тот же i-класс. Поскольку всякий i-класс состоит из 0-совместимых состояний, то выходные сигналы, выдаваемые при переходах $a_m \to a_m x$ и $a_n \to a_n x$, — в случае, если они оба определены,— оказываются непременно одинаковыми. Далее доказательство ничем не отличается от соответствующего доказательства для случая автоматов Мили. Тем самым теорема 9.5 полностью доказана.

Если теперь исходное множество M состоит из конечного числа конечных автоматов (Мили или Мура), то применение операции расщепления i-классов оказывается возможным лишь конечное число раз. В самом деле, исходные 1-классы или 0-классы представляют собою в этом случае конечные множества, так что неограниченное их измельчение невозможно: через конечное число шагов $(i+1)$-классы совпадут с i-классами. Новое применение операции расщепления к $(i+1)$-классам будет повторять применение этой операции к i-классам. Поэтому возникающие в результате ее применения $(i+2)$-классы совпадут с i-классами. Продолжая этот процесс, получим, что для любого $j>i$ j-классы совпадают с i-классами. Таким образом, i-классы будут в этом случае не чем иным, как ∞-классами.

В результате нами доказано следующее предложение.

9.6. *Если применять последовательно операцию расщепления i-классов к конечному множеству M конечных*

автоматов Мили или Мура, отправляясь от 1*-классов (для случая автоматов Мили) или от* 0*-классов (для случая автоматов Мура), то через конечное число шагов для некоторого* $k \geqslant 0$ *процесс расщепления* k*-классов даст в результате те же самые* k*-классы. Удовлетворяющие этому условию (нерасщепляемые далее)* k*-классы будут совпадать с* ∞*-классами исходного множества* M.

Теорема 9.6 усиливает результаты, полученные А у ф е н к а м п о м и Х о н о м [1]). Она является основой, на которой зиждется минимизация конечных автоматов.

Значение теоремы 9.6 станет ясным после того, как мы покажем, каким образом, зная ∞-классы, можно осуществлять минимизацию автоматов.

Обозначим через $K_1, K_2, \ldots, K_n \ldots$ ∞-классы какого-либо множества M автоматов одного и того же типа (не обязательно конечных), а через $(x_1, \ldots, x_n)$ — все буквы входных алфавитов автоматов множества M. Так как ∞-классы являются вместе с тем и j-классами для всех $j=0, 1, 2, \ldots$, то для каждой входной буквы x_k все состояния, входящие в любой ∞-класс K_m, порождают один и тот же выходной сигнал (в случае автоматов Мили) или отмечены одним и тем же выходным сигналом (в случае автоматов Мура), либо соответствующие выходные сигналы не определены. Построим таблицу (функцию) выходов (обычную или сдвинутую) некоторого автомата A, состояниями которого служат ∞-классы $K_1, K_2, \ldots, K_n$, а входными сигналами — буквы $x_1, \ldots, x_n$. В случае автоматов Мили относим каждой паре (K_m, x_k) в ы х о д н о й с и г н а л, с о о т в е т с т в у ю щ и й п а р е (a_m, x_k) д л я л ю б о г о a_m из K_m, д л я к о т о р о г о э т о т с и г н а л о п р е д е л е н. Если же для всех пар (a_m, x_k) соответствующие им выходные сигналы н е о п р е д е л е н ы, то мы считаем, что в ы х о д н о й с и г н а л д л я п а р ы (K_m, x_k) т а к ж е н е о п р е д е л е н.

В случае автоматов Мура мы отмечаем каждый класс K_m в ы х о д н ы м с и г н а л о м, к о т о р ы м о т м е ч е н п р о и з в о л ь н ы й э л е м е н т $a_m \in K_m$. Если же

[1]) D. D. A u f e n k a m p and F. E. H o h n, Analysis of sequential machines. IRE Trans. EC-6, № 4, 1957, p. 276—285 (русский перевод в сб. переводов «Математика», ИЛ, 1959, № 3 : 3, стр. 129—146).

все элементы, входящие в K_m, не отмечены, то мы будем считать отметку класса K_m неопределенной.

Таблицу (функцию) переходов автомата A мы построим по следующему правилу: переход $K_m \to K_m x_k$ будет считаться неопределенным, если для всех состояний a_m, составляющих класс K_m, переходы $a_m \to a_m x_k$ не определены. Если же хотя бы для одного состояния $a_m \in K_m$ переход $a_m \to a_m x_k$ определен, то переход $K_m \to K_m x_k$ также будет считаться определенным, а в качестве состояния $K_m x_k$ будет приниматься любой из ∞-классов K_r (их может быть несколько), содержащий все определенные состояния вида $a_m x_k$ ($a_m \in K_m$). Из предложений 9.5 и 9.6 вытекает, что ∞-классы K_r с требуемыми свойствами обязательно существуют.

Если существуют такие ∞-классы, каждый из которых содержит начальные состояния всех автоматов множества M, то один из таких классов принимается в качестве начального состояния построенного автомата A. В противном случае мы либо вовсе не будем фиксировать начального состояния, либо будем принимать в качестве начальных все ∞-классы, которые содержат начальное состояние хотя бы одного из автоматов множества M.

Описанный прием построения автомата A по данному множеству M автоматов будет называться нормализацией, а все построенные таким образом автоматы — нормальными формами множества автоматов M.

Нормальная форма множества M будет автоматом Мили или Мура соответственно тому, являлись ли автоматами Мили или Мура все автоматы множества M. Легко видеть также, что для случая вполне определенных автоматов нормальная форма определяется однозначно, поскольку, как было отмечено выше, ∞-классы в этом случае не пересекаются и, следовательно, как переходы $K_m \to K_m x_k$ в нормальной форме, так и ее начальное состояние определяются однозначным образом.

Отметим еще, что неопределенностью, возникающей в таблице переходов нормальной формы, обычно пользуются при последующем упрощении на уровне структурного синтеза автоматов. В частности, если переход $K_m \to K_m x_k$ согласно определению нормальной формы таков, что

в качестве $K_m x_k$ может быть выбран любой ∞-класс K_r, то при переходе к этапу структурного синтеза в соответствующем месте таблицы переходов можно поставить специальный значок, означающий, что хотя соответствующий переход и определен, но он может быть переходом в л ю б о е состояние (назовем такой переход б е з р а з л и ч н ы м). В качестве такого значка (во всяком случае на уровне абстрактного синтеза) нельзя, однако, употреблять черточку, поскольку мы условились использовать ее для указания того, что в местах таблицы переходов, отмеченных черточкой, переходы не определены вовсе. Проведение различия между переходами, которые определены, но результат их безразличен, и переходами, результат которых не определен вовсе, существенно для правильности понимания последующих предложений (9.7—9.8). При переходе же к структурному синтезу, где н е о п р е д е л е н н ы х переходов не остается совсем (а существуют лишь б е з р а з л и ч н ы е переходы), это различие утрачивает свое значение.

Разница между неопределенными и безразличными переходами сказывается всякий раз, когда оперируют с отображениями, индуцируемыми частичными автоматами: *принимая безразличные переходы за неопределенные, мы сужаем, вообще говоря, область определения этих отображений и тем самым можем нарушить правильность результатов такого типа, какие содержатся в последующих теоремах* 9.7 *и* 9.8.

Справедливо следующее предложение.

9.7. *Если в любой нормальной форме A произвольного множества M автоматов выбрать в качестве начального состояния ∞-класс, содержащий начальное состояние произвольного автомата A_i из множества M, то автомат A будет являться эквивалентным продолжением автомата A_i.*

Доказательство теоремы 9.7. вытекает непосредственно из определения нормальной формы. В самом деле, пусть $p = x_{i_1} x_{i_2} \ldots x_{i_k}$ — произвольное допустимое слово автомата A_i. Обозначим через a_1 начальное состояние автомата A_i, а через K_1 — содержащий его ∞-класс. Согласно определению нормальной формы, автомат A из состояния K_1 под действием входного сигнала x_{i_1} перейдет в состояние K_{l_1}, представляющее тот ∞-класс, который содержит состояние

$a_1x_{i_1}$ автомата A_i. При этом будет выдан тот же самый выходной сигнал, который дается автоматом A_i при переходе $a_1 \to a_1x_{i_1}$, — безразлично, имеем ли мы дело с автоматами Мили или Мура. В случае состояния $a_{j_1} = a_1x_{i_1}$ мы будем иметь дело снова с точно такой же ситуацией, как и в случае состояния a_1. Поэтому вторые буквы выходных слов, выданных автоматами A и A_i в ответ на слово p, будут снова одинаковыми. Продолжая подобным же образом, мы приходим к выводу, что результаты применения слова p к состояниям a_1 и K_1 совпадают. Ввиду произвольности выбора слова p теорема доказана.

Нормальная форма любого множества автоматов удовлетворяет следующему условию.

9. 8. *Во всякой нормальной форме A любого множества автоматов M нет ни одной пары совместимых между собой состояний.*

В самом деле, по определению ∞-классов (состояний автомата A) для любой пары (K_i, K_j) таких классов найдется состояние a_j, входящее в класс K_j и несовместимое с множеством состояний класса K_i. Следовательно, найдется такое входное слово p и такое состояние a_i в классе K_i, что результаты применения слова p к состояниям a_i и a_j определены и не совпадают между собой. Точно так же, как и при доказательстве предыдущей теоремы, в данном случае можно показать, что результаты применения слова p к состояниям K_i и K_j автомата A совпадают, соответственно, с результатами применения слова p к состояниям a_i и a_j и, следовательно, являются различными. Это означает несовместимость состояний K_i и K_j. В силу произвольности выбора состояний K_i и K_j теорема доказана.

Полученные результаты позволяют производить минимизацию с одновременным объединением в одном автомате произвольных конечных множеств автоматов. Мы будем, однако, в основном применять эти результаты для целей минимизации одного автомата. Если ограничиться к тому же лишь вполне определенными автоматами, то метод исключения недостижимых состояний (§ 1) вместе с операцией нормализации дает возможность осуществлять абсолютную минимизацию автоматов. Это обстоятельство вытекает из следующей теоремы.

9. 9. *Если в произвольном конечном вполне определенном автомате Мили A исключить недостижимые состояния и для полученного таким образом автомата B построить нормальную форму C, то автомат C будет эквивалентен автомату A и будет иметь наименьшее число состояний среди всех автоматов, эквивалентных автомату A.*

Доказательство сформулированной теоремы осуществляется следующим образом. Из предложения 1.9 вытекает, что автомат B эквивалентен автомату A. Ввиду же полной определенности автомата B его нормальная форма C, являясь его эквивалентным продолжением (предложение 9.7), окажется просто эквивалентной ему, а значит, и автомату A.

Пусть D — произвольный абстрактный автомат, эквивалентный автомату A. Поскольку всякий автомат второго рода (в частности, всякий автомат Мура) можно без изменения числа состояний интерпретировать как автомат Мили, мы будем предполагать, что автомат D является автоматом Мили. Будучи эквивалентным автомату A, он будет эквивалентен и автомату C. Эквивалентность же автоматов означает эквивалентность (в смысле определения 9.1) их начальных состояний.

Кроме того, поскольку в автомате B всякое состояние достижимо, то, как следует из способа построения нормальной формы, этим же свойством обладает и автомат C: состояние K автомата C (∞-класс автомата B) получается из его начального состояния K_0 с помощью любого входного слова, которое переводит автомат B из начального состояния в любое состояние, содержащееся в классе K.

Мы можем поэтому для каждого состояния K_i автомата C фиксировать входное слово p_i так, чтобы $K_i = K_0 p_i$ ($i = 1, \ldots, n$). Отправляясь от начального состояния d_0 автомата D, определим аналогичным образом ряд его состояний $d_i = d_0 p_i (i = 1, \ldots, n)$. Поскольку состояния K_0 и d_0 эквивалентны между собой (то есть, иначе говоря, применение к ним любого входного слова дает одинаковые результаты на выходе), то, как нетрудно видеть, для любого $i = 1, \ldots, n$ состояния K_i и d_i также будут попарно эквивалентными. Если бы для каких-то i и j состояния d_i и d_j оказались одинаковыми, то состояния K_i и K_j, будучи эквивалент-

ными одному и тому же состоянию, были бы эквивалентными между собой.

Но, ввиду предложения 9.8, в автомате C все состояния попарно неэквивалентны. Отсюда следует, что все состояния $d_0, d_1, \ldots, d_n$ в автомате D различны. Следовательно, автомат D имеет не меньше состояний, чем автомат C. В силу произвольности выбора автомата D теорема доказана.

Аналогичная теорема имеет место и для случая конечных автоматов Мура.

9. 10. *Если в произвольном конечном вполне определенном автомате Мура A исключить недостижимые состояния, а для полученного таким образом автомата Мура B построить нормальную форму C, то автомат C будет автоматом Мура, эквивалентным автомату A, имеющим сним одинаковые отметки начальных состояний, и будет иметь наименьшее число состояний среди всех автоматов Мура, эквивалентных автомату A и имеющих с ним одинаковые отметки начальных состояний.*

Доказательство этого предложения производится точно так же, как и доказательство предложения 9.9. Единственное добавление должно быть сделано во второй части доказательства — при установлении эквивалентности начальных состояний d_0 и K_0. Дело в том, что в случае автоматов Мура эквивалентность состояний требует не только совпадения результатов применения к этим состояниям любого входного слова, но и совпадения их отметок (см. 9.1).

Если состояния в эквивалентных автоматах могут быть получены из начальных состояний в результате действия непустого входного слова, то их отметки совпадают с последней буквой соответствующих выходных слов, одинаковых в силу эквивалентности автоматов. Поэтому эквивалентность состояний d_l и K_l, отличных от начальных состояний, для автоматов Мура устанавливается точно так же, как и для автоматов Мили. Что же касается начальных состояний, то совпадение их отметок, вообще говоря, требуется постулировать, что и было сделано нами при формулировке теоремы. Легко показать, что без этого теорема может потерять силу. После же такого постулирования доказательство проходит точно так же, как и

в предыдущем случае. Совпадение отметок начальных состояний у автоматов A и C вытекает из определения нормальной формы.

Теоремы 9.9 и 9.10 показывают, что в случае вполне определенных автоматов построение нормальной формы приводит к абсолютной минимизации (в случае автоматов Мура нужно лишь предварительно перепробовать различные отметки начального состояния, не меняющие отображения, индуцируемого автоматом). В случае же частичных автоматов дело обстоит гораздо сложнее: с построением ∞-классов и нормальной формы процесс минимизации, вообще говоря, не заканчивается. При современном состоянии разработки этого вопроса нахождение абсолютных минимизаций основано на использовании методов перебора вариантов. Поэтому на практике либо ограничиваются нахождением нормальной формы минимизируемого автомата, либо применяют перебор среди относительно небольшого числа вариантов, отказываясь от достижения абсолютной минимизации.

Так называемый *основной алгоритм минимизации* состоит в последовательном применении операции исключения недостижимых состояний, операции нормализации и, если потребуется, операции исключения недостижимых состояний в полученной нормальной форме (для вполне определенных автоматов такая потребность никогда не может появиться). При этом в случае минимизации множества M, состоящего из двух и более автоматов, достижимыми будут считаться все те состояния (∞-классы) построенной нормальной формы A, которые могут быть получены в результате применения какого-либо входного слова к любому из состояний нормальной формы (к ∞-классу), содержащему начальное состояние хотя бы одного из автоматов, входящих в M. Иногда удобно все такие ∞-классы называть *начальными состояниями* нормальной формы A.

В следующем параграфе мы рассмотрим дополнения к этому алгоритму, а теперь покажем его работу на примере.

Пример. Провести совместную минимизацию множества M, состоящего из двух автоматов Мили A и B,

заданных таблицами переходов и выходов II. 44, II. 45, II. 46, II. 47.

Таблица II 44

	1	2	3	4	5
x	2	—	5	3	—
y	3	4	—	—	—

Таблица II. 45

	1	2	3	4	5
x	u	—	u	v	u
y	v	u	—	—	—

Таблица II 46

	6	7	8	9
x	7	—	—	—
z	9	8	7	—

Таблица II 47

	6	7	8	9
x	u	—	—	—
z	w	w	w	w

Р е ш е н и е. Применяем основной алгоритм минимизации. Оба заданных автомата связаны. Следовательно, недостижимые состояния отсутствуют. Переходим поэтому к построению нормальной формы заданного множества автоматов. Имеется всего три 1-класса: $a=(1,3,5,6,7,8,9)$, $b=(2, 3, 5, 6, 7, 8, 9)$, $c=(2, 4, 7, 8, 9)$. Заметим, что автоматы A и B имеют различные входные алфавиты (как, впрочем, и выходные). Поэтому объединение в 1-классы производится по совпадению реакций (с точностью до неопределенных реакций) на три входных сигнала x, y, z. В класс a попадаются состояния, имеющие реакции $(uv—)$, $(u— —)$, $(u — w)$ и $(— —w)$, в класс b — состояния с реакциями $(—u—)$, $(u — —)$, $(u — w)$ и $(— —w)$, в класс c — состояния с реакциями $(—u—)$, $(v — —)$ и $(— — w)$.

Для определения, какие из 1-классов расщепляются, найдем множества, в которые они переводятся под воздействием входных сигналов x, y, z: $ax=(2, 5, 7) \subset b$, $ay=(3) \subset a$ (или $ay=(3) \subset b$), $az=(7,8,9)$ принадлежит любому 1-классу, $bx=(5, 7) \subset a$ (или $bx=(5,7) \subset b$), $by=(4) \subset c$. $bz=(7, 8, 9)$ принадлежит любому 1-классу, $cx=(3) \subset a$ (или

$cx=(3)\subset b)$, $cy=(4)\subset c$, $cz=(7, 8)$ принадлежит любому классу.

Таким образом, расщепления 1-классов не происходит, и их можно принять за ∞-классы и строить нормальные формы по найденным реакциям и включениям. Поскольку в трех местах нам предоставляется выбор из двух возможностей и еще в трех местах — выбор из трех возможностей, то существует всего $2^3\cdot 3^3=216$ нормальных форм. В каждой из них за начальное состояние можно принять класс a, содержащий начальные состояния обоих данных автоматов.

Таблицы переходов и выходов одной из нормальных форм имеют вид, представленный на табл. II. 48. и II. 49.

Таблица II. 48

	a	b	c
x	b	a	a
y	a	c	c
z	a	a	a

Таблица II. 49

	a	b	c
x	u	u	v
y	v	u	u
z	w	w	w

Автомат C, в соответствии с теоремой 9.7, индуцирует отображение, продолжающее отображения, индуцируемые исходными автоматами A и B.

§ 10. Некоторые дополнительные приемы минимизации

Описанный в предыдущем параграфе основной алгоритм минимизации автоматов основан на отождествлении всех совместимых между собой состояний. Часто оказывается удобным, однако, проводить отождествление лишь некоторой части совместимых состояний. В предыдущем параграфе все совместимые между собой состояния объединялись в один класс, называемый ∞-классом. Система всех ∞-классов осуществляла *покрытие* множества всех состояний заданных автоматов; иначе говоря, всякое состояние содержалось хотя бы в одном ∞-классе. Эта система обладала также тем свойством, что все состояния, состав

ляющие один и тот же ∞-класс, переводились любым входным сигналом x в один и тот же ∞-класс (зависящий от x).

Обобщим теперь понятие системы ∞-классов, введя следующее определение.

10.1. *Совокупность множеств $K_1, K_2, \ldots, K_m$ состояний заданного множества M автоматов одного и того же типа (Мура или Мили) называется системой инвариантных классов состояний автоматов множества M, если удовлетворяются следующие три условия:*

а) *любое состояние произвольного автомата из множества M содержится хотя бы в одном из классов $K_1, \ldots, K_m$,*

б) *все состояния, входящие в один и тот же класс K_i, являются 0-совместимыми (в случае автоматов Мура) или 1-совместимыми (в случае автоматов Мили),*

в) *любой входной сигнал x переводит (с точностью до неопределенных переходов) все состояния, входящие в один и тот же инвариантный класс K_i, в некоторый другой инвариантный класс K_j (зависящий от выбора класса K_i и входного сигнала x).*

Система инвариантных классов называется правильной, если составляющие ее классы попарно не пересекаются.

Ясно, что система ∞-классов представляет собою частный случай системы инвариантных классов. Нетрудно показать, что любой класс произвольной системы инвариантных классов состоит из совместимых между собой состояний и входит, следовательно, в какой-либо ∞-класс. В связи с этим ∞-классы естественно называть *максимальными инвариантными классами*.

Пусть $K_1, \ldots, K_m$ — произвольная система инвариантных классов состояний некоторого множества автоматов M. Будем для краткости называть ее просто *инвариантной системой*. Построим автомат A, состояниями которого будут все классы $K_1, \ldots, K_m$ инвариантной системы, а входной и выходной алфавиты будут совпадать с объединениями входных и, соответственно, выходных алфавитов всех автоматов множества M.

Функцию переходов автомата A определим следующим образом. Для любого класса K_i и любого входного сигнала x значение $\delta(K_i, x) = K_j$ функции переходов автомата A считается неопределенным, если

неопределенными были соответствующие значения $a_i x$ функций переходов автоматов множества M для всех состояний a_i, входящих в класс K_i. Если же для какого-либо состояния a_i из класса K_i переход $a_i \longrightarrow a_i x$ определен, то определенным считается также переход $K_i \longrightarrow K_i x$, причем в качестве значения для $K_i x$ выбирается один из тех классов K_j, который содержит все состояния вида $a_i x\,(a_i \in K_i)$.

Функцию выходов автомата A (обычную или сдвинутую) определим следующим образом. Для любого класса K_i и любого входного сигнала x значение $\lambda(K_i, x)$ функции выходов автомата A считается неопределенным, если неопределенными были соответствующие значения $\lambda_i(a_i, x)$ функций выходов автоматов множества M для всех состояний a_i, входящих в класс K_i. Если для какого-либо состояния a_i из класса K_i значение выхода $\lambda_i(a_i, x)$ определено, то значение выхода $\lambda(K_i, x)$ также считается определенным и равным $\lambda_i(a_i, x)$ (ввиду условия б) из определения 10.1 определенное таким образом значение $\lambda(K_i, x)$ не зависит от выбора состояния a_i из класса K_i).

Построенный таким образом автомат A (в котором не фиксируется пока начальное состояние) называется *приведенной формой* множества автоматов M по системе инвариантных классов $K_1, \ldots, K_m$, а сама операция получения автомата A — *операцией приведения*.

Описанная в предыдущем параграфе операция нормализации является частным случаем операции приведения, а нормальная форма множества автоматов M — частным случаем приведенной формы того же множества.

Точно так же, как была доказана теорема 9.7 предыдущего параграфа, может быть доказано следующее предложение.

10.2. *Если в любой приведенной форме A произвольного множества автоматов M одного и того же типа выбрать в качестве начального состояния один из инвариантных классов, содержащих начальное состояние произвольного автомата A_i из множества M, то автомат A будет являться эквивалентным продолжением автомата A_i.*

В дальнейшем изложении в настоящем параграфе мы ограничимся случаем, когда заданное множество автоматов M состоит из одного автомата, а все рассматриваемые системы инвариантных классов — пра-

в и л ь н ы е. Первое ограничение не является, по существу, ограничением, поскольку можно всегда осуществить формальное сведéние системы автоматов одного и того же типа в один автомат, объединяя множества их состояний, а также входные и выходные алфавиты, определяя все переходы и выходы так, как они были определены в объединяемых автоматах, и считая, что все остальные переходы и выходы неопределенны.

В случае приведения одного автомата A по правильной системе инвариантных классов можно, отправляясь от описанной выше общей методики приведения, сформулировать следующие частные правила приведения:

1. *В таблицах переходов и выходов (обычной или сдвинутой) автомата A заменяем все содержащиеся в них символы состояний автомата A содержащими их классами заданной правильной системы (ввиду условия правильности системы такие классы определяются однозначно).*

Из условия в) определения 10.1 и правильности системы вытекает, что после применения правила 1 во всех одинаково обозначенных столбцах таблицы переходов на пересечении с любой строкой будут стоять либо символы одного и того же класса, либо черточки (символы неопределенности). В силу условия б) определения 10.1 то же самое будет иметь место и в таблице выходов: во всех одинаково обозначенных столбцах таблицы выходов (обычной или сдвинутой) на пересечении с любой строкой будут стоять либо символы одного и того же выходного сигнала, либо черточки. Условимся называть такие столбцы *совместимыми*. Следующее правило называется *правилом объединения* совместимых столбцов.

2. *Каждая группа одинаково обозначенных (совместимых) столбцов как в таблице переходов, так и в таблице выходов объединяется в один столбец, сохраняющий то же самое обозначение, что и все объединяемые столбцы. Если на пересечении с какой-либо строкой во всех объединяемых столбцах стояли черточки, то черточка ставится и на месте пересечения с этой строкой столбца α, полученного в результате их объединения. В случае же когда хотя бы в одном из объединяемых столбцов на пересечении с данной строкой β стоял определенный (отличный от черточки) символ (он будет одним и тем же во всех объединяемых*

столбцах в силу условия совместимости), то этот символ ставится (на пересечении со строкой β) и в столбце α.

Полученные в результате применений правил 1 и 2 таблицы переходов и выходов (обычная или сдвинутая) задают автомат B, являющийся, как нетрудно видеть, приведенной формой исходного автомата A по заданной (правильной) системе инвариантных классов.

В случае приведения одного автомата по правильной системе инвариантных классов в приведенной форме фиксируется начальное состояние в соответствии со следующим правилом:

3. *В качестве начального состояния в автомате, построенном по правилам* 1 *и* 2, *выбирается инвариантный класс, содержащий начальное состояние исходного автомата* A *(такой класс определяется однозначно ввиду правильности заданной системы инвариантных классов).*

Из проведенных рассуждений вытекает следующее предложение.

10.3. *Приведенная форма автомата по правильной системе инвариантных классов определяется однозначно.*

Из предложения 10.2 и правила 3 вытекает также следующий результат.

10. 4. *Приведенная форма любого автомата по любой правильной системе инвариантных классов является эквивалентным продолжением этого автомата. Приведенная форма любого вполне определенного автомата эквивалентна ему.*

Опишем теперь основные методы, с помощью которых находятся правильные системы инвариантных классов. Следует отметить, во-первых, что в ряде частных случаев такие системы можно находить непосредственно по таблицам переходов и выходов заданного автомата, не прибегая ни к каким специальным построениям. Сформулируем два предложения, относящихся к таким случаям:

10. 5. *Предположим, что в автомате* A *имеется некоторое множество* N *состояний, обладающих следующими свойствами:*

а) *все состояния множества* N 1*-совместимы (если* A *— автомат Мили) или* O*-совместимы (если* A *— автомат Мура),*

б) *в любом столбце таблицы переходов автомата A, обозначенном состоянием из множества N, встречаются только состояния из того же самого множества. В таком случае система, состоящая из множества N и одноэлементных множеств состояний, не входящих в N, является правильной системой инвариантных классов. Приведенная форма автомата A по этой системе получается с помощью замены всех состояний множества N новым состоянием и объединения всех столбцов в таблицах переходов и выходов автомата A, обозначенных этим новым состоянием.*

10. 6. *Если в автомате A имеется множество N таких состояний, что все обозначенные ими столбцы как в таблице переходов, так и в таблице выходов (обычной или сдвинутой) совместимы (в частности, одинаковы), то система, состоящая из множества N и одноэлементных множеств всех остальных состояний, является правильной системой инвариантных классов. Приведенная форма автомата A по этой системе получается с помощью замены всех состояний множества N новым состоянием и объединения всех столбцов в таблицах переходов и выходов автомата A, обозначенных этим новым состоянием.*

Доказательства предложений 10.5 и 10.6 вытекают непосредственно из правил 1 и 2 и определения 10.1.

В общем случае системы инвариантных классов могут быть получены путем расщепления обобщенных 0-классов или обобщенных 1-классов, подобно тому, как в предыдущем параграфе получались ∞-классы. При этом, в отличие от систем i-классов, которые фигурировали в предыдущем параграфе, мы будем пользоваться системами обобщенных i-классов.

10. 7. *Системой обобщенных i-классов ($i=0, 1, 2, \ldots$) состояний автомата A мы будем называть любую совокупность множеств $K_1(i), \ldots, K_m(i)$ его состояний, удовлетворяющую следующим двум условиям:*

а) *любое состояние автомата A содержится хотя бы в одном из множеств $K_1(i), \ldots, K_m(i)$,*

б) *любое множество $K_j(i)$ ($j=1, \ldots, m$) состоит из i-совместимых между собой состояний.*

Система обобщенных i-классов будет называться *правильной*, если входящие в нее классы попарно не пересекаются.

Расщеплением системы обобщенных i-классов $K_1(i),\ldots,K_m(i)$ *автомата A мы будем называть всякую систему обобщенных* $(i+1)$*-классов* $K_1(i+1),\ldots,K_n(i+1)$*, удовлетворяющую следующим двум условиям:*

а) *каждое из множеств* $K_1(i+1),\ldots,K_n(i+1)$ *целиком содержится в одном из множеств* $K_1(i),\ldots,K_m(i)$,

б) *для любого входного сигнала x автомата A и для любого обобщенного* $(i+1)$*-класса* $K_j(i+1)$ *все состояния этого класса переводятся (с точностью до неопределенных переходов) входным сигналом x в состояния, принадлежащие одному и тому же обобщенному i-классу (зависящему от выбора* $(i+1)$*-класса* $K_j(i+1)$ *и входного сигнала x).*

Условимся также в качестве расщепления правильной системы обобщенных i-классов всегда выбирать систему обобщенных $(i+1)$-классов, также являющуюся правильной.

В случае правильных систем находить расщепления удобно с помощью так называемых *i-таблиц*.

Для произвольного конечного автомата A и произвольной системы обобщенных i-классов его состояний i-таблица автомата A получается в результате подстановки в таблицу его переходов на места состояний, содержащих эти состояния обобщенных i-классов. При этом обобщенными i-классами заменяются не только элементы таблицы переходов, но и состояния, обозначающие ее столбцы. Те места таблицы переходов, в которых переходы были не определены, остаются неопределенными и в i-таблице.

Столбцы i-таблицы вместе с обозначающими их обобщенными i-классами мы будем называть *обобщенными столбцами*. Каждый обобщенный столбец α отмечается состоянием автомата A, обозначающим столбец таблицы переходов, из которого возник столбец α.

Обобщенные столбцы i-таблицы называются *совместимыми*, если на соответствующих друг другу местах всех столбцов этого рода, в которых переходы были определены, стоят одинаковые обобщенные i-классы.

Например, если дана таблица переходов некоторого автомата A (табл. II. 50) и два обобщенных i-класса $a_i=(1, 2, 3)$ и $b_i=(4, 5)$, то i-таблица будет иметь вид, показанный на табл. II. 51. Совместимыми будут обобщенные i-столбцы, отмеченные состояниями 2 и 3, а также обобщенные столбцы, отмеченные состояниями 4 и 5

Таблица II. 50

	1	2	3	4	5
x	2	4	—	3	1
y	3	—	5	2	—

Таблица II. 51

	1	2	3	4	5
	a_i	a_i	a_i	b_i	b_i
x	a_i	b_i	—	a_i	a_i
y	a_i	—	b_i	a_i	—

Легко видеть, что состояния, отмечающие совместимые обобщенные столбцы i-таблицы, будут $(i+1)$-совместимыми. Каждое множество таких состояний переводится любым входным сигналом x в множество, содержащееся целиком внутри одного из обобщенных i-классов. Следовательно, справедливо следующее предложение.

10.8. *Для того чтобы получить расщепление правильной системы обобщенных i-классов автомата A, достаточно разбить все состояния автомата A на попарно непересекающиеся множества так, чтобы все состояния, входящие в любое из таких множеств, отмечали бы совместимые между собой обобщенные столбцы i-таблицы автомата A.*

Условимся называть систему обобщенных i-классов *нерасщепляемой*, если существует такое ее расщепление, которое совпадает с исходной системой. Из определения расщепления и определения 10.1 непосредственно вытекает следующее предложение.

10.9. *Всякая нерасщепляемая система обобщенных i-классов состояний автомата является системой инвариантных классов.*

В случае конечных автоматов безграничное расщепление системы обобщенных i-классов, очевидно, невозможно,— через конечное число шагов мы обязательно придем к нерасщепляемой системе. В связи с этим можно построить следующий алгоритм минимизации конечных автоматов, который мы будем называть *общим алгоритмом минимизации*.

Первый шаг общего алгоритма минимизации состоит в разбиении множества состояний данного автомата A на попарно непересекающиеся обобщенные 0-классы (если A — автомат Мура) или

1-классы (если A — автомат Мили). Построенное разбиение называется исходной правильной системой обобщенных i-классов состояний автомата A ($i=0$ или 1).

Второй шаг алгоритма состоит в том, что, отправляясь от исходной системы, на основании предложения 10.8 мы строим ее последовательные расщепления, пока при некотором $i=k$ мы не придем к нерасщепляемой (правильной) системе обобщенных i-классов. Полученная система обобщенных k-классов принимается за систему инвариантных классов.

Третий шаг алгоритма состоит в том, что по найденной (правильной) системе инвариантных классов с помощью сформулированных выше правил 1—3 мы находим приведенную форму B автомата A. Согласно теореме 10.4 автомат B является эквивалентным продолжением исходного автомата A.

Описанный алгоритм основывается на идеях Мили, Ауфенкампа и Хона[1]). Следует заметить, что хотя мы и назвали его общим алгоритмом минимизации, однако он не включает в себя описанного в предыдущем параграфе основного алгоритма минимизации, поскольку последний оперирует, вообще говоря, с попарно пересекающимися классами. Для случая же вполне определенных автоматов общий алгоритм действительно обобщает основной алгоритм.

Можно так усовершенствовать общий алгоритм минимизации, что он сможет работать также и с попарно пересекающимися обобщенными i-классами и инвариантными классами. Однако при этом он сильно усложняется и делается гораздо менее наглядным. Поэтому мы не будем делать таких усовершенствований.

Приведем еще два простых результата, позволяющих осуществлять предварительную минимизацию автоматов.

[1]) G. H. Mealy, A method for synthesizing sequential circuits. Bell. System Tech. J., v. 34, 1955, p. 1045—1079; D. D. Aufenkamp, Analysis of sequential machines, II. IRE Trans. EC—7, № 4, 1958, p. 299—306 (русский перевод в сб. переводов «Математика», 1959, № 3 : 3, стр. 145—158); D. D. Aufenkamp and F. E. Hohn, Analysis of sequential machines. IRE Trans. EC—6, № 4, 1957, p. 276—285 (русский перевод в сб. переводов «Математика», 1959, № 3 : 3, стр. 129—146).

10.10. *Если в автомате Мили A имеется состояние a, для которого функция выходов не определена ни при каком входном сигнале, то такое состояние можно исключить из автомата, выбросив обозначенные им столбцы из таблиц переходов и выходов автомата A и заменив все остальные вхождения состояния a в таблицу переходов автомата A черточками. Возникший в результате такой операции автомат Мили B индуцирует то же самое отображение, что и автомат A.*

Для доказательства предложения 10.10 достаточно заметить, что любое слово p, переводящее автомат A из начального состояния в состояние a, является предзапрещенным, то есть становится запрещенным после присоединения любой буквы входного алфавита. Но выходное слово, появляющееся в результате подачи на вход автомата слова p, очевидным образом не зависит от того, определен или не определен переход, совершающийся под влиянием последней буквы слова p. Так как исключение состояния a может отразиться лишь на преобразовании автоматом слов типа слова p (переводящих автомат в состояние a), то последнее соображение и завершает доказательство предложения 10.10.

10.11. *Если в автомате Мура A имеется (отличное от начального состояния) состояние a с неопределенной отметкой, то это состояние можно исключить из автомата, выбросив обозначенный им столбец из отмеченной таблицы переходов и заменив черточками все остальные вхождения состояния a в эту таблицу. В результате такого исключения отображение, индуцируемое автоматом, не изменится.*

Доказательство предложения 10.11 заключается в последовательном применении к автомату A предложений 1.7 и 1.9 из § 1.

Значение предложений 10.10 и 10.11, а также доказанных выше предложений 10.5 и 10.6 состоит в том, что они часто позволяют производить предварительное упрощение автомата еще до применения к нему общего или основного алгоритма минимизации. Это обстоятельство оказывается весьма благоприятным для сокращения общего времени, затрачиваемого на минимизацию, поскольку с увеличением числа состояний автомата выкладки, связанные с применением общего и основного алгоритма мини-

мизации, становятся весьма громоздкими и сильно подверженными случайным ошибкам. Поэтому никогда не следует упускать возможности предварительной минимизации автомата с использованием относительно простых средств. В качестве таких средств, кроме средств, которые дают предложения 10.5, 10.6, 10.10 и 10.11, используется также метод исключения недостижимых состояний, описанный в § 1.

В заключение настоящего параграфа рассмотрим несколько примеров минимизации автоматов.

Пример 1. Минимизировать автомат Мили A, заданный таблицей переходов II. 52 и таблицей выходов II. 53.

Таблица II. 52

	1	2	3	4	5
x	2	—	5	1	—
y	—	3	—	—	—
z	—	4	—	—	—

Таблица II. 53

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

Решение. Состояние 5 можно исключить на основании теоремы 10.10. После этого столбцы 3 и 4 становятся совместимыми в обеих таблицах: то же самое справедливо и для столбцов 1 и 2. Применяя теорему 10.6, можно заменить каждый из этих двух столбцов одним, после чего мы получаем таблицу переходов II.54 и таблицу выходов II.55.

Таблица II.54

	a	b
x	a	a
y	b	—
z	b	—

Таблица II 55

	a	b
x	u	u
y	v	v
z	u	v

Поскольку состояния a и b не являются 1-совместимыми (что видно из таблицы выходов), то дальнейшая минимизация оказывается невозможной.

Пример 2. Минимизировать автомат Мура A, заданный отмеченной таблицей переходов II.56.

Таблица II.56

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

Решение. Ни один из описанных выше простых приемов минимизации в данном случае неприменим. Поэтому мы применим общий алгоритм минимизации. В качестве исходной системы обобщенных 0-классов выберем систему $a_0=(1, 2, 3)$, $b_0=(4,5)$. 0-таблица будет иметь вид, представленный в табл. II.57.

Таблица II. 57

	1	2	3	4	5
	a_0	a_0	a_0	b_0	b_0
x	a_0	b_0	a_0	b_0	a_0
y	a_0	b_0	a_0	b_0	b_0

Таблица II.58

	1	2	3	4	5
	a_1	b_1	a_1	c_1	d_1
x	b_1	c_1	b_1	c_1	b_1
y	a_1	d_1	a_1	d_1	c_1

По этой таблице на основании предложения 10.8 мы находим следующую систему обобщенных 1-классов: $a_1=(1,3)$, $b_1=(2)$, $c_1=(4)$, $d_1=(5)$ и строим 1-таблицу (табл. II.58).

В соответствии с теоремой 10.8 можно выбрать расщепление системы обобщенных 1-классов, совпадающее с ней самой. Следовательно, эта система нерасщепляема и, на основании теоремы 10.9, является системой инвариантных классов. Можно поэтому по правилам 1, 2 и 3 построить отмеченную таблицу переходов приведенной формы автомата A (табл. II.59).

После перенумерации состояний отмеченная таблица переходов примет вид, показанный в табл. II.60.

Пример 3. Минимизировать автомат Мура A, построенный в примере 2 из § 8.

Таблица II.59

	u	u	v	v
	a	2	4	5
x	2	4	4	2
y	a	5	5	4

Таблица II.60

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

Решение. Автомат A задается следующей отмеченной таблицей переходов II.61:

Таблица II.61

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

На основании теоремы 10.6 можно объединить столбцы 3 и 5, после чего мы придем к отмеченной таблице переходов II.62.

Таблица II.62

	—	1	0	0	0	1	0
	0	1	2	3	4	6	7
0	2	7	7	6	3	7	7
1	1	4	3	7	7	7	7

Построим систему обобщенных 0-классов: $a_0=(0, 2, 3, 4, 7)$, $b_0=(1, 6)$ и выпишем 0-таблицу. Остальные i-таблицы будем для экономии мест выписывать под нею (см. табл. II.63), помещая рядом с каждой i-таблицей соответствующие обобщенные i-классы.

Поскольку процесс расщепления привел к восстановлению исходных состояний, то приведенная форма автомата совпадает с ним самим.

Таблица II.63

	0	1	2	3	4	6	7	
	a_0	b_0	a_0	a_0	a_0	b_0	a_0	$a_0=(0,2,3,4,7),\ b_0=(1,6)$
0	a_0	a_0	a_0	b_0	a_0	a_0	a_0	
1	b_0	a_0	a_0	a_0	a_0	a_0	a_0	
	a_1	b_1	c_1	d_1	c_1	b_1	c_1	$a_1=(0),\ b_1=(1,6),\ c_1=$ $=(2,4,7),\ d_1=(3)$
0	c_1	c_1	c_1	b_1	d_1	c_1	c_1	
1	b_1	c_1	d_1	c_1	c_1	c_1	c_1	
	a_2	b_2	c_2	d_2	e_2	b_2	f_2	$a_2=(0),\ b_2=(1,6),\ c_2=(2),\ d_2=$ $=(3),\ e_2=(4),\ f_2=(7)$
0	c_2	f_2	f_2	b_2	d_2	f_2	f_2	$a_3=(0),\ b_3=(1),\ c_3=(2),\ d_3=$
1	b_2	e_2	d_2	f_2	f_2	f_2	f_2	$=(3),\ e_3=(4),\ f_3=(7),\ g_3=(6).$

При минимизации мы отнесли состояние 0 в класс a_0. Если бы его отнести в другой класс b_0, то снова получилось бы расщепление классов до единичных состояний. Используя теперь теорему 9.10 предыдущего параграфа, нетрудно показать, что после проведенного выше объединения состояний 3 и 5 получен автомат Мура B, имеющий наименьшее возможное число состояний среди всех автоматов Мура, эквивалентных исходному автомату A.

Пример 4. Минимизировать автомат Мили A, рассмотренный в примере 3 из § 9, который задается таблицей переходов II. 64 и таблицей выходов II.65.

Таблица II.64

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

Таблица II.65

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

Р е ш е н и е. Из таблицы выходов автомата A непосредственно видно, что все его состояния попарно 1-несовместимы. Недостижимых состояний также не имеется. Следовательно, дальнейшая минимизация автомата A описанными выше средствами невозможна. Нетрудно показать, что автомат A не может быть минимизирован никакими средствами, ибо для реализации четырех различных столбцов выходных сигналов он должен иметь не менее четырех различных состояний.

ГЛАВА III

СТРУКТУРНАЯ ТЕОРИЯ АВТОМАТОВ

§ 1. Композиция автоматов, структурные схемы

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

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

Структурная теория автоматов и абстрактная теория автоматов являются двумя различными ступенями общей теории автоматов. При синтезе реальных автоматов многие вопросы проще и эффективнее решаются на уровне абстрактной теории. К числу таких вопросов относится определение необходимого объема *памяти автомата* (то

есть числа его состояний), переходов в памяти, а также вопросы, относящиеся к минимизации числа состояний автомата. Рассмотрение этих вопросов на уровне структурной теории повлекло бы за собой загромождение хода рассуждения несущественными (и к тому же подчас затемняющими суть дела) подробностями. Более того, трудности эквивалентных преобразований структурных схем автоматов с памятью делают практически невозможным (по крайней мере в настоящее время) решение на уровне структурной теории такой задачи, как задача минимизации числа состояний автомата. В то же самое время имеется ряд вопросов — таких, например, как вопрос о композиции автоматов,— сама постановка которых выводит за рамки абстрактной теории автоматов. Таким образом, абстрактная теория автоматов и структурная теория автоматов взаимно дополняют друг друга и имеют собственные естественные области приложения.

В структурной теории автоматов сохраняется абстракция дискретного автоматного времени, однако при построении этой теории оказывается более удобным несколько изменить порядок отсчета временных интервалов, принятый в абстрактной теории. В абстрактной теории автоматов было удобнее (для того, чтобы не создавать двух различных теорий для автоматов первого и второго рода) относить входные и выходные сигналы к моменту перехода автомата из одного состояния в другое. В структурной теории мы будем придерживаться более естественного способа отсчета времени, считая моменты перехода автомата из одного состояния в другое г р а н и ц а м и интервалов, относящихся к одному и тому же значению автоматного времени.

При таком способе отсчета времени определение закона функционирования автоматов первого рода (автоматов Мили) будет выглядеть следующим образом:

$$a(t+1)=\delta(a(t),\ x(t)),\quad y(t)=\lambda(a(t),\ x(t))\quad (t=0,\ 1,\ 2,\ldots).$$

Закон функционирования автоматов Мура приобретет вид

$$a(t+1)=\delta(a(t),\ x(t)),\quad y(t)=\lambda(a(t))\quad (t=0,\ 1,\ 2,\ldots).$$

Нетрудно убедиться в том, что и для этой формулировки законов функционирования автоматов сохраняются все результаты, полученные в предыдущей главе. Правда, они нуждаются в несколько иной интерпретации, учитывающей особенности принимаемого теперь способа отсчета времени. В этой интерпретации следует учитывать два обстоятельства.

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

Это замечание касается в равной степени как автоматов Мура, так и автоматов Мили и обусловливает необходимость рассмотрения входных и выходных последовательностей вида $x(0)\,x(1)\ldots x(t-1)$ вместо последовательностей вида $x(1)\,x(2)\ldots x(t)$. Ясно, что подобный сдвиг во времени ни в коей мере не влияет на правильность полученных в предыдущей главе результатов.

Второе обстоятельство, относящееся только к автоматам Мура, связано с тем, что при новом способе отсчета времени результатом применения входного слова $x(0)x(1)\ldots x(t-1)$ к начальному состоянию автомата Мура следует считать не выходное слово $y(0)\,y(1)\ldots y(t-1)$, а выходное слово $y(1)\,y(2)\ldots y(t)$.

Это обстоятельство связано с тем, что для автоматов Мура выходной сигнал, индуцированный каким-либо входным сигналом $x(t)$, при новом способе отсчета времени относится не к моменту t появления сигнала $x(t)$, а к непосредственно следующему за ним моменту времени $t+1$. Все результаты предыдущей главы сохраняют свою силу при условии их интерпретации с учетом этого обстоятельства. Единственный новый момент, вносимый введением нового способа отсчета времени, состоит в том, что теперь пустое слово оказывается всегда представленным в автоматах Мура выходным сигналом $y(0)$, в то время как ранее оно не могло быть представлено никаким выходным сигналом. Эта особенность новой интерпретации, однако, несущественна в силу принятого в предыдущей главе условия, согласно которому пустое слово не принимается во внимание при представлении событий.

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

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

В каждом автомате осуществляется определенная циркуляция элементарных сигналов. Элементарные входные сигналы поступают сначала на входные узлы, а затем по присоединенным к ним элементарным входным каналам передаются внутрь автомата. Элементарные выходные сигналы, выходя из автомата по элементарным выходным каналам, поступают на выходные узлы, к которым эти каналы присоединены.

При графическом изображении автоматов узлы обозначаются точками или маленькими кружочками, элементарные входные и выходные каналы — сплошными линиями, а сами автоматы — различными геометрическими фигурами (прямоугольниками, треугольниками, кругами и т. п.). Для того чтобы отличить входные элементарные каналы от выходных элементарных каналов направление передачи по ним элементарных сигналов иногда отмечают стрелками. Согласно принятому соглашению относительно циркуляции элементарных сигналов это однозначно определяет характер соответствующих элементарных каналов. Например, на рис. 5 изображен автомат с двумя

выходными и тремя входными элементарными каналами.

Под термином «автомат» в структурной теории автоматов понимается абстрактный автомат с явно заданными элементарными входными и выходными каналами и соответствующими им входными и выходными узлами. При этом предполагается также заданной некоторая определенная нумерация как входных, так и выходных узлов автомата. Входной и выходной сигналы такого автомата задаются конечными упорядоченными наборами элементарных сигналов. Подобные наборы мы будем называть *векторами* в структурном алфавите, составляющие их элементарные сигналы — *компонентами вектора*, а число компонент — *размерностью вектора*. Нумерация компонент этих векторов соответствует нумерации входных и выходных узлов, так что на i-й входной узел передается i-я компонента *входного вектора* (то есть вектора, изображающего входной сигнал), а на j-й выходной узел — j-я компонента *выходного вектора* (вектора, изображающего выходной сигнал).

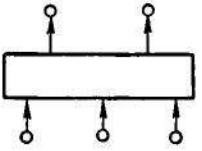

Рис. 5.

В отличие от абстрактных входных и выходных сигналов, рассматривавшихся в предыдущей главе, векторные представления таких сигналов называются *структурными* (*входными* и *выходными*) сигналами. Переход от абстрактных входных и выходных сигналов к структурным сигналам носит название *кодирования* соответствующих абстрактных сигналов в структурном алфавите автомата. Вектор, получающийся в результате кодирования какого-либо абстрактного сигнала, обозначается обычно той же буквой, что и этот сигнал, но жирным шрифтом. При этом записью $\boldsymbol{x}$ мы будем выражать обычно переменный входной структурный сигнал, а записью $\boldsymbol{y}$ — переменный выходной структурный сигнал автомата.

В структурной теории принято несколько отличное от абстрактной теории понимание частичных автоматов. Частичные автоматы рассматриваются здесь как такие автоматы, у которых функции переходов и выходов

в действительности (в каждой конкретной реализации автомата) всюду определены, однако в некоторых точках значения этих функций для нас безразличны, так что при желании мы можем определить их иначе, выбрав другую реализацию соответствующего автомата.

Мы будем считать также, что на всех выходных узлах всякого автомата Мура выходные сигналы возникают в каждый момент автоматного времени, независимо от того, подаются ли какие-либо сигналы на его входные узлы или нет. Что касается автоматов Мили, то там положение будет иным: сигналы на выходных узлах автомата Мили в тот или иной момент автоматного времени появляются тогда и только тогда, когда в т о т ж е с а м ы й момент времени сигналы поданы на в с е его входные узлы.

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

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

Переходя к описанию способов к о м п о з и ц и и автоматов, условимся всякий раз при рассмотрении той или иной системы автоматов считать, не оговаривая этого осо-

бо, что все входящие в систему автоматы имеют один и тот же структурный алфавит и работают в одном и том же дискретном автоматном времени. Заметим, что введение общего автоматного времени для системы автоматов не означает, вообще говоря, отказ от рассмотрения а с и н х р о н н ы х автоматов. Речь идет здесь о том, что совместная работа автоматов в системе определяет общее дискретное время для всех входящих в нее автоматов, отличное, вообще говоря, от того автоматного времени, в котором эти автоматы работали бы вне данной системы.

Сформулируем теперь определение *общего способа композиции автоматов.*

Пусть $A_1,\ldots,A_n$ (где $n\geqslant 0$) — конечное множество автоматов. Произведем объединение этих автоматов в систему совместно работающих автоматов следующим образом.

Введем в рассмотрение некоторое конечное множество узлов, которые назовем *внешними входными узлами*, и некоторое конечное множество других узлов, которые назовем *внешними выходными узлами*. Эти узлы предполагаются отличными от входных и выходных узлов рассматриваемых автоматов, которые, в отличие от только что введенных в н е ш н и х узлов, мы будем называть *внутренними* (*входными* и *выходными*) *узлами*. Чтобы подчеркнуть различие между двумя введенными типами узлов, внешние узлы называются иногда также *полюсами*. При графическом изображении системы автоматов внутренние узлы чаще всего обозначаются точками, а внешние — кружочками. При построении системы автоматов фиксируется определенная нумерация как для внешних входных, так и для внешних выходных узлов (полюсов).

Собственно композиция автоматов состоит в том, что в полученной системе, состоящей из данных автоматов $A_1,\ldots,A_n$ и внешних узлов, производится *отождествление некоторых узлов* (как внешних, так и внутренних). С точки зрения совместной работы системы автоматов смысл операции отождествления узлов состоит в том, что элементарный сигнал, попадающий на один из узлов, входящих в множество отождествленных между собой узлов, попадает тем самым на все узлы этого множества. В случае электронных цифровых автоматов операции отождествления узлов соответствует соединение этих узлов проводни-

ками. Мы будем поэтому часто вместо отождествления узлов говорить об их *соединении* друг с другом.

При графическом изображении отождествление узлов производится либо с помощью их фактического совмещения, либо с помощью соединения их сплошными линиями (вообще говоря, ломаными и, возможно, проходящими через другие узлы).

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

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

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

Условимся говорить, что полученный только что описанным способом автомат A есть результат композиции исходных автоматов $A_1,\ldots,A_n$. Входными узлами этого автомата служат внешние входные узлы схемы, а выходными узлами — внешние выходные узлы.

Следует подчеркнуть, что операция отождествления узлов определена неоднозначно. Поэтому существует, вообще говоря, много различных возможностей для компози-

ции одних и тех же автоматов. Вместе с тем далеко не всякое отождествление узлов приводит к схеме, которую можно рассматривать в качестве структурной схемы некоторого автомата. Для того чтобы это можно было сделать, необходимо при отождествлении узлов соблюдать два условия, которые мы будем называть *условиями корректности построения схемы*.

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

Второе условие корректности построения схемы состоит в том, что *неоднозначность элементарных сигналов в каком-нибудь узле схемы хотя бы в один момент времени является недопустимой.*

Существуют два источника неоднозначности элементарного сигнала в узле. Во-первых, неоднозначность может иметь место в случае, если к какому-нибудь узлу подсоединено одновременно несколько выходных узлов, являющихся источниками элементарных сигналов. При построении схемы необходимо, таким образом, избегать подобных соединений. На практике, однако, подсоединение к одному и тому же узлу нескольких выходных узлов часто не приводит к возникновению неоднозначности. Это бывает тогда, когда среди элементарных сигналов определена некоторая естественная упорядоченность, в силу которой одновременный приход нескольких элементарных сигналов эквивалентен приходу лишь наибольшего из всех фактически пришедших сигналов. Мы будем говорить, что в этом случае имеет место *естественное разделение элементарных сигналов*. Такое естественное разделение наблюдается, например, в случае релейно-контактных схем, а также во многих электронных схемах с импульсным представлением элементарных сигналов.

Во-вторых, источником неоднозначности элементарных сигналов в узле могут быть так называемые *циклические цепи* или *петли* структурных схем. Условимся называть *цепью* конечную упорядоченную последовательность автоматов $B_1,\ldots,B_k$, входящих в схему, у которых хотя бы один из выходных узлов каждого предыдущего автомата соединен с некоторым входным узлом непосредственно следующего за ним автомата. Если к тому же хотя бы один из выходных узлов последнего автомата цепи соединен с каким-нибудь входным узлом первого автомата той же цепи, то цепь называется циклической (о ней также говорят, что она образует петлю в данной схеме). Обо всех узлах, которые имеются в виду при определении цепи и петли, говорят как об узлах этой цепи или этой петли.

О цепи $B_1,\ldots,B_k$ говорят, что она *начинается* любым из входных узлов автомата B_1 и *кончается* любым из выходных узлов автомата B_k. Условимся также, что два отождествленных между собою узла a и b или даже один единственный узел составляют цепь (состоящую из пустого множества автоматов). В первом случае цепь начинается любым из узлов a или b и кончается другим из этих узлов Во втором случае цепь начинается и кончается узлом a. Циклические цепи, состоящие из пустого множества автоматов, называются тривиальными и в дальнейшем обычно не рассматриваются.

Покажем на примере, каким образом может возникнуть неоднозначность в петле. Предположим, что каждый из трех автоматов, входящих в цепь, изображенную на рис. 6, обладает следующим свойством: он работает в структурном алфавите, состоящем из двух элементарных сигналов 0 и 1 и превращает входной сигнал 0 в выходной сигнал 1, а входной сигнал 1 — в выходной сигнал 0; при этом входной и соответствующий ему выходной сигналы возникают в один и тот же момент времени.

Легко видеть, что такое определение сигналов во всех входящих в цепь узлах не может быть сделано без противоречия. Действительно, полагая, что в какой-то момент времени в узле *1* имелся сигнал 0, мы получим, что в узле *2* в тот же самый момент времени должен иметься сигнал 1,

в узле *3* — сигнал 0, а, следовательно, в узле *1* — сигнал 1. Полагая, что в узле *1* имелся сигнал 1, мы также придем к противоречию.

Петли, подобные только что рассмотренной петле, *некорректно построены*. Некорректность такого рода не будет иметь места, если хотя бы один из входящих в петлю автоматов является *автоматом Мура*, а не автоматом Мили. Действительно, автомат Мура реагирует на тот или иной входной сигнал выходным сигналом, возникающим на один элементарный промежуток времени *позже*, чем вызвавший его появление входной сигнал. Поэтому в рассмотренном примере сигнал 0, переданный в момент времени t в узел *1*, вызвал бы в результате передачи его вдоль петли появление в узле *1* сигнала 1 не в момент времени t, а в один из более поздних моментов времени. Таким образом, никакого противоречия не возникло бы.

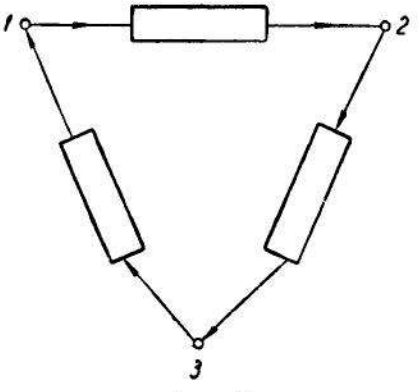

Рис. 6.

Разберем теперь вопрос о том, каким образом при построении схемы можно удовлетворить первому условию корректности. При включении в структурный алфавит естественного нулевого сигнала это условие соблюдается автоматически. В противном же случае для соблюдения этого условия необходимо, при проведении операции отождествления узлов, выполнить определенные, описываемые ниже требования.

Прежде всего заметим, что все внешние входные узлы и все внутренние узлы, являющиеся выходными узлами автоматов Мура, по определению, получают элементарные сигналы во все моменты автоматного времени. Назовем все эти узлы *задающими узлами*. Нетрудно сформулировать условие, обеспечивающее передачу элементарных

сигналов от задающих узлов на все узлы автомата в каждый момент автоматного времени. Для этой цели достаточно напомнить, что выходные сигналы у автоматов Мили, по определению, возникают одновременно с поступлением сигналов на все его входные узлы. Соответствующее условие можно сформулировать, следовательно, таким образом:

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

Назовем это условие *первым условием правильности построения схемы.* Проведенные выше рассуждения позволяют сформулировать два других условия правильности схемы:

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

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

Схема, в которой выполнены все три условия: 1.1, 1.2 и 1.3, называется *правильной схемой*, а операция, состоящая в построении такой схемы — *правильной композицией автоматов.*

Из проведенных выше рассуждений вытекает следующий результат.

1.4. *Всякая правильная схема может рассматриваться как структурная схема некоторого автомата.*

Предложение 1.4 допускает, очевидно, и другую, эквивалентную, формулировку.

В результате правильной композиции любого конечного множества автоматов получается схема некоторого автомата.

Таким образом, правильная композиция задает некоторый класс операций на множестве автоматов. Рассмотрим некоторые частные случаи таких операций.

Первый случай — *операция пересоединения внешних узлов* (композиция пустого множества автоматов). В этом случае схема предполагается состоящей только из внешних (входных и выходных) узлов. Из условий 1.1 и 1.2

непосредственно следует, что каждый выходной узел в такой схеме должен быть соединен в точности с одним входным узлом. Схема представляет собою автомат Мили без памяти (с одним-единственным состоянием). Компонентами структурного выходного сигнала автомата служат компоненты его структурного входного сигнала, взятые, вообще говоря, в другом порядке и, быть может, повторенные по несколько раз.

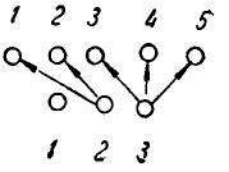

Рис. 7.

На рис. 7 показан один из примеров такой композиции.

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

Второй случай — операция *подстановки входов в автомате*. Эта операция состоит в следующем. Рассматривается произвольный автомат A с m входными и с n выходными узлами. Строится система, состоящая из автомата A из m внешних входных и из n выходных узлов, i-й выходной узел автомата A отождествляется с i-м внешним выходным узлом ($i=1,\ldots,n$). Каждый входной узел автомата A отождествляется в точности с одним внешним входным узлом. Полученный в результате композиции автомат действует так же, как и исходный автомат A при условии, что на автомат A подается не исходный структурный входной сигнал x, а структурный входной сигнал, полученный из него в результате некоторой подстановки (вообще говоря, неоднозначной) его компонент. На рис. 8 показан один из примеров такой конструкции.

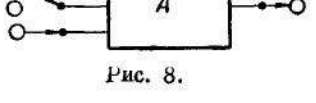

Рис. 8.

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

Четвертый случай — операция *суперпозиции автоматов*. Эта операция состоит в следующем. Рассматриваются два автомата A и B. Пусть у автомата A имеется m входных и n выходных узлов, а у автомата

B — n входных и p выходных узлов. Суперпозиция этих автоматов заключается в построении системы, состоящей из автоматов A и B, m внешних входных и p внешних выходных узлов посредством соединения i-го внешнего входного узла с i-м входным узлом автомата A ($i=1,\ldots,m$), j-го выходного узла автомата A с j-м входным узлом автомата B ($j=1,\ldots,n$) и k-го выходного узла автомата B с k-м внешним выходным узлом ($k=1,\ldots,p$).

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

Все описанные конструкции являются частными случаями правильной композиции автоматов, причем такой композиции, в результате которой получаются схемы без петель. Нетрудно убедиться в том, что операции подстановки входов и выходов (второй и третий случаи) могут быть получены в результате последовательного применения операций пересоединения внешних узлов и суперпозиции автоматов (первый и четвертый случаи). Операции же, описанные в первом, четвертом и пятом случаях, не выражаются, очевидно, друг через друга.

В дальнейшем мы будем иметь дело в основном с правильными схемами. Однако не следует забывать, что правильные схемы не исчерпывают всех корректно построенных схем; это позволяет в ряде случаев пользоваться схемами, не принадлежащими к числу правильных. Укажем два важнейших случая такого рода.

Во-первых, в случае наличия в структурном алфавите естественного нулевого сигнала можно не соблюдать, очевидно, условие правильности 1.1 и тем не менее получать корректно построенные схемы. Оказывается возможным, однако, так интерпретировать эти схемы, что

условие 1.1 будет в них выполненным. Эта интерпретация осуществляется с помощью введения так называемого нулевого автомата.

Нулевым автоматом мы будем называть автомат Мили с одним входным и одним выходным узлом, отличающийся тем, что на его выходном узле при л ю б о м сигнале на входном узле появляется нулевой сигнал.

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

В о - в т о р ы х, в случае, когда имеется естественное разделение элементарных сигналов, можно, очевидно, не соблюдать условие правильности 1.2 и, тем не менее, получать корректно построенные схемы. Однако и в этом случае оказывается возможным так интерпретировать указанные схемы, что условие 1.2 будет выполняться. Этой цели можно достичь, введя понятие *разделяющего автомата Мили*, или просто *разделения*. В разделяющем автомате Мили на единственный выходной узел автомата передается всегда самый большой элементарный сигнал (в заданной естественной упорядоченности элементарных сигналов) из числа входных сигналов, поданных в то же самое время на его входные узлы. Если в схеме с естественным разделением сигналов имеется соединение какого-нибудь узла с несколькими внешними входными или внутренними выходными узлами $\alpha_1,\ldots,\ \alpha_k$, то можно считать этот узел соединенным с выходным узлом разделяющего автомата, входные узлы которого соединены с узлами $\alpha_1,\ldots,\ \alpha_k$. При таком дополнении схемы ее работа, очевидно, не изменится, но условие 1.2 будет уже соблюдаться.

В заключение рассмотрим вопрос о состояниях автомата A, полученного в результате композиции некоторых автоматов $A_1,\ldots,\ A_k$. Ясно, что состояниями автомата A можно считать упорядоченные наборы $(a_1,\ldots,\ a_k)$ состояний автоматов $A_1,\ldots,\ A_k$. Такой набор мы будем называть *структурным состоянием* автомата A и обозначать малой латинской жирной буквой $\boldsymbol{a}$.

В дальнейшем будем предполагать, что в результате композиции автоматов, кроме указанных комбинаций уже имевшихся состояний, других состояний не возникает. Тогда справедливо следующее:

1.5. *Число состояний автомата, полученных в результате композиции автоматов $A_1,\ldots,A_n$, равно произведению чисел состояний всех этих автоматов.*

§ 2. Канонический метод структурного синтеза автоматов

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

2.1. *Пусть нам задано некоторое конечное множество автоматов в одном и том же структурном алфавите Σ. Назовем эти автоматы элементарными автоматами и предположим, что каждый из элементарных автоматов имеется в нашем распоряжении в неограниченном числе экземпляров. Пусть далее задан произвольный конечный автомат A в том же самом структурном алфавите Σ. Необходимо найти алгоритм, позволяющий по заданному автомату A строить некоторую композицию элементарных автоматов так, чтобы полученный в результате композиции автомат индуцировал отображение, продолжающее отображение, индуцируемое автоматом A.*

Следует подчеркнуть, что далеко не при всяком выборе системы элементарных автоматов эта задача имеет решение. В том же случае, когда она имеет решение (для произвольного конечного автомата A), будем говорить, что заданная система элементарных автоматов *структурно полна*.

Иногда пользуются также понятием ослабленной структурой полноты. Именно, говорят, что система элементарных автоматов *ослабленно структурно полна*, если для любого отображения φ, индуцируемого конечным автоматом, можно найти такую композицию элементарных автоматов, что получаемый в результате этой композиции автомат индуцирует отображение, которое продолжает

либо само отображение φ, либо отображение, полученное из отображения φ в результате применения к нему операции выравнивания длин слов (см. гл. 2, § 2). Разумеется, при этом предполагается, что автомат, индуцирующий отображение φ, задан в том же структурном алфавите, что и все элементарные автоматы, о которых идет речь. Заметим также, что операция выравнивания в применении к отображению φ, являющемуся автоматным отображением, означает просто дописывание р а в н о г о ч и с л а пустых букв справа к входным словам и слева к выходным словам отображения φ.

В настоящее время нет сколько-нибудь эффективных методов (существенно более простых, чем метод перебора всех вариантов) решения основной задачи структурного синтеза при любом выборе структурно полных систем элементарных автоматов. Мы будем поэтому излагать так называемый *канонический метод* структурного синтеза автоматов, пригодный для структурно полных систем элементарных автоматов некоторого специального вида.

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

При принятых нами предположениях результатом композиции логических элементов являются всегда автоматы без памяти или, как их иначе называют, *комбинационные схемы*. Каждая комбинационная схема (автомат без памяти) характеризуется *векторной функцией выходов*, устанавливающей зависимость структурного выходного сигнала $y(x)$ от структурного входного сигнала $x(t)$ в о д и н и т о т ж е м о м е н т автоматного времени:

$$y(t) = \lambda(x(t)).$$

Задание такой функции эквивалентно заданию системы обычных (скалярных) функций, устанавливающих зави-

симость каждой компоненты вектора $y(x)$ от компонент (в общем случае всех) вектора $x(t)$.

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

2.2. *Пусть нам задано некоторое конечное множество логических элементов (автоматов без памяти) в одном и том же структурном алфавите* $\mathfrak{B}$; *предположим, что каждый из таких элементов имеется в нашем распоряжении в неограниченном числе экземпляров. Требуется найти общий конструктивный прием (алгоритм), позволяющий по любому конечному автомату* A *без памяти в структурном алфавите* $\mathfrak{B}$ *осуществить некоторую композицию заданных логических элементов так, чтобы полученная в результате композиции комбинационная схема имела ту же самую векторную функцию выходов, что и заданный автомат* A.

Далеко не при всяком выборе системы логических элементов сформулированная задача имеет решение. В том случае, когда она имеет решение (для произвольного конечного автомата без памяти), мы будем говорить, что заданная система логических элементов *функционально полна.*

Следует заметить, что на практике векторная функция выходов λ исходного, для комбинационного синтеза, автомата без памяти A бывает задана не во всех точках (точнее говоря, в некоторых точках значения ее для нас безразличны). В таком случае комбинационная схема, получаемая при решении задачи комбинационного синтеза, должна иметь векторную функцию выходов, совпадающую с функцией λ во всех точках, в которых значения функции λ определены.

Основная идея канонического метода структурного синтеза автоматов заключается в том, чтобы свести задачу структурного синтеза произвольных автоматов к задаче структурного синтеза комбинационных схем (автоматов без памяти). Для осуществления этой идеи производится специальный выбор запоминающих элементов. При каноническом методе структурного синтеза в качестве запоминающих элементов выбираются автоматы Мура, обладающие *полной системой переходов* и *полной системой выходов.*

2.3. *Полнота системы переходов в автомате означает, что для любой упорядоченной пары состояний этого автомата найдется входной сигнал, переводящий первый элемент этой пары во второй. Иначе говоря, если* $\delta(a, x)$ *— функция переходов автомата, то для полноты системы переходов в этом автомате необходимо и достаточно, чтобы для любой пары* (a, b) *его состояний уравнение* $b = \delta(a, x)$ *было бы разрешимым относительно входного сигнала* x.

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

Из определения 2.4. непосредственно вытекает следующий результат.

2.5. *В автомате Мура с полной системой выходов можно отождествить внутренние состояния с выходными сигналами автомата.* Значение этого результата состоит в том, что он позволяет употреблять один и тот же (структурный) алфавит не только для обозначения (кодирования) входных и выходных сигналов, но и для кодирования внутренних состояний автоматов (в случае автоматов Мура с полной системой выходов).

Для автоматов с полной системой переходов оказывается целесообразным ввести следующее определение.

2.6. *Функцией входов* $x = \mu(a, b)$ *автомата* A *с полной системой переходов, имеющего функцию* $\delta(a, x)$ *в качестве своей функции переходов, называется функция (вообще говоря, неоднозначная), заданная на упорядоченных парах состояний автомата* A. *Для каждой такой пары* (a, b) *в качестве значения функции входов* $\mu(a, b)$ *выбирается (непустое) множество всех тех входных сигналов* x, *для которых* $\delta(a, x) = b$.

Функции входов конечных автоматов мы будем задавать *таблицами входов*. Таблица входов отличается тем, что на пересечении a-й строки и b-го столбца в ней поме-

щено множество входных сигналов, вызывающих переход автомата из состояния a в состояние b. Следует заметить, что почти все употребляющиеся в реальных цифровых автоматах запоминающие элементы представляют собою автоматы Мура с полной системой переходов и с полной системой выходов. Таким образом, ограничения, вводимые каноническим методом структурного синтеза, с практической точки зрения являются несущественными.

Можно доказать справедливость следующего утверждения, которое мы будем называть *теоремой о структурной полноте.*

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

Для доказательства теоремы 2.7 рассмотрим произвольный конечный автомат A. Возьмем какие-либо запоминающие элементы $A_1, \ldots, A_p$, обладающие полными системами переходов и выходов, причем такие, что произведение чисел их состояний не меньше, чем число состояний автомата A. Каждому состоянию a автомата A отнесем конечную упорядоченную последовательность $(a_1, \ldots, a_p)$ состояний автоматов $A_1, \ldots, A_p$ так, что различным состояниям автомата A ставятся в соответствие различные последовательности. Этот процесс называется *кодированием состояний* автомата A.

После кодирования состояний (выполняемого произвольным образом) возникают структурные состояния автомата A (см. § 1). Используя предложение 2.5, мы можем считать эти структурные состояния векторами в структурном алфавите; компонента a_i в структурном состоянии $(a_1, \ldots, a_p)$ заменяется при этом групповой компонентой структурного выходного сигнала v_i запоминающего элемента A_i, соответствующего состоянию a_i.

Обозначим структурное состояние $(v_1, \ldots, v_p)$ через v. Его можно одновременно рассматривать не только как структурное состояние, но и как структурный выходной сигнал объединения (см. § 1) автоматов $A_1, \ldots, A_p$. Обозначим через u структурный входной сигнал этого объединения. Структурный входной сигнал u естественно назвать *структурным входным сигналом памяти автомата* A. Рассмотрим, кроме того, *внешний структурный входной сигнал* x автомата A. Благодаря проведенному выше кодированию состояний мы можем функцию переходов и закон функционирования автомата A представить в векторном виде

$$v(t+1) = \delta(v(t),\ x(t)). \tag{1}$$

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

$$\mu(v(t),\ v(t+1)) = u(t). \tag{2}$$

Подставляя в формулу (2) значение $v(t+1)$ из формулы (1), мы приходим к новой формуле

$$u(t) = \mu(v(t),\ \delta(v(t),\ x(t))) = \varrho(v(t),\ x(t)). \tag{3}$$

Определяемая этой формулой векторная функция $\varrho(v, x)$ называется (векторной) *функцией возбуждения* автомата A или *функцией входов его памяти*. Она определяет, какой структурный входной сигнал $u(t)$ в любой момент времени t должен быть подан на память автомата A, находящегося в состоянии $v(t)$, если на внешние входные узлы автомата подается в т о т ж е с а м ы й момент времени структурный входной сигнал $x(t)$.

Благодаря совпадению всех рассмотренных сигналов во времени сигнал $u(t)$ можно рассматривать как струк-

тупный выходной сигнал некоторой комбинационной схемы, отличающейся тем, что ее структурный входной сигнал получается в результате объединения структурного входного сигнала $x(t)$ автомата A и *структурного выходного сигнала* $v(t)$ *его памяти*. Реализуя эту схему в виде композиции заданных логических элементов (что возможно благодаря предположению о функциональной полноте системы логических элементов), мы осуществим, очевидно, все те переходы, которые предусматриваются функцией переходов автомата A.

Построенная комбинационная схема называется *схемой обратных связей* автомата A, а уравнение (4)

$$u = \varrho(v, x), \tag{4}$$

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

Заметим, что функция возбуждения $\varrho(v, x)$ автомата является, вообще говоря, неоднозначной функцией, поскольку неоднозначными, в общем случае, являются функции входов μ_i запоминающих элементов A_i ($i=1,\ldots, p$). Неоднозначностью этой функции можно воспользоваться для получения возможно более простой схемы обратных связей в автомате.

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

Построением схемы обратных связей процесс структурного синтеза исходного автомата A еще не заканчивается. Необходимо еще обеспечить в каждый момент времени t образование требуемого законом функционирования авто-

мата A структурного выходного сигнала $y(t)$. В зависимости от того, является ли заданный автомат A автоматом Мили или Мура, образование его выходного сигнала будет определяться либо законом $y(t)=\lambda(v(t), x(t))$, либо законом $y(t)=\lambda(v(t))$ (см. § 1).

И в том, и в другом случае необходимый структурный выходной сигнал может быть обеспечен комбинационной схемой, которую мы будем называть *схемой выходов* исходного автомата A. В случае автоматов Мили эта схема реализует векторную функцию $y=\lambda(v,x)$, а в случае автоматов Мура—векторную функцию $y=\lambda(v)$. В обоих случаях схема выходов автомата может быть получена в результате композиции заданных логических элементов в силу

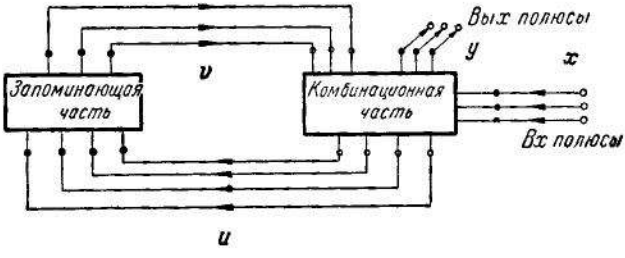

Рис. 9.

предположения о функциональной полноте системы этих элементов. Тем самым теорема 2.7 полностью доказана.

Заметим еще, что при структурном синтезе автомата обе построенные комбинационные схемы (схема обратных связей и схема выходов) обычно объединяются в одну общую комбинационную схему, называемую *комбинационной частью автомата*,— в противовес его *запоминающей части*, представляющей объединение (см. § 1) всех запоминающих элементов автомата. С помощью такого объединения часто оказываются возможными дальнейшие упрощения схемы.

Структурная схема всякого автомата, синтезированного в соответствии с каноническим методом структурного синтеза, имеет вид, изображенный на рис. 9.

Нетрудно видеть, что эта структурная схема будет корректно построенной или правильной, если корректно

построена или соответственно правильна его комбинационная часть (вместе с входными и выходными полюсами).

Таким образом, канонический метод синтеза полностью сводит все вопросы, возникающие при синтезе произвольных конечных автоматов, к соответствующим вопросам, относящимся к случаю автоматов без памяти.

В качестве примера рассмотрим применение канонического метода структурного синтеза к синтезу схемы автомата Мили A (в структурном алфавите (1, 2, 3)), заданного таблицами переходов и выходов III. 1 и III.2.

Таблица III.1

	a_1	a_2	a_3	a_4	a_5
1	a_2	a_1	—	a_3	—
2	a_5	a_2	a_1	a_5	—
3	a_4	—	a_3	—	a_4

Таблица III.2

	a_1	a_2	a_3	a_4	a_5
1	(1,1)	(2,2)	—	(1,3)	—
2	(3,1)	(3,3)	(2,1)	(3,2)	—
3	(3,3)	(1,1)	(2,2)	—	(2,3

В качестве элемента памяти выберем автомат Мура B, задаваемый таблицей переходов III.3.

Таблица III.3

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

Принимая, что отметка состояния i в автомате B есть также i (i=1, 2, 3), мы получим, очевидно, автомат Мура с полной системой переходов и с полной системой выходов.

Для кодирования состояний автомата A в структурном алфавите достаточно взять два элемента памяти, поскольку $3^2 > 5$. Систему кодирования состояний автомата A

выбираем произвольным образом, например, так:

$$a_1=(1,1),\quad a_2=(2,2),\quad a_3=(3,3),\quad a_4=(1,2),$$
$$a_5=(1,3).$$

После этого таблица переходов автомата A представится в виде табл. III. 4.

Выпишем таблицу входов элемента памяти B (см. табл. III.5).

Таблица III.4

	(1,1)	(2,2)	(3,3)	(1,2)	(1,3)
1	(2,2)	(1,1)	—	(3,3)	—
2	(1,3)	(2,2)	(1,1)	(1,3)	—
3	(1,2)	—	(3,3)	—	(1,2)

Таблица III.5

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

Используя табл. III.5 и таблицу переходов автомата A (табл. III.1), легко построить (векторную) функцию возбуждения автомата A. Обозначим через $\boldsymbol{x}=(x)$ внешний структурный входной сигнал автомата A; через $\boldsymbol{v}=(v_1,v_2)$ — структурный выходной сигнал этого автомата, а через $\boldsymbol{u}=(u_1,u_2)$ — структурный входной сигнал его памяти. Для задания функции возбуждения автомата A нужно задать зависимость входных сигналов u_1 и u_2 от сигналов x, v_1, v_2.

Для задания функций возбуждения здесь и дальше мы будем строить таблицу, выписывая явно значения компонент функции возбуждения на всех наборах элементарных сигналов, на которых эта функция задана. С целью сокращения места таблицу для функции возбуждения будем объединять с таблицей для (векторной) функции выходов автомата, задающей компоненты внешнего структурного выходного сигнала $\boldsymbol{y}=(y_1,y_2)$ как функции элементарных сигналов x, v_1, v_2. Наборы элементарных сигналов x, v_1, v_2, на которых не определены ни функция возбуждения, ни функция выходов, мы не будем включать в таблицу.

Получаемую таким образом объединенную таблицу для функций возбуждения и выходов автомата мы условимся

называть *функциональной таблицей* данного автомата. Она является исходным пунктом для синтеза соответствующей комбинационной схемы. В рассматриваемом случае из таблиц переходов и выходов автомата A (табл. III.1 и III.2) и таблицы входов элемента памяти B (табл. III.5) мы непосредственно получаем функциональную таблицу автомата A (табл. III. 6)

Таблица III.6

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

После построения этой таблицы задача структурного синтеза автомата A оказалась сведенной к задаче синтеза комбинационной схемы с тремя входными полюсами (x, v_1, v_2) и четырьмя выходными полюсами (u_1, u_2, y_1, y_2). Методы синтеза комбинационных схем мы будем рассматривать в последующих параграфах настоящей главы.

§ 3. Булевы функции

Из соображений технического характера при построении схем современных электронных цифровых автоматов в качестве структурного алфавита выбирается обычно алфавит, состоящий из двух элементов. Такой алфавит принято называть *двоичным*, а его буквы отождествлять с цифрами 0 и 1.

Употребление двоичного алфавита в качестве структурного алфавита в цифровых автоматах не следует путать с употреблением двоичной системы счисления для пред-

ставления чисел в таких автоматах. Как известно, с двоичным структурным алфавитом уживается не только двоичная, но также десятичная и любая другая система представления чисел в автоматах.

В современных электронных цифровых автоматах наиболее часто употребляются такие представления элементарных сигналов, составляющих двоичный структурный алфавит: 1) наличие импульса электрического тока и отсутствие импульса электрического тока, 2) высокий электрический потенциал и низкий электрический потенциал, 3) электрический ток большой силы и электрический ток малой силы, 4) импульсы различной (положительной и отрицательной) полярности. В случае 1) наличие импульса обычно принимается за единицу, а его отсутствие — за нуль. В других случаях возможны различные способы отождествления реальных элементарных сигналов с нулем и единицей. Заметим, что в первом из рассмотренных случаев имеет место естественное разделение элементарных сигналов, а отсутствие импульса служит естественным нулевым сигналом (см. § 1).

При синтезе комбинационных схем в двоичном структурном алфавите условия работы таких схем задаются с помощью функций, принимающих лишь два значения 0 и 1, и зависящих от переменных, каждая из которых также может принимать лишь два значения 0 и 1.

Условимся о том, что в дальнейшем переменные, способные принимать лишь два значения: 0 и 1, будут называться *двоичными* или *булевыми переменными*.

Функции от любого конечного числа двоичных переменных, также способные принимать лишь два значения — 0 и 1, принято называть *переключательными*, или *булевыми*, *функциями*. Это название функций связано с тем, что реализующие такие функции комбинационные схемы осуществляют переключения выходных каналов, подавая на них сигналы то одного, то другого вида. Название «булевы» возникло в связи с использованием функций рассматриваемого типа в *алгебре логики*, начало которой было положено трудами английского ученого 19 в. Дж. Б у л я.

Понятие переключательной функции иногда трактуется более широко. А именно под переключательной функ-

цией можно понимать любую функцию от конечного числа аргументов, которая вместе со всеми своими аргументами принимает значения из произвольного конечного множества значений. В то же время термин «булева» употребляется исключительно для обозначения функций двоичных переменных, то есть переменных, могущих принимать лишь два значения. Поэтому в дальнейшем изложении мы будем называть эти функции булевыми, а не переключательными, оставляя термин «переключательные» для более широкого класса функций.

Областью определения булевой функции от n переменных служит совокупность всевозможных n-мерных упорядоченных наборов (векторов размерности n), компонентами которых являются буквы двоичного алфавита (двоичные цифры) 0 и 1. Эти наборы называются *кортежами*, а иногда, используя геометрическую терминологию, также *точками*. Последнее название связано с тем, что имеется естественная возможность отождествить различные n-мерные (булевы) наборы с вершинами единичного n-мерного куба, одна из вершин которого совмещена с началом координат.

В дальнейшем изложении мы почти всегда будем иметь дело с наборами, компонентами которых являются двоичные цифры 0 и 1. Поэтому обычно мы не будем специально оговаривать это обстоятельство. В случае же необходимости подчеркнуть, что рассматриваются именно наборы с двоичными, а не с какими-нибудь иными компонентами, они будут называться *булевыми*, или *двоичными*, наборами.

Для любого $n=1, 2,\ldots$ среди n-мерных двоичных наборов можно ввести естественную (так называемую *лексикографическую*) упорядоченность. Для этой цели заметим прежде всего, что любой двоичный набор можно рассматривать как представление некоторого целого неотрицательного числа в двоичной системе счисления. Например, набор (1 1 1 0) представляет число $0+1\cdot 2+1\cdot 2^2+1\cdot 2^3=14$, а набор (0 1 0 1)—число $1+0\cdot 2+1\cdot 2^2+0\cdot 2^3=5$. Число, которое представляется данным набором (рассматриваемым как двоичный код), мы условимся называть *номером* этого набора.

Естественную упорядоченность n-мерных (двоичных) наборов мы получим, расположив эти наборы в порядке

возрастания их номеров. Первым в таком расположении будет *нулевой набор*, все компоненты которого являются нулями, последним — *единичный набор*, все компоненты которого являются единицами. Единичный набор следует отличать от набора, имеющего номер единица. У этого последнего набора лишь последняя компонента равна 1, а все остальные компоненты являются нулями.

Двоичный набор полностью определяется своим номером и своей размерностью. Поэтому, когда размерность наборов задана, их можно отождествлять с их номерами. Набор, имеющий номер i, мы будем называть *i-м набором*. Наборы размерности n нумеруются целыми числами от 0 до 2^n-1. Отсюда, в частности, вытекает следующее предложение.

3.1. *Имеется точно 2^n двоичных n-мерных наборов* ($n=1, 2, \ldots$).

Нетрудно также подсчитать число различных булевых функций от n переменных. Для каждого набора значений переменных, независимо от значений функции на других наборах, мы можем выбрать в качестве значения функции 0 или 1. Следовательно, прибавление каждого нового набора в область определения булевой функции увеличивает число различных булевых функций ровно в два раза.

На одном наборе можно определить две различные булевы функции, на двух наборах — 2^2 различные булевы функции, и т. д. Продолжая подобным образом и учитывая результат, сформулированный в теореме 3.1, мы придем к предложению

3.2. *Имеется точно 2^{2^n} различных булевых функций от n переменных* ($n=1,2, \ldots$).

Следует отметить, что здесь и далее к числу функций от n переменных относятся и такие функции $f(x_1, x_2, \ldots, x_n)$, которые в действительности не зависят от тех или иных переменных x_i. Таким образом, всякая функция от меньшего числа переменных автоматически включается в число функций любого большего числа переменных. В частности, *функции-константы* (тождественный нуль и тождественная единица), которые естественно рассматривать как функции от нуля переменных, можно рассматривать

также как функции одного, двух, трех и, вообще, любого (конечного) числа переменных.

Условимся называть *невырожденными функциями от n переменных* такие функции, которые существенно зависят от всех этих переменных. Функции же n переменных, сводящиеся к функциям от меньшего числа переменных, назовем *вырожденными*.

Булевы функции могут задаваться с помощью таблиц. В левой части таблицы, задающей булеву функцию, в порядке естественной упорядоченности сверху вниз выписываются наборы значений переменных, на которых определены данные функции. В правой части таблицы против каждого набора значений переменных выписываются соответствующие значения функции.

Наряду с функциями, значения которых заданы на всех наборах значений переменных, оказывается целесообразным рассматривать также и неполностью определенные булевы функции. В структурной теории автоматов неполное задание булевой функции означает возможность выбора значений функции на тех наборах, на которых она не была первоначально задана, произвольным образом. Таким образом, неполностью заданная булева функция будет рассматриваться фактически как целое семейство всюду определенных булевых функций, некоторые из значений которых могут варьироваться. При табличном задании булевых функций неопределенные значения функций принято обозначать черточкой или волной $\sim$.

В теории булевых функций особое значение имеют функции одного и двух переменных, к систематическому изучению которых мы сейчас переходим. Имеется всего $2^{2^1}=4$ различные булевы функции одной переменной. Эти функции можно задать сводной таблицей (табл. III.7).

Из построенных четырех функций функции g_1 и g_4 представляют собою константы 0 и 1. Функция g_2 повторяет значения переменной x и потому просто совпадает с ней: $g_2=x$. Единственной нетривиальной функцией является функция g_3. Мы будем называть эту функцию *отрицанием* переменной x и обозначать через $\bar{x}$: $g_3=\bar{x}$. Иногда

функцию g_3 называют также *инверсией*, а для ее обозначения используют символ x'.

Таблица III.7

x \ f	g_1	g_2	g_3	g_4
0	0	0	1	1
1	0	1	0	1

Число булевых функций от двух переменных равно, согласно теореме 3.2, $2^{2^2}=2^4=16$. Выпишем сводную таблицу всех этих функций (табл. III.8).

Таблица III. 8

x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Из выписанных функций шесть функций, а именно функции f_1, f_4, f_6, f_{11}, f_{13}, f_{16}, будут вырожденными. Действительно, легко видеть, что $f_1=0$, $f_4=x$, $f_6=y$, $f_{11}=\overline{y}$, $f_{13}=\overline{x}$, $f_{16}=1$. Таким образом, все названные шесть функций существенно зависят не более чем от одной переменной каждая. Остальные 10 функций будут невырожденными. Мы введем для них специальные названия и обозначения.

Функция f_8 носит название *конъюнкции*, или *произведения*, или *функции совпадения*, или *логического* «и». Для ее обозначения мы будем пользоваться знаком умножения, который, как и в обычной алгебре, применительно к буквенным выражениям мы будем опускать: $f_8=x\cdot y=xy$.

Если значения переменных x и y функции f_2 понимать как обычные числа 0 и 1, то значение функции f_2 можно находить, как легко проверить, с помощью обычного перемножения соответствующих значений переменных x и y.

Функция f_7 носит название *функции неравнозначности*, или *суммы по модулю два*; для ее обозначения мы будем употреблять обычный знак суммы: $f_7 = x+y$. Легко проверить, что, понимая под 0 и 1 обычные числа, мы можем находить значения функции f_7 в результате суммирования по модулю два соответствующих значений переменных x и y. Часто суммирование по модулю два мы будем называть в дальнейшем просто суммированием, поскольку в других смыслах термины «суммирование» и «сумма» употребляться в настоящей главе не будут.

Функция f_8 носит название *дизъюнкции*, или *функции разделения*, или *логического* «или». Для ее обозначения мы будем употреблять обычный знак дизъюнкции, применяющийся в математической логике: $f_8 = x \vee y$.

Функцию f_9 мы будем называть *отрицанием дизъюнкции* и обозначать через $\overline{x \vee y}$. Смысл этого обозначения станет ясным после введения понятия о суперпозиции булевых функций.

Функция f_{10} носит название *функции равнозначности*, или *логической эквивалентности*, или просто *эквивалентности* и обозначается через $x \equiv y$.

Функции f_{12} и f_{14} носят название *импликации*. Для их обозначения употребляют стрелку, соединяющую между собой переменные x и y, расположенные в определенном порядке:

$$f_{12} = y \to x. \quad f_{14} = x \to y.$$

Функция f_{15} носит название *штриха Шеффера*, или *отрицания конъюнкции*. Для ее обозначения используется обычно вертикальная черточка, разделяющая переменные: $f_{15} = x|y$. Можно обозначать эту функцию также через $\overline{xy}$.

Оставшиеся без названий невырожденные функции f_3 и f_5 можно называть *отрицаниями импликации*. Иногда они называются также *функциями запрета*. Их можно обозначать следующим образом: $f_3 = x\overline{y}$, $f_5 = \overline{x}y$.

Значение функций одного и двух переменных в общей теории булевых функций состоит в том, что из них может быть построена любая булева функция. Средством для такого построения является суперпозиция булевых функций или, что то же самое, подстановка одних булевых функций вместо аргументов в другие булевы функции. Возможность такой подстановки обусловливается тем, что, в силу определения, области значений булевых функций и области значений их аргументов совпадают между собой.

Используя суперпозицию, мы можем аргументами каждой из рассмотренных выше функций одного и двух переменных считать произвольные булевы функции. Тем самым булевы функции одного и двух переменных (как, впрочем, и булевы функции любого числа переменных) могут рассматриваться не как функции, а как операции на множестве всех булевых функций. В частности, отрицание может рассматриваться как *монарная* (одноместная) операция, а конъюнкция, дизъюнкция и суммирование по модулю два — как *бинарные* (двуместные) операции.

С помощью этих операций могут быть построены различные алгебры булевых функций. Две наиболее замечательные алгебры такого рода будут рассмотрены ниже. А сейчас заметим лишь, что некоторые из введенных выше обозначений для функций двух переменных используют явным образом понятие суперпозиции функций. Так, отрицание дизъюнкции (функция f_9) есть не что иное, как результат подстановки дизъюнкции переменных x и y вместо переменной x в функцию g_3 (в отрицание переменной x). В предложенном выше обозначении для функции $\overline{x \vee y}$ явным образом выполнена указанная подстановка (суперпозиция функций).

Аналогично обстоит дело с обозначениями функций f_3 и f_5, указывающими на возможность представления этих функций в виде суперпозиции отрицания и умножения. Нетрудно видеть, что при таком истолковании выражений $x\overline{y}$ и $\overline{xy}$ (рассматривавшихся выше просто как символы, введенные по определению) их значения действительно совпадают с требуемыми значениями функций f_3 и f_5 (ср. табл. III. 8).

На этих простых примерах мы видим уже, каким образом происходит представление одних булевых функций через суперпозицию других булевых функций. В последующих параграфах способы такого представления будут разработаны в гораздо более общем виде.

Следует заметить, что с ростом числа аргументов происходит весьма быстрый рост числа булевых функций. Так, имеется $2^{2^3}=2^8=256$ различных булевых функций от трех переменных. Число булевых функций от пяти переменных превышает 4 миллиарда. При каждом увеличении числа переменных на единицу происходит (как нетрудно убедиться с помощью теоремы 3.2) возведение в квадрат числа различных булевых функций, полученного на предыдущем этапе.

Быстрый рост числа булевых функций с ростом числа переменных практически исключает возможность их непосредственной табуляризации[1]) и классификации. Задача классификации несколько упрощается в случае объединения некоторых булевых функций в классы и типы.

Говорят, что две булевы функции $f_1(x_1, x_2, \ldots, x_n)$ *и* $f_2(y_1, y_2, \ldots, y_n)$ *относятся к одному и тому же классу, если после некоторой взаимно однозначной подстановки переменных:* x_1 *на место* y_{i_1}, x_2 *на место* y_{i_2}, ... *и* x_n *на место* y_{i_n}, *одна из этих функций совпадает со второй. Говорят, что функции* f_1 *и* f_2 *относятся к одному и тому же типу, если одна из них переходит в другую после выполнения взаимно однозначной подстановки переменных с инверсиями:* x_1 *на место* $\tilde{y}_{i_1}$, x_2 *на место* $\tilde{y}_{i_2}$, ... *и* x_n *на место* $\tilde{y}_{i_n}$, *где* y_{i_k} *означает либо* y_{i_k}, *либо* $\tilde{y}_{i_k}$ $(k=1, \ldots, n)$.

К. Шенноном[2]) была доказана следующая теорема.

3.3. *Число различных классов булевых функций от* n *переменных равно* $\frac{2^{2^n}}{n!}(1+\varepsilon_n)$, *где* ε_n — *положительная величина, стремящаяся к нулю при* n, *стремящемся к* ∞.

---

[1]) Под табуляризацией здесь понимается составление таблиц, включающих в себя все булевы функции с любым данным числом аргументов.

[2]) C. E. Shannon, The synthesis of two-terminal switching circuits. Bell. System Tech. J., v. 28, 1949, p. 59—98.

Г. Н. Поваров[1]) использовал методы К. Шеннона для доказательства следующего предложения.

3.4. *Число различных типов булевых функций от n переменных равно* $\frac{2^{2^n}}{2^n n!}(1+\delta_n)$, *где* δ_n — *положительная величина, стремящаяся к нулю при* n, *стремящемся к* ∞.

Точное вычисление величин ε_n и δ_n в теоремах 3.3 и 3.4 произведено лишь для малых значений числа переменных. В табл. III.9 указано число булевых функций, число классов и число типов функций для $n=1, 2, 3, 4$[2]).

Таблица III.9

Число переменных в булевых функциях	Число булевых функций	Число классов булевых функций	Число типов булевых функций
1	4	4	3
2	16	12	6
3	256	80	22
4	65536	3984	402

В одном из следующих параграфов мы выпишем явно все типы булевых функций от трех переменных. Что же касается булевых функций от двух переменных, то существующие типы таких функций нетрудно получить непосредственно, рассматривая таблицы всех 16 булевых функций от двух переменных f_1—f_{16}. Первые два типа составляют константы 0 и 1. К третьему типу относятся функции f_4, f_6, f_{11}, f_{13}, являющиеся, по существу, функциями одного переменного. К четвертому типу относится функция $f_2=xy$, а также функции f_3, f_5, и f_9. Пятый тип составляют функции $f_7=x+y$ и f_{10}, а шестой тип — функции $f_8=x\vee y$, f_{12}, f_{14}, f_{15}.

---

[1]) Г. Н. Поваров, Математическая теория синтеза контактных $(1, k)$-полюсников, ДАН СССР, т. 100, № 5, 1955, стр. 909—912.

[2]) См. табл. 12 в статье С. В. Яблонского «Функциональные построения в k-значной логике». Тр. Матем. ин-та им. В. А. Стеклова, т. 51, 1958, стр. 56.

§ 4. Две замечательные алгебры булевых функций

При построении теории синтеза комбинационных схем в двоичном структурном алфавите большую роль играют две системы операций, которые можно ввести на множестве всех булевых функций. Первая система включает три операции: отрицание, умножение и дизъюнкцию, а вторая — две операции: умножение и сложение по модулю два.

Множество булевых функций, рассматриваемое вместе с операциями отрицания, умножения и дизъюнкции, называют обычно булевой алгеброй. Множество же булевых функций, рассматриваемое вместе с операциями умножения и сложения (по модулю два) мы будем называть алгеброй Жегалкина.

Как в булевой алгебре, так и в алгебре Жегалкина[1]) мы будем рассматривать различные выражения, составленные с помощью операций алгебры из констант 0 и 1 и из букв (переменных) $x, y, z, \ldots$, под которыми мы будем понимать произвольные булевы функции (в частности, обычные двоичные переменные). Условимся относительно порядка выполнения действий: в булевой алгебре при отсутствии в выражении скобок первыми должны выполняться операции отрицания, затем — операции умножения и, наконец, — операции дизъюнкции; в алгебре Жегалкина первыми выполняются операции умножения, а затем — операции сложения. Наличие в выражении скобок изменяет обычный порядок действий: в первую очередь должны производиться операции внутри скобок. В булевой алгебре при отрицании сложных выражений часто опускают скобки. Роль скобок при этом играет сам символ отрицания. Например, вместо $(\overline{xy})$ обычно пишут просто $\overline{xy}$. При этом, разумеется, имеет силу все сказанное относительно изменения порядка действий при наличии скобок.

Любое выражение булевой алгебры или алгебры Жегалкина представляет собою некоторую булеву функцию и в свою очередь может быть обозначено одной буквой.

---

[1]) Названной по имени предложившего ее советского ученого П. И. Жегалкина.

Два выражения мы будем считать *равными*, если они представляют одинаковые булевы функции входящих в них переменных[1]). Нетрудно видеть, что определяемые равенства будут *тождественными* в том смысле, что подстановка вместо любой буквы одновременно в левой и в правой частях равенства одного и того же выражения соответствующей алгебры не нарушает равенства. Нашей ближайшей целью является установление тождественных соотношений, имеющих место в двух введенных нами замечательных алгебрах. Средством для установления справедливости выписываемых ниже тождеств служит, в основном, проверка с помощью перебора всех значений переменных, входящих в данное тождество. Выпишем сначала основные тождества для булевой алгебры. Непосредственно из определения отрицания (см. § 3) вытекает справедливость следующего тождества, называемого обычно *правилом двойного отрицания*:

$$\overline{\overline{x}} = x. \tag{1}$$

Из определения дизъюнкции и умножения вытекают *законы коммутативности* для дизъюнкции и умножения

$$x \vee y = y \vee x, \tag{2}$$

$$xy = yx. \tag{3}$$

Непосредственно проверяется справедливость *законов ассоциативности* для дизъюнкции и для умножения:

$$x \vee (y \vee z) = (x \vee y) \vee z, \tag{4}$$

$$x(yz) = (xy)z. \tag{5}$$

В самом деле, из определения дизъюнкции (см. § 3) следует, что левая часть тождества (4) равна нулю тогда и только тогда, когда все переменные x, y и z одновременно обращаются в нуль. Поскольку то же самое имеет место и для правой части тождества (4), то справедливость этого тождества доказана. Аналогичным образом доказывается справедливость тождества (5). Различие состоит лишь в том, что здесь как левая, так и правая части обращаются

[1]) Более точно, равные функции должны принимать одинаковые значения на любом наборе значений переменных, входящих в представляющие эти функции булевы выражения.

в единицу (а не в нуль) при единственном наборе значений переменных ($x=1$, $y=1$, $z=1$).

Наличие законов ассоциативности позволяет записывать сложные дизъюнкции и сложные произведения без всяких скобок, как выражения вида $x_1 \vee x_2 \vee \ldots \vee x_n$ или соответственно вида $x_1\, x_2\, \ldots\, x_n$.

Непосредственной проверкой с помощью подстановки различных значений переменных устанавливается также справедливость *первого* и *второго законов дистрибутивности* для булевой алгебры:

$$x\,(y \vee z) = xy \vee xz, \tag{6}$$

$$x \vee yz = (x \vee y)\,(x \vee z). \tag{7}$$

Действительно, легко проверить, что в тождестве (6) левая и правая части равны, каждая, единице тогда и только тогда, когда переменная x и хотя бы одна из переменных y или z принимают значения, равные единице. Аналогично, в тождестве (7) как левая, так и правая части обращаются в нуль тогда и только тогда, когда переменная x и хотя бы одна из переменных y или z принимают значения, равные нулю.

Следующими важными тождествами являются так называемые *правила де Моргана*[1]):

$$\overline{x \vee y} = \bar{x}\,\bar{y}, \tag{8}$$

$$\overline{xy} = \bar{x} \vee \bar{y}. \tag{9}$$

В первом тождестве де Моргана как левая, так и правая части обращаются в единицу тогда и только тогда, когда как x, так и y равны нулю. Аналогично во втором тождестве правая и левая части обращаются в нуль при единичном наборе значений переменных.

Для операций с константами нуль и единица имеют силу следующие очевидные правила:

$$\bar{0} = 1, \qquad \bar{1} = 0, \tag{10}$$

$$x \cdot 1 = x, \qquad x \cdot 0 = 0, \tag{11}$$

$$x \vee 0 = x, \qquad x \vee 1 = 1. \tag{12}$$

---

[1]) Де Морган — английский математик (19 в.), один из создателей алгебры логики.

Наконец, легко проверяется справедливость следующих двух важных соотношений:

$$x \vee \bar{x} = 1, \tag{13}$$

$$x\bar{x} = 0. \tag{14}$$

Из выписанных тождеств булевой алгебры, которые мы будем считать основными, легко выводятся новые тождества. Приведем некоторые из таких тождеств, особенно полезные при проведении различных преобразований в булевой алгебре.

Для того чтобы подчеркнуть, что мы имеем дело не с основными, а с производными тождествами, будем употреблять в них вместо малых латинских букв большие буквы. Следует подчеркнуть, что это различие является чисто номинальным, поскольку как в первом, так и во втором случае входящие в формулы буквы можно считать произвольными выражениями булевой алгебры. Заметим также, что тождества (10), (11), (12) позволяют исключить из любого выражения булевой алгебры входящие в него константы, если только само это выражение не сводится к константе. Поэтому в дальнейшем, говоря о произвольных выражениях булевой алгебры, можно считать, что в это выражение не входят в явном виде константы 0 и 1, случай же, когда само выражение сводится к константе, оговаривать особо.

Прежде всего укажем на так называемые *законы поглощения* в булевой алгебре:

$$A \vee AB = A, \tag{15}$$

$$A(A \vee B) = A. \tag{16}$$

Первый закон поглощения может быть доказан с помощью выписывания следующей цепочки равенств:

$$A \vee AB = A \cdot 1 \vee AB = A(1 \vee B) = A(B \vee 1) = A \cdot 1 = A.$$

Проведенные преобразования последовательно используют тождества (11), (6), (2), (12), (11).

Аналогичным образом доказывается второй закон поглощения:

$$A(A \vee B) = (A \vee 0)(A \vee B) = A \vee 0 \cdot B = A \vee B \cdot 0 = A \vee 0 = A.$$

При этом последовательно используются тождества (12), (7), (3), (11), (12).

Полагая в тождестве (15) B равным единице, а в тождестве (16) B равным нулю, мы приходим к следующим важным частным случаям этих тождеств, которые будем называть *законами идемпотентности* дизъюнкции и умножения:

$$A \vee A = A, \tag{17}$$

$$AA = A. \tag{18}$$

Заметим, что частными случаями тождеств (15) и (16) являются также вторые из тождеств (11) и (12).

Рассмотрим выражение $A \vee \overline{A}B$. Используя второй закон дистрибутивности, его можно представить в виде $(A \vee \overline{A})(A \vee B)$, после чего с помощью тождеств (13), (3) и (11) мы приведем его к виду $A \vee B$. Следовательно, имеет место тождество

$$A \vee \overline{A}B = A \vee B. \tag{19}$$

Заметим также, что, комбинируя законы коммутативности для дизъюнкции и умножения с тождествами (6), (7), (11), (12), (13), (14), (19), мы придем к новым тождествам, отличающимся от старых порядком расположения членов. Мы условимся в дальнейшем при ссылках на то или иное из этих тождеств иметь в виду возможность произвольных перестановок входящих в них дизъюнктивных членов или сомножителей.

С помощью математической индукции из тождеств (11), (12), (17), (18) легко выводятся следующие предложения:

4.1. *Выбрасывая из произвольной дизъюнкции дизъюнктивные члены, равные нулю, мы не изменим величины этой дизъюнкции.*

4.2. *Если в дизъюнкции хотя бы один из членов равен единице, то вся дизъюнкция равна единице.*

4.3. *Выбрасывая из произвольного произведения все сомножители, равные единице, мы не изменим величины этого произведения.*

4.4. *Если в произведении хотя бы один сомножитель равен нулю, то все произведение обращается в нуль.*

4.5. *Дизъюнкция или произведение любого числа одинаковых членов A равняется A.*

Правила де Моргана (8) и (9) также поддаются обобщению с помощью применения математической индукции. В результате такого обобщения мы придем к следующему предложению.

4.6. *Если $A(a_1,\ldots, a_n)$ — произвольное выражение булевой алгебры (построенное из выражений $a_1,\ldots, a_n$ с помощью применения операций отрицания, дизъюнкции и умножения), то отрицание этого выражения равняется $B(\overline{a_1},\ldots, \overline{a_n})$, где выражение $B(a_1,\ldots, a_n)$ получается из выражения $A(a_1, \ldots, a_n)$ с помощью замены всех умножений на дизъюнкции, а всех дизъюнкций — на умножение, при условии сохранения всех имевшихся ранее в выражении $A(a_1, \ldots, a_n)$ отрицаний.*

В самом деле, если выражение $A(a_1,\ldots, a_n)$ представляет собою дизъюнкцию каких-то членов: $A_1 \vee A_2 \vee \ldots \vee A_m$, то последовательное применение тождества (8), приведет нас к соотношению

$$\overline{A_1 \vee A_2 \vee \ldots \vee A_m} = \overline{A_1}\,\overline{A_2}\ldots\overline{A_m}. \tag{20}$$

Тем самым осуществляется сведéние проблемы нахождения отрицания данного выражения к проблеме нахождения отрицания выражений с меньшим числом булевых операций.

В случае, когда исходное выражение представляет собою произведение, аналогичное сведéние осуществляется с помощью тождества

$$\overline{A_1 A_2 \ldots A_n} = \overline{A_1} \vee \overline{A_2} \vee \ldots \vee \overline{A_n}. \tag{21}$$

Это тождество получается в результате последовательного применения тождества (9).

Наконец, в случае, когда исходное выражение $A = A(a_1,\ldots,a_n)$ имеет в качестве последней операции отрицание (то есть, если A имеет вид $\overline{C}$), то отрицание выражения A имеет вид $\overline{\overline{C}}$, то есть совпадает (в силу тождества (1)) с выражением C, которое опять-таки имеет меньшее, по сравнению с выражением A, число булевых операций.

Поскольку для выражений, сводящихся к одному из выражений $a_1, \ldots, a_n$, справедливость утверждения 4.6 очевидна, то тем самым его доказательство завершено.

В качестве примера применения теоремы 4.6 рассмотрим отрицание выражения $B \underset{Df}{=} \overline{A_1} \vee \overline{A_2\,(A_3 \vee \overline{A_4})}$[1]). Согласно теореме 4.6, мы будем иметь: $B = \overline{\overline{A}_1}\,(\overline{A_2 \vee \overline{A}_3 A_4})$. Разумеется, двойное отрицание, в силу тождества (1), можно при этом не писать. Вообще нетрудно убедиться в справедливости следующего предложения.

4.7. *Если к некоторому выражению A булевой алгебры применена более чем одна операция отрицания $B \underset{Df}{=} \overline{\overline{\vdots}\,A}$, то, не изменяя значений, принимаемых выражением B, можно исключить любое четное число этих операций.*

В силу предложения 4.7 можно всегда предполагать, что при построении выражений булевой алгебры ни к какой части этого выражения не применяется более чем одна операция отрицания (разумеется, при этом не исключается, что в н у т р и отрицаемого выражения находятся и другие отрицания).

Если теперь $A = A\,(a_1, \ldots, a_n)$ — произвольное выражение булевой алгебры, то, применяя предложение 4.6 ко всем отрицаниям, входящим в выражение A, а затем используя теорему 4.7, мы придем к следующему предложению.

4.8. *Любое выражение $A(a_1, \ldots, a_n)$ булевой алгебры с помощью тождественных соотношений, существующих в этой алгебре, может быть приведено к выражению B, построенному из переменных $a_1, \ldots, a_n$ и их отрицаний с помощью одних лишь умножений и дизъюнкций.*

Назовем выражение B, указанное в теореме 4.8, *безинверсной формой* исходного выражения A. Теорема 4.8 утверждает, что любое выражение булевой алгебры может быть приведено к безинверсной форме.

Отметим одно обобщение законов дистрибутивности, получаемое путем последовательного применения тождеств (6) и (7) в сочетании с законами коммутативности:

4.9. *Произведение $A_1 A_2 \ldots A_n$, каждый сомножитель которого является дизъюнкцией некоторого числа членов: $A_i = a_{i_1} \vee a_{i_2} \vee \ldots \vee a_{i_{k_i}}$ $(i = 1, \ldots, n)$, можно преобразовать в рав-*

[1]) Знак «$\underset{Df}{=}$» читается «есть по определению», «равно по определению».

ную ему дизъюнкцию произведений $a_{1j_1}, a_{2j_2}, \ldots, a_{nj_n}$, *взятых для всевозможных наборов значений* $j_1, j_2, \ldots, j_n$. *Дизъюнкцию* $B_1 \vee B_2 \vee \ldots \vee B_m$, *каждый член которой представляет собой произведение некоторого числа сомножителей:* $B_i = b_{i_1} b_{i_2} \ldots b_{i_{r_i}}$ $(i = 1, \ldots, m)$, *можно преобразовать в равное ей произведение дизъюнкций* $b_{1j_1} \vee b_{2j_2} \vee \ldots \vee b_{mj_m}$, *взятых для всевозможных наборов значений* $j_1, j_2, \ldots, j_m$.

Например, выражение $(a_1 \vee a_2)(\bar{a}_1 \vee a_3)$ равняется выражению $a_1\bar{a}_1 \vee a_1a_3 \vee a_2\bar{a}_1 \vee a_2a_3$, а выражение $ab \vee \bar{b}c$ равняется выражению $(a \vee \bar{b})(a \vee c)(b \vee \bar{b})(b \vee c)$.

Перейдем теперь к установлению системы основных тождеств в алгебре Жегалкина.

Непосредственной проверкой устанавливается, что для сложения по модулю два, как и для умножения, имеют место *законы коммутативности и ассоциативности*:

$$x + y = y + x, \quad xy = yx, \tag{22}$$

$$x + (y + z) = (x + y) + z, \quad x(yz) = (xy)z. \tag{23}$$

Имеет силу *дистрибутивный закон для умножения по отношению к сложению*:

$$x(y + z) = xy + xz. \tag{24}$$

В самом деле, как левая, так и правая части этого равенства равны единице тогда и только тогда, когда x равен 1 и в точности одна из переменных y или z также равна 1. Заметим, что закон дистрибутивности для сложения по отношению к умножению уже не имеет силы.

Для операций с константами непосредственной проверкой легко устанавливаются следующие тождества:

$$x \cdot 1 = x, \quad x \cdot 0 = 0, \quad x + 0 = x. \tag{25}$$

Непосредственной проверкой устанавливается *закон приведения подобных членов при сложении*, а также и *закон идемпотентности для умножения*:

$$x + x = 0, \tag{26}$$

$$xx = x. \tag{27}$$

Выписанные тождества (22)—(27) мы будем называть *основными тождествами алгебры Жегалкина*.

Рассмотрим теперь взаимосвязь, существующую между операциями булевой алгебры и алгебры Жегалкина, для чего введем объединенную алгебру, использующую операции отрицания, дизъюнкции, умножения и сложения по модулю два.

Непосредственной проверкой (подстановкой различных возможных наборов значений переменных) устанавливаются следующие тождества:

$$\bar{x} = 1 + x, \tag{28}$$

$$x \vee y = x + y + xy, \tag{29}$$

$$x + y = \bar{x}y \vee x\bar{y} \tag{30}$$

Первые два из этих тождеств позволяют преобразовать любое выражение булевой алгебры в равное ему (представляющее ту же самую булеву функцию) выражение алгебры Жегалкина. Тождество (30) позволяет осуществить обратный переход — от выражения в алгебре Жегалкина к равному ему выражению булевой алгебры.

Рассмотрим примеры преобразований выражений булевой алгебры в равные им выражения алгебры Жегалкина, а также примеры преобразований выражений в алгебре Жегалкина в равные им выражения булевой алгебры.

Пример 1. Выражение $\overline{\bar{x} \vee xy}$ преобразовать в выражение алгебры Жегалкина и упростить полученное после преобразования выражение.

Решение. Используя тождества (28) и (29), мы получаем:

$$\overline{\bar{x} \vee xy} = 1 + (1 + x) + x(1 + y) + (1 + x)x(1 + y).$$

Тождества (22), (23), (24), (25) и (27) позволяют преобразовать полученное выражение в вид

$$1 + 1 + x + x + xy + x + x + xy + xy.$$

Поскольку $1 + 1 = 0$, $x + x = 0$, $xy + xy = 0$, то в силу тождества (26) мы имеем окончательно:

$$1 + 1 + x + x + xy + x + x + xy + xy = xy.$$

Пример 2. Выражение $(x + 1)y + (x + 1)$ преобразовать в выражение булевой алгебры и упростить полученное выражение.

Решение. Из соотношений (28) и (30) выводим:

$$(x+1)y+(x+1)=\bar{x}y+\bar{\bar{x}}=\overline{\bar{x}y}\cdot\bar{\bar{x}}\vee\bar{x}y\cdot\bar{\bar{x}}.$$

Пользуясь основными тождествами булевой алгебры, последнее выражение легко преобразовать в вид

$$\overline{\bar{x}\vee\bar{y}}\,\bar{\bar{x}}\vee\bar{x}yx=(x\vee\bar{y})\,\bar{\bar{x}}\vee 0=$$
$$=(x\vee\bar{y})\,\bar{\bar{x}}=x\bar{\bar{x}}\vee y\bar{\bar{x}}=0\vee\bar{y}\,\bar{\bar{x}}=\bar{\bar{x}}\,\bar{y}.$$

Заметим, что если в исходном выражении $(x+1)y+(x+1)$ множитель $x+1$ вынести за скобку, то получится выражение $(x+1)(y+1)$, которое на основе тождеств (22) и (28) сразу преобразуется в вид $\bar{x}\bar{y}$.

§ 5. Нормальные формы

Основной целью настоящего параграфа является установление некоторых специальных форм выражений в булевой алгебре и в алгебре Жегалкина, к которым можно свести любое выражение в этих алгебрах.

Начнем наши рассуждения с булевой алгебры, и прежде всего введем понятия *элементарной дизъюнкции* и *элементарного произведения*.

Элементарной дизъюнкцией называется выражение, представляющее собою дизъюнкцию любого конечного множества попарно различных между собою букв, над частью которых (быть может, пустой или, наоборот, содержащей все буквы) поставлены черточки (знаки отрицания).

К числу элементарных дизъюнкций мы будем относить также выражения, состоящие из одной буквы (с отрицанием или без отрицания), а также константу **нуль**. Считается, что в первом случае мы имеем дело с дизъюнкцией одного члена, во втором — с дизъюнкцией пустого множества членов. Дизъюнкцию пустого множества членов мы, по определению, будем считать всегда равной нулю.

В силу приведенного определения элементарными дизъюнкциями будут, например, выражения:

$$0,\ x_1,\ x\vee y,\ \bar{a}\vee b\vee c,\ \bar{x}\vee\bar{y}.$$

В то же время выражения:

$$x_1 \vee x_1 \vee x_2, \quad a \vee b \vee \bar{a}$$

не являются элементарными дизъюнкциями, поскольку в первое выражение дважды входит буква x_1, а во второе — дважды входит буква a, причем один раз с отрицанием, а второй раз — без отрицания.

Элементарным произведением называется выражение, представляющее собою произведение любого конечного множества попарно различных между собою букв, над частью которых (быть может, пустой или, наоборот, содержащей все буквы) поставлены черточки (знаки отрицания).

К числу элементарных произведений мы будем также относить выражения, состоящие из одной буквы (с отрицанием или без отрицания), а также константу единица. Для этой цели условимся произведения, состоящие из одного сомножителя, считать равными этому сомножителю; условимся также рассматривать в качестве элементарных произведения, состоящие из пустого множества сомножителей, и считать их, по определению, равными единице.

В силу приведенного определения к числу элементарных произведений относятся, например, выражения

$$1, \quad x, \quad xy, \quad \bar{a}x, \quad \bar{x}\,\bar{y}\,\bar{z}.$$

В то же время выражения

$$\overline{xy}, \quad yxy, \quad x\bar{x}$$

не являются элементарными произведениями, ибо в первом случае знак отрицания стоит над всем выражением (а не над составляющими его буквами), во втором и в третьем случаях в выражения дважды входят одинаковые буквы.

Элементарные произведения и элементарные дизъюнкции, отличающиеся только порядком расположения членов, мы в дальнейшем не будем различать.

Условимся буквы и их отрицания называть *первичными термами*. Нетрудно доказать следующее предложение.

5.1. *Дизъюнкция любого числа первичных термов равна либо единице, либо элементарной дизъюнкции. Произведение любого числа первичных термов равно либо нулю, либо элементарному произведению.*

В самом деле, если в некоторую дизъюнкцию первичных термов какая-либо буква входит вместе со своим отрицанием, то, в силу тождества (13) и предложения 4.2, вся дизъюнкция равна единице. В противном же случае, используя предложение 4.5, мы сможем привести рассмотренную дизъюнкцию к элементарной дизъюнкции. Аналогичное рассуждение нетрудно провести и относительно произведения первичных термов.

Введем теперь следующее важное определение.

5.2. *Дизъюнктивной нормальной формой называется дизъюнкция любого конечного множества попарно различных элементарных произведений. Конъюнктивной нормальной формой называется произведение любого конечного множества попарно различных элементарных дизъюнкций.*

Заметим, что в этом определении не исключается случай дизъюнкций и произведений пустого множества членов или множества, состоящего из одного-единственного члена. Таким образом, константа 0, например, может рассматриваться как дизъюнктивная нормальная форма, представляющая собою дизъюнкцию пустого множества элементарных произведений, либо как конъюнктивная нормальная форма, представляющая собою произведение, состоящее из одной элементарной дизъюнкции 0 — дизъюнкции пустого множества первичных термов. Аналогично константа 1 также может рассматриваться и как конъюнктивная, и как дизъюнктивная нормальная форма.

Справедлива следующая важная теорема.

5.3. *Существует единый конструктивный метод, позволяющий для любого выражения булевой алгебры найти равные ему дизъюнктивную и конъюнктивную нормальную форму.*

Требуемый конструктивный прием может быть построен следующим образом. На первом этапе с помощью метода, использованного при доказательстве предложения 4.8, исходное выражение $G=G(x_1,\ldots,x_n)$ преобразуется в равное ему выражение F, построенное из букв $x_1,\ldots,x_n$ и их отрицаний с помощью одних лишь умножений и дизъюнкций.

На втором этапе последовательно, шаг за шагом, применяется теорема 4.9. Предположим вначале, что нам требуется привести выражение к дизъюнктивной нормальной форме. Если выражение F не содержит дизъюнкций, то оно является, очевидно, произведением первичных термов и с помощью предложения 5.1 может быть либо сведено к нулю, либо к элементарному произведению, то есть к некоторому частному случаю дизъюнктивной нормальной формы.

В общем случае будем применять редукцию, представляя выражение F в виде дизъюнкции членов, каждый из которых содержит меньшее число дизъюнкций, чем исходное выражение F. Ясно, что, осуществляя подобную редукцию, мы через конечное число шагов придем к дизъюнкции выражений, не содержащих дизъюнкций; такие выражения, как было показано выше, сводятся либо к нулю, либо к элементарным произведениям. Выбрасывая нули и объединяя одинаковые члены (см. предложения 4.1 и 4.5) дизъюнкции, мы придем, очевидно, к дизъюнктивной нормальной форме. Дело, таким образом, сводится к нахождению метода редукции.

Редукция выполняется тривиальным образом, если выражение F представляет собою дизъюнкцию не менее чем двух членов. В противном случае выражение F обязательно является произведением не менее чем двух сомножителей: $F=F_1F_2\ldots F_n (n\geqslant 2)$. Если в выражении F имеется не менее чем одна дизъюнкция, то можно предполагать, очевидно, что хотя бы один из сомножителей является дизъюнкцией не менее чем двух членов.

В таком случае, применяя к произведению $F_1F_2\ldots F_n$ теорему 4.9, мы представим его в виде дизъюнкции членов, каждый из которых содержит меньшее число дизъюнкций, чем исходное выражение F. Тем самым для случая дизъюнктивных нормальных форм редукция проведена.

Случай конъюнктивных нормальных форм рассматривается точно так же. Только что проведенное доказательство, если в нем всюду заменить дизъюнкции на произведения и наоборот, становится доказательством 5.3 для случая конъюнктивной нормальной формы. Тем самым теорема 5.3 полностью доказана.

Следует заметить, что ввиду принятого нами определения порядка действий в булевой алгебре описанный прием приведения к дизъюнктивной нормальной форме сводится к последовательному раскрытию (в соответствии с первым дистрибутивным законом) всех скобок в выражении, которое содержит знаки отрицания лишь над символами переменных, с последующим исключением нулей и объединением равных членов. Раскрытие скобок производится точно так же, как и в обычной алгебре.

Приведение к конъюнктивной нормальной форме не допускает прямой аналогии с действиями, принятыми в обычной (школьной) алгебре, и осуществляется в связи с этим более формальным приемом, в точном соответствии с последовательностью действий, описанной в доказательстве теоремы 5.3.

Рассмотрим в виде примера процесс приведения к дизъюнктивной и к конъюнктивной нормальной форме выражения:

$$f \underset{\text{Df}}{=} (xy \vee \bar{y}z)\,\overline{\overline{xu}}.$$

На первом этапе опускаем знаки отрицания непосредственно на переменные входящие в выражение:

$$f = (xy \vee \bar{y}z)\,(x \vee \bar{u}).$$

Раскрывая скобки, мы придем к дизъюнктивной нормальной форме:

$$f = xy \vee x\bar{y}z \vee xy\bar{u} \vee \bar{y}z\bar{u}.$$

Эта форма допускает дальнейшие упрощения (например, ее третий член поглощается первым); мы, однако, не будем доводить до конца этих упрощений.

При приведении к конъюнктивной нормальной форме применим теорему 4.9. к выражению $(xy \vee \bar{y}z)(x \vee \bar{u})$:

$$f = (xy \vee \bar{y}z)\,(x \vee \bar{u}) = (x \vee \bar{y})\,(x \vee z)\,(y \vee \bar{y})\,(y \vee z)\,(x \vee \bar{u}) = \\ = (x \vee \bar{y})\,(x \vee z)\,(y \vee z)\,(x \vee \bar{u}).$$

Последнее из полученных выражение представляет собой конъюнктивную нормальную форму.

В дальнейших рассуждениях будем предполагать фиксированным некоторое множество M булевых переменных $x_1, \ldots, x_n$ (обычно в качестве такого множества выбирается множество аргументов той или иной булевой функции).

Элементарные дизъюнкции (соответственно элементарные произведения) называются конституентами нуля (соответственно конституентами единицы) для данного множества M булевых переменных, если они содержат (либо в прямом, либо в инверсном виде) все переменные из множества M.

Для переменных $x_1, \ldots, x_n$ произвольную конституенту нуля можно представить в виде $\tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_n$, а произвольную конституенту единицы — в виде $\tilde{x}_1 \tilde{x}_2 \ldots \tilde{x}_n$, где $\tilde{x}_i$ означает либо x_i, либо $\bar{x}_i$ ($i=1, \ldots, n$).

Нетрудно убедиться в том, что для любой конституенты единицы K существует один и только один набор α значений переменных, входящих в K, на котором эта конституента обращается в единицу. В самом деле, если $K=\tilde{x}_1 \tilde{x}_2 \ldots \tilde{x}_n$, то набор $\alpha=(\alpha_1, \alpha_2, \ldots, \alpha_n)$ тогда и только тогда обращает K в единицу, когда i-я компонента α_i набора α для всех $i=1, \ldots, n$ обращает в единицу соответствующий сомножитель $\tilde{x}_i$ из числа сомножителей, составляющих данную конституенту (см. предложение 4.4). Этим условием каждая компонента α_i (а следовательно, и весь набор α) определяется однозначно: если $\tilde{x}_i=x_i$, то $\alpha_i=1$, если $\tilde{x}_i=\bar{x}_i$, то $\alpha_i=0$. Построенный таким образом набор α и конституента единицы K, которая на этом наборе обращается в единицу, называются *соответствующими друг другу*.

Аналогичным образом для всякой конституенты нуля $R=\tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_n$ имеется один и только один *соответствующий этой конституенте* набор $\beta=(\beta_1, \ldots, \beta_n)$ значений переменных $x_1, \ldots, x_n$, на котором эта конституента равна нулю. В силу предложения 4.2 для этой цели надо выбрать значение $\beta_i (i=1,2,\ldots, n)$ равным нулю, если $\tilde{x}_i=x_i$, и равным единице, если $\tilde{x}_i=\bar{x}_i$.

Условимся нумеровать конституенты нуля и единицы с помощью тех же номеров, что и соответствующие им наборы значений переменных.

Для построения булевой алгебры существенное значение имеют понятия *совершенной дизъюнктивной* и *совершенной конъюнктивной нормальной формы*:

Дизъюнктивная (соответственно конъюнктивная) нормальная форма называется совершенной, если все составляющие ее элементарные произведения (соответственно элементарные дизъюнкции) являются конституентами единицы (соответственно конституентами нуля) для одного и того же множества M переменных. Совершенная дизъюнктивная (конъюнктивная) нормальная форма называется совершенной дизъюнктивной (соответственно конъюнктивной) нормальной формой булевой функции $f=f(x_1, \ldots, x_n)$, если она равна этой функции и если множество составляющих ее переменных совпадает с множеством аргументов функции f.

Справедливо следующее важное предложение.

5.4. *Любая булева функция имеет одну и только одну совершенную дизъюнктивную нормальную форму, а также одну и только одну совершенную конъюнктивную нормальную форму.*

Действительно, пусть $f(x_1, \ldots, x_n)$— произвольная булева функция. Через f_i обозначим значение этой функции на i-м наборе значений переменных, через K_i— конституенту единицы, а через R_i — конституенту нуля, соответствующие этому набору ($i=0, 1, \ldots, 2^n-1$).

Обозначим через φ совершенную дизъюнктивную нормальную форму $f_0K_0 \vee f_1K_1 \vee \ldots \vee f_{2^n-1}K_{2^n-1}$, а через ψ— совершенную конъюнктивную нормальную форму $(f_0 \vee R_0)(f_1 \vee R_1) \ldots (f_{2^n-1} \vee R_{2^n-1})$. В соответствии с предложениями 4.1, 4.2, 4.3 и 4.4, можно считать форму φ совпадающей с дизъюнкцией тех конституент K_i, для которых соответствующие значения функции f_i равны 1, а форму ψ совпадающей с произведением тех конституент R_j, для которых соответствующие значения функции f_j равны 0.

Как форму φ, так и форму ψ можно рассматривать как некоторые функции переменных $x_1, \ldots, x_n$. На i-м наборе мы будем иметь $K_i=1$, $K_j=0$ (при $j \neq i$), $R_i=0$. $R_j=1$ (при $j \neq i$). Используя снова предложения 4.1, 4.2, 4.3, 4.4, мы приходим к выводу, что значения функций φ и ψ на i-м наборе совпадают с f_i, то есть со значением функции f на i-м наборе. Ввиду произвольности набора

приходим к выводу о равенстве между собой всех трех функций f, φ, ψ Следовательно, согласно определению совершенных нормальных форм, φ является совершенной дизъюнктивной, а ψ — совершенной конъюнктивной нормальной формой функции f.

Произвольную совершенную дизъюнктивную нормальную форму ξ функции f можно, очевидно, записать в виде

$$g_0 K_0 \vee g_1 K_1 \vee \ldots \vee g_{2^n-1} K_{2^n-1},$$

где $g_0, g_1, \ldots, g_{2^n-1}$— коэффициенты, равные единице или нулю в соответствии с тем, входит или не входит соответствующая конституента в данную форму ξ. На i-м наборе значений переменных значение формы ξ равно, как нетрудно видеть, g_i. С другой стороны, по условию, значения функции f и формы ξ должны совпадать на всех наборах. Следовательно, для любого i справедливо $g_i = f_i$, так что форма ξ совпадает с формой φ. Ввиду произвольности выбора ξ это означает единственность совершенной дизъюнктивной нормальной формы. Единственность совершенной конъюнктивной нормальной формы доказывается совершенно аналогично. Тем самым теорема 5.4 полностью доказана.

Рассмотрим теперь произвольную дизъюнктивную нормальную форму φ от переменных $x_1, \ldots, x_n$. Пусть некоторое элементарное произведение p, входящее в эту форму, не содержит какой-либо из переменных x_i, например, переменной x_1; тогда его можно заменить равным ему (в силу тождеств (13) и (11) из § 5) выражением $p\,(x_1 \vee \bar{x}_1) = p x_1 \vee p\bar{x}_1$. Продолжая этот процесс относительно остальных переменных множества $x_1, \ldots, x_n$, не входящих в то или иное элементарное произведение, мы после повторения этой процедуры некоторое конечное число раз получим совершенную дизъюнктивную нормальную форму выражения φ, поскольку в каждое составляющее ее элементарное произведение будут входить в с е буквы $x_1, \ldots, x_n$.

Назовем описанный процесс п р и в е д е н и е м (произвольной) д и з ъ ю н к т и в н о й н о р м а л ь н о й ф о р м ы φ к с о в е р ш е н н о м у виду.

Аналогичным образом можно построить процесс п р и в е д е н и я (произвольной) к о н ъ ю н к т и в н о й

нормальной формы ψ к совершенному виду. Для этой цели к входящей в конъюнктивную нормальную форму произвольной элементарной дизъюнкции q, не содержащей, например, буквы x_1, добавляется равный нулю член $x_1\overline{x_1}$. Затем к полученному выражению применяют второй дистрибутивный закон: $q = x_1\overline{x_1} = (q \vee x_1)(q \vee \overline{x_1})$. Продолжая аналогично, мы сможем в каждую элементарную дизъюнкцию ввести все недостающие в ней переменные, после чего форма ψ превратится в совершенную. Отсюда, учитывая 5.3, получаем:

5.5. *Существует единый конструктивный метод, позволяющий для любого выражения булевой алгебры найти равные этому выражению совершенную дизъюнктивную и совершенную конъюнктивную нормальные формы.*

Заметим, что при приведении произвольного выражения булевой алгебры к совершенной дизъюнктивной и к совершенной конъюнктивной нормальной форме мы можем ограничиться использованием лишь основных тождеств булевой алгебры (1)—(14) из § 4, поскольку все остальные применяемые нами тождества являются их следствиями. Сам процесс приведения обратим: применяя все тождества, использованные при приведении данного выражения к совершенной конъюнктивной или совершенной дизъюнктивной нормальной форме, в обратном порядке, мы можем восстановить исходное выражение по его совершенной нормальной форме [1]).

Пусть теперь нам даны два равных выражения булевой алгебры f и g. Их можно рассматривать как булевы функции входящих в них переменных (букв). Согласно теореме 5.4 они имеют одну и ту же совершенную дизъюнктивную нормальную форму φ. Приводя выражения (функции) f к совершенной дизъюнктивной нормальной форме φ и восстанавливая по форме φ выражение g, мы осуществим непрерывную цепь тождественных преобразований, опирающихся лишь на тождества (1)—(14) из § 4 и переводящих выражение f в выражение g. Следовательно,

---

[1]) Этот процесс обратного перехода оказывается, разумеется, возможным лишь тогда, когда известна последовательность тождеств, использованная при приведении исходного выражения к соответствующей совершенной форме.

5.6. *С помощью основных тождеств булевой алгебры можно осуществить преобразование произвольного выражения булевой алгебры в любое равное ему выражение булевой алгебры.*

Иначе говоря, система тождеств (1)—(14) из § 4 оказывается достаточной для выполнения любых преобразований в булевой алгебре. Это свойство естественно назвать *свойством полноты* системы тождеств (1)—(14). Исключив некоторые из тождеств этой системы (относя их в число неосновных тождеств), можно, тем не менее, сохранить за оставшимся множеством основных тождеств свойство полноты. Мы, однако, не будем заниматься исследованием системы тождеств (1)—(14) с этой точки зрения.

В алгебре Жегалкина роль совершенных форм булевой алгебры играют полиномы специального вида, которые мы будем называть *каноническими полиномами.*

Каноническим полиномом называется конечная сумма попарно различных произведений переменных таких, что в одном и том же произведении никакая переменная не встречается более одного раза. При этом к числу произведений относятся произведения, состоящие из одного сомножителя (отдельные переменные), и произведение, состоящее из пустого множества сомножителей (константа 1)

Выражения $xyz+xy+z+1$, x, 1 являются, очевидно, каноническими полиномами, а выражения xx, $xy+yx+1$ — нет, поскольку в первом полиноме имеется произведение, содержащее две буквы x, а второй полином содержит два одинаковых (с точностью до перестановки сомножителей) произведения xy и yx.

Условимся к числу элементарных полиномов относить также константу 0, рассматривая ее как сумму пустого множества членов.

С помощью тождеств (22), (26) и (27) из § 4 любой полином (сумму произвольных произведений переменных) в алгебре Жегалкина легко привести к каноническому виду. Используя же тождества (22), (23), (24) и (25) из § 4, нетрудно любое выражение алгебры Жегалкина преобразовать в полином. Это делается точно так же, как и в обычной (школьной) алгебре,— путем раскрытия всех скобок. Различие заключается лишь в том, что в данном случае мы можем иметь дело лишь с коэффициентами,

равными нулю или единице. Приведение же подобных членов делается в соответствии с тождеством (26) из § 4.

Точно таким же способом, какой был применен при доказательстве теоремы 5.3, можно показать, что описанный прием проведения преобразований выражений алгебры Жегалкина заканчивается через конечное число шагов. Поскольку он приводит к построению канонического полинома, мы можем сформулировать следующий результат.

5.7. *Существует единый конструктивный прием, позволяющий для любого выражения алгебры Жегалкина найти равный ему канонический полином.*

Процесс отыскания канонического полинома, равного заданному выражению алгебры Жегалкина, мы будем называть приведением выражения к каноническому виду, а сам этот полином — *каноническим видом исходного выражения.*

Заметим, что приведение к каноническому виду может быть осуществлено с помощью одних лишь основных тождеств и допускает восстановление (с помощью основных тождеств) исходного выражения по каноническому виду (разумеется, для этого нужно знать цепочку преобразований, которая привела к построению канонического вида данного выражения). Докажем следующую теорему.

5.8. *Если два канонических полинома равны между собой (представляют одинаковые булевы функции от входящих в них переменных), то они совпадают между собой с точностью до перестановки членов.*

Действительно, предположение о несовпадении заданных полиномов φ и ψ приводит к выводу о том, что в одном из них, например в полиноме φ, должно быть слагаемое $p=x_{i_1}x_{i_2}\ldots x_{i_k}$, не содержащееся в другом. Допустим, что выбранное слагаемое p имеет наименьшее возможное число букв, так что все слагаемые, имеющие менее чем k букв, одинаковы в обоих полиномах.

Рассмотрим набор α значений переменных, принимая для всех переменных, входящих в слагаемое p, значения, равные 1, а для всех остальных переменных — значения, равные 0. Любое слагаемое q первого или второго полинома, отличное от p и содержащее равное с ним или большее число букв, обязательно содержит буквы, не входящие в p и, следовательно, обращается в нуль на наборе α.

Слагаемое p принимает на этом наборе значение 1. Подставляя значения переменных из набора α в полиномы φ и ψ, мы приведем их к виду $1+\varphi_0$, ψ_0, где φ_0 и ψ_0 — значения, принимаемые суммами тех членов полиномов φ и ψ соответственно, которые содержат меньшее по сравнению с членом p число букв. Поскольку, ввиду выбора члена p, эти суммы одинаковы, то $\varphi_0=\psi_0$. Но тогда, очевидно, $1+\varphi_0\neq\psi_0$ и, следовательно, наши полиномы φ и ψ представляют разные булевы функции. Если же они представляют одинаковые булевы функции, то их несовпадение не может иметь места. Тем самым теорема доказана.

Если теперь нам даны два произвольных выражения f и g в алгебре Жегалкина, то, приводя их к каноническому виду, мы получим, в силу теоремы 5.8, один и тот же канонический полином φ. Цепь преобразований, приводящая f к φ и восстанавливающая g по φ, использует лишь основные тождества алгебры Жегалкина. Тем самым мы получили доказательство следующего предложения.

5.9. *С помощью основных тождеств* (22)—(27) *алгебры Жегалкина можно осуществить преобразование произвольного выражения этой алгебры в любое равное ему* (*представляющее ту же функцию*) *выражение той же алгебры.*

Предположим, что мы имеем некоторую дизъюнкцию конституент единицы: $f=K_{i_1}\vee K_{i_2}\vee\ldots\vee K_{i_m}$. Заменяя в ней знаки дизъюнкции знаками суммы, мы придем к некоторому выражению в объединенной алгебре: $g=K_{i_1}+K_{i_2}+\ldots+K_{i_m}$ [1]). Покажем, что выражения f и g равны между собой. Действительно, ни при каком наборе значений переменных две различные конституенты единицы не могут одновременно обратиться в единицу, поскольку i-я конституента единицы обращается в единицу лишь на i-м наборе. Следовательно, при определении значений функций f и g нам придется определять значение дизъюнкции (или соответственно суммы) членов, из которых только один может равняться единице, остальные же обязательно равны нулю. Но в этом случае очевидно, что как дизъюнкция, так и сумма по модулю два приводят к одному и тому же результату (к 0, если все члены равны нулю, и к 1, если один и только один из членов равен единице).

[1]) Объединенная алгебра использует как операции булевой алгебры, так и операции алгебры Жегалкина.

Тем самым мы доказали следующее предложение.

5.10. *Если в совершенной дизъюнктивной нормальной форме f произвести замену всех знаков дизъюнкции знаками суммы, а все отрицаемые переменные заменить согласно тождеству $\bar{x}=1+x$, то получится выражение алгебры Жегалкина, представляющее ту же булеву функцию, что и исходная форма f.*

Поскольку в виде совершенной дизъюнктивной нормальной формы может быть представлена любая булева функция (теорема 5.4), то из предложений 5.7, 5.8 и 5.10 вытекает справедливость следующего результата.

5.11. *Любая булева функция может быть представлена в алгебре Жегалкина в виде одного и (с точностью до порядка членов) только одного канонического полинома.*

Представление выражения алгебры Жегалкина в виде канонического полинома мы будем называть приведением выражения к каноническому виду.

В заключение параграфа рассмотрим примеры приведения выражений булевой алгебры и алгебры Жегалкина к рассмотренным выше нормальным и каноническим формам.

Пример 1. Выражение $f=\overline{(x \vee yz)}\,(x \vee z)$ привести к совершенной дизъюнктивной и к совершенной конъюнктивной нормальной форме.

Решение.

1) $f=\bar{x}\,(\bar{y} \vee z)\,(x \vee z)=(\bar{y} \vee z)\,(\bar{x}\bar{x} \vee \bar{x}z)=\bar{x}\,\bar{y}z \vee \bar{x}z=$
$=\bar{x}\,\bar{y}z \vee \bar{x}z\,(y \vee \bar{y})=\bar{x}\,\bar{y}z \vee \bar{x}yz.$

2) $f=\bar{x}\,(\bar{y} \vee z)\,(x \vee z)=(\bar{x} \vee y\bar{y})\,(\bar{y} \vee z \vee \bar{x}\bar{x})\,(x \vee z \vee y\bar{y})=$
$=(\bar{x} \vee y)\,(\bar{x} \vee \bar{y})\,(x \vee \bar{y} \vee z)\,(\bar{x} \vee \bar{y} \vee z)\,(x \vee y \vee z)\cdot$
$\cdot\,(x \vee \bar{y} \vee z)=(\bar{x} \vee y \vee z\bar{z})\,(\bar{x} \vee \bar{y} \vee z\bar{z})\,(x \vee \bar{y} \vee z)\cdot$
$\cdot\,(\bar{x} \vee \bar{y} \vee z)\,(x \vee y \vee z)\,(x \vee \bar{y} \vee z)=(\bar{x} \vee y \vee z)\cdot$
$\cdot\,(\bar{x} \vee y \vee \bar{z})\,(\bar{x} \vee \bar{y} \vee \bar{z})\,(\bar{x} \vee \bar{y} \vee z)\,(x \vee \bar{y} \vee z)\cdot$
$\cdot\,(x \vee \bar{y} \vee z)\,(x \vee y \vee z)\,(x \vee \bar{y} \vee z)=(\bar{x} \vee \bar{y} \vee \bar{z})\cdot$
$\cdot\,(\bar{x} \vee \bar{y} \vee z)\,(\bar{x} \vee y \vee \bar{z})\,(\bar{x} \vee y \vee z)\,(x \vee \bar{y} \vee z)\,(x \vee y \vee z).$

Пример 2. Привести к каноническому виду выражение:

$$g=(x+y+1)(z+1)+(x+y)(x+1).$$

Решение: $g=xz+yz+z+x+y+1+x+xy+x+y=xz+yz+xy+x+z+1.$

§ 6. Анализ и синтез комбинационных схем

Комбинационной схемой в настоящем параграфе мы будем называть произвольную композицию конечных автоматов без памяти [1]), называемых обычно *логическими элементами*. Всякая комбинационная схема обладает некоторым множеством внешних входных узлов (входных полюсов), некоторым множеством внешних выходных узлов (выходных полюсов) и некоторым множеством внутренних узлов. Первые два множества предполагаются всегда непустыми, а третье может оказаться и пустым. Комбинационную схему с m входными и с n выходными полюсами принято называть (m, n)-*полюсником*. Элементарный сигнал в каждом узле схемы обозначается специальной буквой, называемой переменной, соответствующей данному узлу. Эта буква может принимать различные значения из структурного алфавита схемы. Переменные, соответствующие входным, выходным и внутренним узлам схемы, называются соответственно *входными*, *выходными* и *внутренними переменными*. Переменные, соответствующие узлам комбинационной схемы, рассматриваются обычно как обозначения не только элементарных сигналов в узлах, но и самих этих узлов. Условимся в дальнейшем через $x_1,\ldots,x_m$ обозначать входные переменные, через $y_1,\ldots,y_n$ — выходные, а через $z_1, \ldots, z_p$ — внутренние. В случае, когда характер переменных неизвестен, мы будем употреблять для их обозначения буквы u, v, w и др. с различными индексами.

Договоримся в качестве логических элементов рассматривать лишь такие автоматы без памяти, которые обладают одним элементарным выходным каналом. Такое предположение не нарушает общности, поскольку всякий логический элемент, имеющий $k (k>1)$ элементарных вы-

[1]) В том числе и такую, для которой не выполняются условия корректности построения, сформулированные в § 1.

ходных каналов, можно заменить k логическими элементами, имеющими те же самые элементарные входные каналы и лишь по одному элементарному выходному каналу. В силу сделанного предположения структурная функция выходов каждого логического элемента сводится к некоторой переключательной функции, которую мы будем называть *переключательной функцией данного логического элемента*.

Используя метод задания логических элементов переключательными функциями, можно ввести аналитическое описание комбинационных схем с помощью *систем переключательных уравнений*. Такое описание осуществляется следующим образом. Вводятся обозначения переменных (элементарных сигналов), соответствующих всем узлам рассматриваемой комбинационной схемы. Все переменные, соответствующие отождествляемым (соединяемым между собою) узлам, приравниваются друг другу. Для каждого логического элемента A, входящего в схему, выписывается уравнение вида $v=f(u_1,\ldots, u_k)$, где f — переключательная функция данного элемента, v — переменная, обозначающая узел, к которому подсоединен выходной канал логического элемента A, а u_i — переменная, обозначающая узел, к которому подсоединен i-й входной канал элемента A ($i=1, \ldots, k$). Система всех полученных таким образом уравнений называется *системой уравнений непосредственных связей* рассматриваемой комбинационной схемы.

Следует отметить, что обычно в целях экономии обозначений отождествляемые между собою узлы комбинационной схемы заменяются просто одним узлом. Исключение составляют лишь входные и выходные узлы схемы: если некоторый входной узел x_i схемы отождествлен (соединен) с каким-либо выходным узлом y_i, то эти узлы всегда сохраняют свою индивидуальность (не заменяются одним узлом). Благодаря этому обстоятельству множества входных и выходных переменных можно считать попарно непересекающимися. Можно считать также, не нарушая общности, что в системе уравнений непосредственных связей любой комбинационной схемы отсутствуют тривиальные уравнения, т. е. уравнения вида $v=u$, где v и u — просто некоторые переменные, за исклю-

чением случая, когда u является входной, а v — выходной переменной. Иногда, впрочем, целесообразно включать в систему и тривиальные уравнения.

В связи с введенными понятиями естественно сформулировать следующее определение.

6.1. *Комбинационной системой переключательных уравнений называется система уравнений с переменными, принимающими значения в некотором структурном алфавите, все уравнения которой имеют вид* $v=f(u_1,\ldots,u_k)$ *и для которой фиксированы непустые попарно непересекающиеся множества входных и выходных переменных.*

Заметим, что в уравнениях комбинационных систем переменная, стоящая в левой части какого-либо уравнения, может, вообще говоря, встречаться и в правой части того же самого уравнения.

Легко видеть, что всякая комбинационная система переключательных уравнений может рассматриваться как система уравнений непосредственных связей некоторой комбинационной схемы, и наоборот. (См. примечание на стр. 222.) Итак, справедливо следующее предложение.

6.2. *Имеет место взаимно однозначное соответствие между комбинационными схемами и комбинационными системами переключательных уравнений.*

Теорема 6.2 полностью сводит изучение комбинационных схем к изучению комбинационных систем переключательных уравнений.

В ряде случаев комбинационные системы переключательных уравнений оказывается удобным трактовать как систему переключательных функций, в которых зафиксированы некоторые обозначения как для самих функций, так и для их аргументов, а также выделены множества входных и выходных переменных. Условимся называть такие системы *структурными системами переключательных функций*. Известное неудобство при этом составляет то обстоятельство, что иногда та или иная функция структурной системы может оказаться обозначенной той же буквой, что и некоторый ее аргумент. Однако в будущем мы ограничимся в основном так называемыми *правильными* структурными системами, для которых подобное обстоятельство никогда не может иметь места.

Трактуя в качестве структурной системы переключательных функций систему уравнений непосредственных связей комбинационной схемы, мы приходим к понятию *системы функций непосредственных связей* данной схемы.

Как известно из § 1, далеко не всякая композиция автоматов и, в том числе, далеко не каждая комбинационная схема может рассматриваться как структурная схема некоторого автомата. В случае, когда такое рассмотрение оказывается возможным, соответствующая комбинационная схема называется *корректно построенной*. К числу корректно построенных комбинационных схем относятся так называемые *правильные комбинационные схемы*.

Правильной комбинационной схемой мы будем называть комбинационную схему без петель, у которой к каждому узлу, отличному от входных полюсов схемы, подсоединен точно один выходной канал какого-либо логического элемента или точно один входной полюс; выходные каналы логических элементов не подсоединены ни к одному из входных полюсов. Комбинационные схемы, на которые наложено лишь условие отсутствия петель, называются условно правильными.

Говорят, что узел, присоединенный к одному из входных каналов какого-либо логического элемента, *непосредственно связан с узлом*, подсоединенным к выходному каналу того же самого элемента. Будем говорить также, что узел a *связан* с узлом b, если существует цепь, начинающаяся узлом a и кончающаяся узлом b (см. § 1).

В комбинационной схеме S без петель обязательно существуют узлы, с которыми не связаны непосредственно никакие другие узлы схемы. Действительно, если бы это было не так, то для любого узла схемы можно было бы построить цепь неограниченной длины, кончающуюся данным узлом. Ввиду ограниченности общего числа узлов в схеме в этой цепи какой-либо узел рано или поздно повторится дважды, то есть цепь образует петлю, что, однако, исключено первоначальным условием.

Таким образом, в комбинационной схеме S без петель существуют узлы, с которыми не связаны непосредственно никакие другие узлы. Назовем такие узлы *узлами нулевой ступени*. В числе оставшихся узлов (если таковые

имеются) обязательно найдутся узлы, с которыми не связаны непосредственно никакие другие узлы, кроме узлов нулевой ступени. Назовем такие узлы *узлами первой ступени*. Их существование доказывается точно так же, как и существование узлов нулевой ступени: из их отсутствия вытекала бы возможность построения цепи неограниченной длины.

Описанный процесс продолжается до тех пор, пока не будут исчерпаны все узлы схемы: узлами i-й ступени называются узлы, не являющиеся узлами меньших ступеней $(0, 1, \ldots, i-1)$, причем такие, что с ними непосредственно связаны лишь узлы низших ступеней. Если в схему S входят узлы n-й ступени, но не входят узлы $(n+1)$-й ступени, то схема S называется *n-ступенной комбинационной схемой* (без петель).

Условимся называть структурную систему Q переключательных функций *ступенчатой системой*, если все входящие в нее переменные разбиваются на попарно непересекающиеся классы $K_0, K_1, \ldots, K_n$ так, что переменными класса K_0 не обозначается ни одна из функций системы, а для любого $i=1, \ldots, n$ переменные класса K_i обозначают лишь такие функции, аргументы которых обозначены переменными классов $K_0, \ldots, K_{i-1}$. Переменные, входящие в класс K_i, называются переменными i-й ступени.

Из проведенных выше рассуждений непосредственно вытекает справедливость следующего результата.

6.3. *Ступенчатые системы переключательных функций, и только они, могут рассматриваться как системы функций непосредственных связей комбинационных схем без петель.*

Рассмотрим теперь правильную комбинационную схему P. В силу приведенного выше определения в схеме P входные полюсы и все отождествленные с ними узлы будут узлами нулевой ступени. На эти узлы элементарные сигналы подаются извне. Из определения понятия узла i-й ступени следует, что к узлам первой ступени подсоединены выходные каналы логических элементов, все входные каналы которых подсоединены к узлам нулевой ступени. Таким образом, на узлах первой ступени возникнут вполне определенные элементарные сигналы. Это

в свою очередь приведет к возникновению вполне определенных элементарных сигналов на узлах второй ступени, и т. д.

В результате оказывается, что в правильной комбинационной схеме P элементарные сигналы во всех узлах схемы оказываются вполне определенными функциями элементарных сигналов на входных полюсах. Таким образом, схема P построена корректно. Из проведенных рассуждений легко следует также, что рассмотренное определение правильных комбинационных схем является частным случаем определения правильных схем, данного в § 1.

Условимся называть *правильной системой переключательных функций* любую структурную систему переключательных функций, которую можно рассматривать как систему функций непосредственных связей какой-либо правильной комбинационной схемы.

Справедливо следующее предложение.

6.4. *Структурная система переключательных функций тогда и только тогда правильна, когда она является ступенчатой системой, у которой все входные переменные входят в функции системы лишь в качестве аргументов, а каждая из остальных переменных обозначает в точности одну из функций системы.*

Справедливость сформулированного предложения следует из приведенных выше рассуждений. Необходимо лишь напомнить, что узлам, отождествленным с входными узлами в правильной комбинационной схеме P, будут отвечать функции вида $v=x_i$.

Если рассматривать произвольный автомат без памяти, то его функционирование определяется заданием его структурной функции выходов (см. § 1). В свою очередь структурная функция выходов задается системой переключательных функций, определяющих зависимость элементарного сигнала на каждом из выходных полюсов автомата от элементарных сигналов на его входных полюсах. Назовем эти функции *выходными функциями* данного автомата без памяти. В частности, если комбинационная схема может рассматриваться как автомат без памяти (то есть, если она корректно построена), то можно говорить о *выходных функциях этой схемы.*

В теории комбинационных схем можно сформулировать две основные задачи. Первая задача, которую естественно называть *задачей анализа комбинационных схем*, может быть сформулирована следующим образом.

Найти общий конструктивный прием (алгоритм), позволяющий по любой корректно построенной комбинационной схеме построить выходные функции этой схемы.

Вторая задача, называемая *задачей синтеза комбинационных схем*, формулируется так:

Найти общий конструктивный прием (алгоритм), позволяющий по любой заданной функционально полной (см. § 2) *системе логических элементов (переключательных функций) и любой системе переключательных функций (в том же самом структурном алфавите)* $f_1, \ldots, f_k$ *найти корректно построенную (из элементов данного типа) комбинационную схему, для которой функции* $f_1, \ldots, f_k$ *служили бы выходными функциями.*

Для случая правильных комбинационных схем решение задачи анализа получается в результате последовательного применения к системам функций непосредственных связей этих схем так называемых *операций внутренней суперпозиции*.

Операция внутренней суперпозиции может быть применена к любой структурной системе S переключательных функций. Суть этой операции состоит в пополнении системы S новой переключательной функцией по следующим правилам:

1. Из системы S выбирается произвольная функция $v=f(u_1, \ldots, u_q)$ такая, что какой-либо из ее аргументов, например u_i, обозначен в системе S той же буквой, что и некоторая функция $u_i=\varphi(w_1, \ldots, w_r)$ той же системы.

2. Образуется новая функция $v=f(u_1, \ldots, u_{i-1}, \varphi(w_1, \ldots, w_r), u_{i+1}, \ldots, u_q)$, получаемая с помощью подстановки функции φ вместо переменной u_i в функцию f. Эта новая функция обозначается той же буквой v, что и исходная функция f, и заменяет функцию f в системе S.

Из проведенного выше рассмотрения законов передачи элементарных сигналов в правильных комбинационных схемах от узлов нулевой ступени к узлам высших ступеней непосредственно следует справедливость следующего факта.

6.5. *В результате применения конечного числа операций внутренней суперпозиции к правильной системе переключательных функций P, независимо от порядка выполнения этих операций, возникает одна и та же система переключательных функций, обозначенных всеми переменными исходной системы P, отличными от входных переменных (по одной функции на каждую такую переменную), причем аргументами этих функций являются одни лишь входные переменные системы S.*

Система переключательных функций P являлась системой функций непосредственных связей некоторой правильной комбинационной схемы Q. Система же функций S, полученная из системы P методом, описанным в теореме 6.5, будет задавать элементарные сигналы во всех узлах схем, отличных от входных узлов, как функции элементарных сигналов в ее входных узлах. Среди этих функций будут, в частности, и все выходные функции исходной комбинационной схемы Q.

Таким образом, теорема 6.5 дает общий прием решения задачи анализа для случая правильных комбинационных схем. Операция, заключающаяся в переходе от системы P к системе S в теореме 6.5, будет называться нами операцией *полной внутренней суперпозиции* функций системы P.

Решение задачи синтеза комбинационных схем мы проведем лишь для случая двоичного структурного алфавита. Причина такого ограничения состоит в том, что при этом мы можем воспользоваться развитым выше аппаратом теории булевых функций. В случае же произвольного структурного алфавита необходимо было бы предварительно развить соответствующий аппарат для произвольных переключательных функций. В то же время особой практической необходимости подобного обобщения теории пока еще не имеется, поскольку подавляющее большинство встречающихся на практике комбинационных схем оперирует именно с двоичным структурным алфавитом.

Заметим также, что с теоретической точки зрения достаточно решить задачу синтеза для случая комбинационных схем лишь с одним выходным полюсом. Действительно, комбинационную схему, обладающую несколькими выходными полюсами, можно всегда представлять себе в

виде композиции схем, каждая из которых обладает лишь одним выходным полюсом, с отождествленными входными полюсами.

Полное решение (по крайней мере в теоретическом плане) задачи синтеза для случая правильных комбинационных схем в двоичном структурном алфавите будет дано в следующем параграфе; пока мы рассмотрим лишь один весьма специальный случай синтеза, который явится отправным пунктом для имеющих в дальнейшем место построений. Речь идет о синтезе правильных комбинационных схем из логических элементов, реализующих булевы функции, принятые в качестве основных операций булевой алгебры. Для таких элементов приняты специальные наименования.

Логический элемент, реализующий функцию отрицания $\overline{x}$, принято называть *инвертором*. *Совпадением* называют обычно элемент, выходной сигнал которого представляет собой произведение (конъюнкцию) всех его входных сигналов, а *разделением* — элемент, выходной сигнал которого является дизъюнкцией всех его входных сигналов. Заметим, что наряду с двухвходовыми совпадениями и разделениями употребляются и многовходовые элементы этого же типа, реализующие соответственно функции $x_1x_2 \ldots x_n$ и $x_1 \vee x_2 \vee \ldots \vee x_n$. Однако в этой главе мы будем употреблять лишь двухвходовые совпадения, не оговаривая этого всякий раз особо.

Все результаты настоящего параграфа, полученные для произвольного структурного алфавита, имеют, разумеется, силу и для двоичного структурного алфавита. Заметим лишь, что теперь мы будем переключательные функции всюду именовать булевыми функциями. В частности, правильные комбинационные схемы будут задаваться с помощью правильных систем булевых функций.

Для рассматриваемого специального случая синтеза возможен, однако, и несколько иной способ аналитического представления схем.

Условимся называть правильными булевыми схемами правильные комбинационные схемы в двоичном структурном алфавите, построенные из инверторов, двухвходовых совпадений и двухвходовых разделений. Правильную булеву

схему с m входными и с n выходными полюсами будем называть правильным булевым (m, n)-полюсником.

Мы будем предполагать дополнительно, что правильный булев (m, n)-полюсник не содержит л и ш н и х у з л о в, то есть таких узлов, которые не связаны никакими цепями с его выходными полюсами.

Как уже отмечалось выше, синтез произвольных комбинационных схем сводится к синтезу $(m,1)$-полюсников, причем, как легко видеть, можно ограничиться лишь такими $(m,1)$-полюсниками, которые являются правильными комбинационными схемами, не содержащими лишних узлов. В связи с этим значительный интерес приобретает задача синтеза правильных булевых $(m,1)$-полюсников.

Рассмотрим систему S (булевых) функций непосредственных связей произвольного булева $(m,1)$-полюсника P. Пусть $x_1, \ldots, x_m, z_1, \ldots z_p, y$ — все переменные, входящие в систему S. Поставим задачу определения значения выходного сигнала y в зависимости от входных сигналов $x_1, \ldots, x_m$. Такое определение предполагает, что последовательно, шаг за шагом, находятся значения всех переменных $z_1, \ldots, z_p, y$. Ввиду отсутствия лишних узлов значение y будет найдено лишь на последнем шаге, после того как будут найдены значения всех переменных $z_1, \ldots, z_p$, причем все эти значения существенны для нахождения значения y. На каждом шаге над найденными ранее значениями производится одна из операций булевой алгебры, а весь процесс эквивалентен нахождению значения некоторого булева выражения от входных переменных $x_1, \ldots, x_m$ (без констант). Соответствующее булево выражение, очевидно, полностью определяет исходный $(m, 1)$-полюсник P.

Обратно, произвольное выражение A булевой алгебры, построенное из переменных $x_1, \ldots, x_m$ (без констант), можно записать в виде системы функций непосредственных связей некоторого правильного булева $(m,1)$-полюсника. Для этой цели достаточно ввести соответствующие буквенные обозначения для всех составных частей выражения так, чтобы любая обозначенная часть выражения могла быть получена из других его обозначенных частей с помощью применения одной булевой операции.

Таким образом, оказывается справедливым следующее предложение.

6.6. *Существует естественным образом определяемое однозначное соответствие между правильными булевыми* $(m, 1)$*-полюсниками и выражениями булевой алгебры от* m *переменных (без констант).*

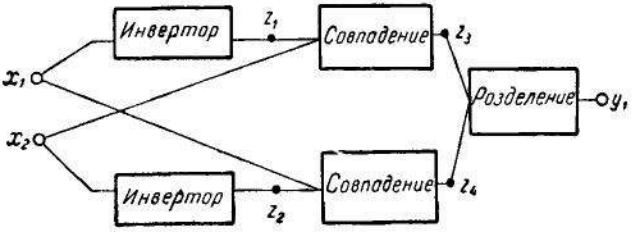

Рис. 10

Рассмотрим в виде примера процесс представления в виде булева выражения правильного булева (2,1)-полюсника P, изображенного на рис. 10.

Система функций непосредственных связей схемы P имеет вид

$$z_1 = \bar{x}_1, \quad z_2 = \bar{x}_2, \quad z_3 = z_1 x_2, \quad z_4 = x_1 z_2, \quad y_1 = z_3 \vee z_4.$$

Эта система может быть представлена эквивалентным образом в виде следующего булева выражения:

$$y_1 = \bar{x}_1 x_2 \vee x_1 \bar{x}_2.$$

Это выражение и будет выражением булевой алгебры, соответствующим (2,1)-полюснику P.

Легко видеть, что справедливо следующее предложение.

6.7. *Если к системе* S *функций непосредственных связей правильного булева* $(m,1)$*-полюсника* P *с входными переменными* $x_1, \ldots, x_m$ *и выходной переменной* y *применить операцию полной внутренней суперпозиции, не производя никаких упрощений, получаемых в результате суперпозиции выражений, то единственная функция, обозначенная буквой* y*, в заключительной системе функций будет представ-*

лять собою булево выражение, соответствующее схеме исходного $(m,1)$-полюсника P.

В предыдущем параграфе был указан общий конструктивный прием, позволяющий представлять произвольную булеву функцию $(x_1, \ldots, x_m)$ в виде некоторого специального выражения булевой алгебры,— в виде совершенной дизъюнктивной нормальной формы. При этом совершенная дизъюнктивная нормальная форма функции $f(x_1, \ldots, x_m)$ содержит лишь переменные $x_1, \ldots, x_m$ и не содержит констант 0 и 1, за исключением случая, когда функция f тождественно равна нулю. Однако и в этом последнем случае функцию f можно представить в виде булева выражения, не содержащего констант,— например в виде произведения $x_1\overline{x_1}$.

Поскольку всякое булево выражение однозначно определяет правильный булев $(m,1)$-полюсник, то описанный прием решает задачу синтеза правильных булевых $(m,1)$-полюсников. Умея же решать эту задачу, мы можем построить также правильный булев (m,n)-полюсник, реализующий любую заданную систему выходных функций.

Если теперь необходимо решить общую задачу синтеза комбинационных схем в двоичном структурном алфавите, то достаточно научиться строить из заданной системы логических элементов (1,1)-полюсник, реализующий отрицание, и (2,1)-полюсник, реализующий умножение или дизъюнкцию. Умея реализовать отрицание и умножение, или отрицание и дизъюнкцию, мы с помощью формул де Моргана (см. § 4) можем построить схемы, реализующие также дизъюнкцию или соответственно умножение. Имея же логические элементы, реализующие отрицание, умножение и дизъюнкцию, можно, как было показано выше, строить схемы, реализующие любые системы выходных функций (в двоичном структурном алфавите).

Нами доказано, следовательно, следующее предложение.

6.8. *Для решения общей задачи синтеза комбинационных схем в двоичном структурном алфавите достаточно уметь строить из заданной системы логических элементов комбинационные схемы, имеющие своими выходными функциями отрицание $\overline{x}$ и произведение x_1x_2.*

Следует подчеркнуть, что метод решения общей задачи синтеза, намечаемый теоремой 6.8, имеет скорее теоретическое, чем практическое значение, поскольку он приводит, как правило, к излишне сложным схемам. Необходимо поэтому разрабатывать практически более удобные методы решения задач синтеза и минимизации комбинационных схем применительно к определенным типам логических элементов. Такие методы развиваются в двух последующих главах, а сейчас мы ограничимся тем, что дадим классификацию основных типов задач синтеза комбинационных схем, встречающихся на практике.

Так называемая *общая задача комбинационного синтеза* состоит в разработке методов построения корректных комбинационных схем, реализующих любые заданные выходные системы булевых функций из логических элементов любых фиксированных типов, при условии, что на входные узлы схемы могут подаваться лишь входные переменные. При этом как в данной задаче, так и в других задачах комбинационного синтеза необходимо заботиться о минимизации числа логических элементов, используемых при построении любой данной схемы.

В общую задачу комбинационного синтеза вводятся часто некоторые дополнительные условия. Так, в случае, когда на входные полюсы схемы могут подаваться не только входные переменные, но и константы (0 и 1), мы получаем общую задачу синтеза при наличии констант. В случае, когда на входные полюсы подаются не только входные переменные, но и их отрицания, мы приходим к задаче синтеза при наличии инверсий переменных. Введенные дополнительные условия могут встречаться на практике в различных сочетаниях.

Канонической задачей комбинационного синтеза мы условимся называть задачу синтеза правильных булевых (m, n)-полюсников при наличии констант и инверсий переменных из двухвходовых совпадений и разделений.

Важной задачей является также *задача синтеза вентильных схем*. В этой задаче употребляется специальный тип логического элемента, называемый *вентилем*. С точки зрения чисто функциональной вентиль представляет собою двухвходовое совпадение, реализующее функцию xy

Однако он имеет одну отличительную особенность, заключающуюся в том, что один из его входных каналов, называемый *управляющим входом вентиля*, может подсоединяться лишь к входным полюсам вентильной схемы, на которые подаются входные переменные и их отрицания. В вентильной схеме обязательно имеется так называемый *вентильный входной полюс*, на который обычно подается входной сигнал, тождественно равный единице. Что касается сигнала 0, то в вентильных схемах он служит *естественным нулевым сигналом*: он возникает, в частности, на всех изолированных узлах схемы (то есть на узлах, с которыми не связан никакой другой узел схемы). В вентильных схемах имеет место также и *естественное разделение* сигналов, причем это разделение таково, что сигнал 1 оказывается старшим по отношению к сигналу 0. Иначе говоря, одновременный приход на какой-либо узел схемы элементарных сигналов, из которых хотя бы один равен 1, обеспечивает наличие в этом узле сигнала 1. Сигнал 0 в узле может быть лишь в том случае, когда все пришедшие одновременно в этот узел элементарные сигналы равны нулю. Это означает, очевидно, что *элементарный сигнал в узле определяется как дизъюнкция всех одновременно приходящих в него элементарных сигналов.*

Кроме обычных так называемых *односторонних вентилей* на практике употребляются также *двусторонние вентили*. Односторонний вентиль пропускает сигнал 1 от вентильного входного канала к выходному каналу тогда и только тогда, когда на управляющий вход вентиля подан сигнал 1. Двусторонний вентиль действует точно так же, но, в отличие от одностороннего вентиля, он пропускает сигналы и в обратном направлении (от выходного канала к вентильному входному каналу), в силу чего фактически стирается разница между входным и выходным каналом (не считая управляющего входа). Примером двусторонних вентилей являются обычные электромагнитные реле. В чисто электронных схемах обычно встречаются лишь односторонние вентили.

Вентильные схемы допускают изучение с помощью систем уравнений непосредственных связей (систем функций непосредственных связей). Однако, поскольку в вентильных схемах обычно встречаются петли, для их

изучения целесообразно развивать особый аппарат. Задача развития такого аппарата частично решается в пятой главе.

В заключение параграфа заметим, что в случае наличия естественного нулевого сигнала и естественного разделения сигналов описанного выше типа всякая условно правильная схема (схема без петель) естественным образом интерпретируется как правильная схема. Для этой цели схема дополняется (способом, описанным в § 1) нулевыми и разделяющими элементами, причем, в силу описанного выше характера естественного разделения в двоичном алфавите, разделяющий элемент будет являться логическим элементом разделения в смысле настоящего параграфа.

Система S функций непосредственных связей любой условно правильной комбинационной (двоичной) схемы превращается в правильную систему булевых функций в результате выполнения следующих двух операций:

а) Вводится обозначение нулевой функции (константы 0) с помощью всех переменных системы S, отличных от входных переменных и таких, что они не обозначают в системе S никаких функций (включая тривиальные функции $y=x_l$).

б) Если какой-либо переменной v в системе S обозначено несколько функций $f_1, \ldots, f_k$, то вместо всех этих функций в систему вводится лишь одна функция, обозначенная переменной v, а именно дизъюнкция всех функций $f_1, \ldots, f_k$.

С помощью этих двух правил задача анализа условно правильных комбинационных (двоичных) схем, заданных системами функций непосредственных связей, сводится (при условии наличия естественного нулевого сигнала и естественного разделения сигналов) к соответствующей задаче для правильных комбинационных схем.

Пусть, например, схема задана системой S функций непосредственных связей:

$$z_1 = x_1 x_2 \quad z_1 = \bar{x}_1, \quad y = \bar{z}_1 \vee z_2,$$

где x_1, x_2 — входные переменные, y — выходная переменная. При наличии естественного нулевого сигнала и естественного разделения сигналов система S преобразуется

в правильную систему

$$z_1 = x_1 x_2 \vee \bar{x}_1, \quad z_2 = 0, \quad y = \bar{z}_1 \vee z_2.$$

После полной внутренней суперпозиции система принимает вид:

$$z_1 = x_1 x_2 \vee \bar{x}_1, \quad z_2 = 0,$$
$$y = \overline{x_1 x_2 \vee \bar{x}_1} = (\bar{x}_1 \vee \bar{x}_2)\, x_1 = x_1 \bar{x}_2.$$

§ 7. Теорема о функциональной полноте

Основной целью настоящего параграфа является установление необходимых и достаточных условий функциональной полноты системы логических элементов в двоичном структурном алфавите.

Напомним, что система логических элементов называется *функционально полной*, если существует общий конструктивный прием, позволяющий строить из логических элементов заданной системы корректные комбинационные схемы, имеющие любые наперед заданные выходные функции.

Произведем некоторое сужение задачи, ограничившись только правильными комбинационными схемами. Как было показано в предыдущем параграфе, выходные функции таких схем получаются в результате суперпозиций функций, реализуемых выбранными логическими элементами, и тривиальных функций $f(x_1,\ldots,x_m) \equiv x_{m1}$ (x_i пробегает все входные переменные).

Это обстоятельство приводит к следующему естественному определению.

7.1. *Система S булевых функций называется функционально полной, если для любой булевой функции $f(x_1, \ldots, x_m)$ можно построить равную ей булеву функцию, представляющую собой результат суперпозиции (вообще говоря, многократной) функций $x_1, \ldots, x_m$ и функций системы S, взятых в любом конечном числе экземпляров каждая (m — любое натуральное число).*

Оказывается полезным наряду с понятием функциональной полноты ввести так называемое понятие *ослабленной функциональной полноты* системы булевых функций.

7.2. *Система S булевых функций называется ослабленно функционально полной, если для любой булевой функции $f(x_1, \ldots, x_m)$ можно построить равную ей булеву функцию, представляющую собою результат суперпозиции (вообще говоря, многократной) функций $x_1, \ldots, x_m$, констант 0 и 1 и функций системы S, взятых в любом конечном числе экземпляров каждая (m есть любое неотрицательное число).*

Введенные определения легко распространить на произвольные переключательные функции, однако мы не будем делать такого распространения и ограничимся лишь случаем булевых функций (то есть случаем двоичного структурного алфавита).

При установлении необходимых и достаточных условий функциональной полноты (найденных впервые Э. П о с т о м [1])) большую роль играют пять замечательных классов булевых функций, замкнутых относительно операции суперпозиции.

Первый класс составляют *функции, сохраняющие константу* 0, то есть такие булевы функции $f(x_1, \ldots, x_m)$, что $f(0, 0, \ldots, 0)=0$. Поскольку на одном из наборов значения функции фиксированы, произвольными остаются (в случае n переменных) лишь $2^n - 1$ наборов. Следовательно, имеется ровно $\frac{1}{2} 2^{2^n}$ булевых функций от n переменных, сохраняющих константу нуль.

Нетрудно убедиться в том, что справедливо следующее предложение.

7.3. *Суперпозиция любого числа булевых функций, сохраняющих константу* 0, *является также функцией, сохраняющей константу* 0.

Для обоснования этого предложения достаточно рассмотреть суперпозицию двух функций, например функций $z=f(y_1, \ldots, y_m)$ и $y_1=g(x_1, \ldots, x_n)$. При подстановке нулевых значений аргументов в суперпозицию

$$v=f(g(x_1, \ldots, x_n), y_2, \ldots, y_m)$$

этих функций, мы найдем, что значение функции v будет также равно нулю.

---

[1]) E. P o s t, Introduction to a general theory of elementary propositions, Amer. J. Math., v. 43, 1921, p. 163—185.

Второй замечательный класс булевых функций составляют *функции, сохраняющие константу* 1, то есть такие булевы функции $f(x_1, \ldots, x_n)$, что $f(1,1, \ldots, 1)=1$. Как и в предыдущем случае, нетрудно показать, что имеется ровно $\frac{1}{2}2^{2^n}$ булевых функций от n переменных, сохраняющих константу 1. Справедливо также следующее предложение, аналогичное предложению 7.1.

7.4. *Суперпозиция любого числа булевых функций, сохраняющих константу* 1, *является функцией, сохраняющей константу* 1.

Третий замечательный класс булевых функций составляют так называемые *самодвойственные функции*. Булевы функции $f(x_1, \ldots, x_n)$ и $g(x_1, \ldots, x_n)$ называются двойственными друг другу, если $g(x_1, \ldots, x_n)=\overline{f(\bar{x}_1,\ldots,\bar{x}_n)}$ (очевидно, что в этом случае будет справедливым также и соотношение $f(x_1, \ldots,x_n)=\overline{g(\bar{x}_1, \ldots, \bar{x}_n)}$). Булева функция $f(x_1, \ldots, x_n)$ называется *самодвойственной*, если она двойственна по отношению к себе самой, то есть если выполняется соотношение $f(x_1, \ldots,x_n)=\overline{f(\bar{x}_1, \ldots,\bar{x}_n)}$. Если условиться называть *противоположными наборами* набор $(\alpha_1, \ldots, \alpha_n)$ и набор $(\bar{\alpha}_1, \ldots, \bar{\alpha}_n)$, то нетрудно следующим образом перефразировать определение самодвойственных функций:

Булева функция называется самодвойственной, если на любых двух противоположных наборах она принимает противоположные значения (0 и 1).

При расположении наборов в естественном (лексикографическом) порядке [1]) противоположные друг другу наборы помещаются симметрично относительно середины этого расположения, так как $(i-1)$-й и (2^n-i)-й наборы для любого $i=1, \ldots, 2^n$ оказываются противоположными друг другу. Отсюда следует, что столбец значений самодвойственной функции должен быть антисимметричным

[1]) Естественным порядком называется такой порядок расположения наборов, при котором каждый набор совпадает с представлением в двоичной системе счисления своего номера в этом расположении: нулевым набором будет при этом набор 0...000, первым набором — набор 0...001, вторым — набор 0... 010, третьим — набор 0...011 и т. д.

относительно своей середины. По этому признаку легко выделить самодвойственные функции в случае табличного способа задания функций.

Как явствует из сказанного, самодвойственная функция полностью определяется заданием значений, принимаемых ею на половине всех наборов, причем значения ее на наборах этой половины могут быть выбраны произвольно. Следовательно, имеется точно $2^{1/2 \cdot 2^n} = \sqrt{2^{2^n}}$ самодвойственных булевых функций от n переменных.

Справедливо следующее предложение.

7.5. *Суперпозиция любого числа самодвойственных функций является снова самодвойственной функцией.*

Для доказательства этого предложения достаточно рассмотреть суперпозицию двух самодвойственных функций, например функций $z=f(y_1,\ldots, y_m)$ и $y_1=g(x_1,\ldots, x_n)$. Двойственной функцией для суперпозиции $v=f(g(x_1,\ldots,x_n), y_2,\ldots, y_m)$ этих функций будет функция $v^*=\overline{f(g(\bar{x}_1,\ldots\bar{x}_n), \bar{y}_2,\ldots,\bar{y}_m)}$. Ввиду самодвойственности функции g имеем $g(\bar{x}_1,\ldots,\bar{x}_n)=\bar{y}_1$. Но тогда $v^*=f(\bar{y}_1, \bar{y}_2,\ldots,\bar{y}_m)$ и ввиду самодвойственности функции f мы имеем $v^*=v$, что и завершает доказательство сформулированного в предложении 7.5 свойства суперпозиции самодвойственных функций.

Четвертый замечательный класс булевых функций составляют так называемые *линейные функции*. Булева функция $f(x_1, \ldots, x_n)$ называется *линейной*, если, после представления ее каноническим полиномом в алгебре Жегалкина, в этом полиноме не окажется членов, имеющих степени, выше чем первая, то есть если представляющий эту функцию полином имеет вид

$$a_0 + a_1x_1 + \ldots + a_nx_n.$$

Коэффициенты a_0, a_1, ..., a_n могут составлять любой набор значений 0 и 1, причем в силу однозначности определения канонического полинома (см. § 4) различным наборам коэффициентов будут соответствовать различные булевы функции. Таким образом, имеется точно 2^{n+1} линейных булевых функций от n переменных.

Из определения линейных функций непосредственно следует справедливость следующего предложения.

7.6. *Суперпозиция любого числа линейных булевых функций представляет собою снова линейную функцию.*

Пятый замечательный класс булевых функций составляют так называемые *монотонные функции*.

Для того чтобы определить монотонные функции, необходимо наряду с рассмотренной выше естественной упорядоченностью наборов ввести еще один вид упорядоченности.

Для любых двух наборов $a=(\alpha_1, \ldots, \alpha_n)$, $b=(\beta_1, \ldots, \beta_n)$, имеющих одинаковую размерность, отношение $a \leqslant b$ эквивалентно отношениям $\alpha_i \leqslant \beta_i$ для всех $i=1, \ldots, n$. При этом мы считаем, что $0 \leqslant 0$, $0 \leqslant 1$, $1 \leqslant 1$.

В силу этого определения $(0101) \leqslant (1101)$. В то же время наборы $a=(0101)$ и $b=(1010)$ несравнимы в том смысле, что для них не выполняется ни отношение $a \leqslant b$, ни отношение $b \leqslant a$.

Булева функция $f(x_1, \ldots, x_n)$ называется монотонной, если для любых наборов $a=(\alpha_1, \ldots, \alpha_n)$ и $b=(\beta_1, \ldots, \beta_n)$, таких, что $a \leqslant b$, имеет место неравенство $f(\alpha_1, \ldots, \alpha_n) \leqslant f(\beta_1, \ldots, \beta_n)$.

Среди булевых функций имеются как монотонные функции (например, функции x, $x_1 \vee x_2$), так и немонотонные функции (например, функции $\bar{x}$, $x_1 \bar{x}_2$). Справедливо следующее предложение.

7.7. *Суперпозиция любого числа монотонных булевых функций является в свою очередь монотонной функцией.*

Для доказательства предложения 7.7 достаточно, очевидно, рассмотреть суперпозицию двух монотонных функций, например функций $z=f(y_1, \ldots, y_m)$ и $y_1=g(x_1, \ldots, x_n)$. Возьмем два произвольных набора a_1 и a_2 значений переменных $x_1, \ldots, x_n, y_1, \ldots, y_m$ таких, что $a_1 \leqslant a_2$. Каждый из наборов a_1 и a_2 однозначно определяет соответствующие наборы (b_1, c_1 и b_2, c_2) значений переменных $x_1, \ldots, x_n$ и переменных $y_1, \ldots, y_m$. Ввиду определения неравенства для наборов, из неравенства $a_1 \leqslant a_2$ вытекает справедливость неравенств $b_1 \leqslant b_2$ и $c_1 \leqslant c_2$.

Обозначим через v_1 и v_2 значения суперпозиции функций f и g, т. е. $v=f(g(x_1, \ldots, x_n), y_2, \ldots, y_m)$ на наборах a_1 и a_2, а через g_1 и g_2 — значения функции $g(x_1, \ldots, x_n)$ на наборах b_1 и b_2 соответственно. Если к тому же наборы

c_1 и c_2 имеют вид $c_1=(\alpha_1, \ldots, \alpha_m)$, $c_2=(\beta_1, \ldots, \beta_m)$, то мы получим, очевидно:

$$v_1=f(g_1, \alpha_2, \ldots, \alpha_m),$$
$$v_2=f(g_2 \beta_2, \ldots, \beta_m).$$

Ввиду монотонности функции g справедливо неравенство $g_1 \leqslant g_2$. Поскольку из $c_1 \leqslant c_2$ вытекает, что $\alpha_2 \leqslant \beta_2, \ldots, \alpha_m \leqslant \beta_m$, то мы придем к неравенству

$$(g_1, \alpha_2, \ldots, \alpha_m) \leqslant (g_1, \beta_2, \ldots, \beta_m).$$

Из этого неравенства в силу монотонности функции f мы получаем неравенство $v_1 \leqslant v_2$, чем и завершается доказательство монотонности суперпозиции функций.

К числу немонотонных булевых функций относится отрицание $\bar{x}$. С другой стороны, оказывается, что всякая немонотонная функция содержит, в некотором смысле, в своем составе функцию отрицания. Более точно высказанное утверждение можно сформулировать следующим образом.

7.8. *В результате суперпозиции произвольной немонотонной функции $f(x_1, \ldots, x_n)$ с константами 0 и 1 может быть получена функция отрицания $\bar{x}_i$ одного из аргументов функции $f(x_1, \ldots, x_n)$.*

Доказательство предложения 7.8 мы начнем с того, что зафиксируем два набора a и b значений переменных $x_1, \ldots, x_n$ так, чтобы $a \leqslant b$ и $f(a)=1$, $f(b)=0$ (через $f(a)$ и $f(b)$ обозначены значения функции f на наборах a и b). Это всегда можно сделать ввиду немонотонности функции f. Согласно определению понятия неравенства наборов, в наборах a и b на некоторых из мест, соответствующих друг другу, могут стоять одинаковые компоненты. На остальных же местах в наборе a стоят нули, а в наборе b — единицы. Заменяя в каком-либо порядке эти нули в наборе a единицами, мы получим ряд так называемых с о с е д н и х д р у г д р у г у наборов $a_0=a, a_1, a_2, \ldots, a_n=b$ (*соседними* мы будем называть наборы, отличающиеся значением только одной из компонент).

Поскольку $f(a_0)=1$, а $f(a_n)=0$, то найдутся два соседних друг другу набора a_k и a_{k+1} таких, что $f(a_k)=1, f(a_{k+1})=0$. Предположим, что наборы a_k и a_{k+1} отличаются зна-

чением i-й компоненты. В силу принятого нами способа построения наборов $a_0, \ldots, a_n$ эта компонента обязательно равна 0 в наборе a_k и равна 1 в наборе a_{k+1}.

Следовательно,

$$a_k = (\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n),$$
$$a_{k+1} = (\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n).$$

Подставляя в функцию $f(x_1, \ldots, x_n)$ вместо переменной x_j константу $\alpha_j (j=1, 2, \ldots, i-1, i+1, \ldots, n)$, мы получим функцию одной переменной $g(x_i) = f(\alpha_1, \ldots, \alpha_{i-1}, x_i, \alpha_{i+1}, \ldots, \alpha_n)$.

В силу принятого способа выбора наборов a_k и a_{k+1} мы будем иметь:

$$g(0) = 1 \quad \text{и} \quad g(1) = 0.$$

Следовательно, $g(x_i) = \overline{x_i}$, и теорема доказана.

Теперь можно приступить к доказательству основной теоремы настоящего параграфа — *теоремы о функциональной полноте*. Доказательство этой теоремы будет вестись по методу А. В. Кузнецова и С. В. Яблонского [1]).

7.9. Теорема о функциональной полноте. *Для того чтобы система S булевых функций была функционально полной, необходимо и достаточно, чтобы эта система содержала хотя бы одну функцию, не сохраняющую константу* 0, *хотя бы одну функцию, не сохраняющую константу* 1, *хотя бы одну несамодвойственную функцию, хотя бы одну нелинейную функцию и хотя бы одну немонотонную функцию.*

Доказательство необходимости описываемых в этой теореме свойств системы булевых функций вытекает из предложений 7.3, 7.4, 7.5, 7.6 и 7.7, а также из того достаточно просто проверяемого факта, что тривиальные функции $f(x_1, \ldots, x_n) \equiv x_i$ $(i=1, \ldots, n)$ принадлежат каждому из введенных выше пяти замечательных классов булевых функций.

Доказательство достаточности описываемых свойств мы начнем с того, что зафиксируем прежде всего пять функ-

[1]) См. С. В. Яблонский, Функциональные построения в k-значной логике, Труды матем. ин-та им. В. А. Стеклова, т. 51, изд. АН СССР, М.— Л., 1958, стр. 5—142.

ций, принадлежащих заданной системе S: функцию f_1, не сохраняющую 0, функцию f_2, не сохраняющую 1, несамодвойственную функцию f_3, нелинейную функцию f_4 и немонотонную функцию f_5 (при этом не исключено, что некоторые из этих функций могут оказаться совпадающими).

Нашей целью является построение в виде суперпозиции функций системы S и переменных $x_1, \ldots, x_n$ любой заданной булевой функции $f(x_1, \ldots, x_n)$ этих переменных.

Если функция $f(x_1, \ldots, x_n)$ является константой, то условимся считать ее функцией не менее чем одной переменной. Таким образом, для образования суперпозиций у нас в запасе имеется всегда хотя бы одна переменная x_1.

Образуем суперпозицию функций f_1 и x_1, имеющую вид $g(x_1)=f_1(x_1, x_1, \ldots, x_1)$. Так как функция f_1 не сохраняет 0, то $g(0)=1$. Если теперь $g(1)=1$, то $g(x_1)\equiv 1$. Имея константу 1 и подставляя ее в функцию f_2, не сохраняющую единицу, мы получим вторую константу — 0.

Несколько сложнее обстоит дело в том случае, когда $g(1)=0$. Из равенств $g(0)=1$ и $g(1)=0$ мы заключаем, что $g(x_1)=\bar{x}_1$. Поскольку функция f_3 — несамодвойственная, найдется такой набор значений $(\alpha_1, \ldots, \alpha_k)$ ее аргументов, что $f_3(\alpha_1, \ldots, \alpha_k)=f_3(\bar{\alpha}_1, \ldots, \bar{\alpha}_k)$. Построим суперпозицию $h(x_1)$ функций f_3 и $\varphi_i(x_1)$, равных либо x_1, либо $\bar{x}_1$, по следующему правилу:

$$h(x_1)=f_3(\varphi_1(x_1), \ldots, \varphi_k(x_1)).$$

Здесь $\varphi_i(x_1)=x_1$, если $\alpha_i=0$, и $\varphi_i(x_1)=\bar{x}_1$, если $\alpha_i=1$. Тогда мы имеем:

$$h(0)=f_3(\varphi_1(0), \ldots, \varphi_k(0))=f_3(\alpha_1, \alpha_2, \ldots, \alpha_k)=$$
$$=f_3(\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_k)=f_3(\varphi_1(1), \ldots, \varphi_k(1))=h(1).$$

Следовательно, $h(0)=h(1)$, то есть, иначе говоря, $h(x)_1$ есть функция, тождественно равная константе 0 или константе 1. Образуя суперпозицию этой константы с построенной функцией $g(x_1)=\bar{x}_1$, мы получим вторую константу.

Итак, в любом случае из функций системы S и переменных $x_1, \ldots, x_n$ могут быть построены константы 0 и 1. Применяя теорему 7.8 к функции f_5, мы можем образовать

суперпозицию этих функций, дающую функцию $\bar{x}_i$, а значит, и все функции $\bar{x}_1, \ldots, \bar{x}_n$. Таким образом, уже построены все функции от одной переменной. Из теории нормальных форм, изложенной в § 5, следует, что заданная функция $f(x_1, \ldots, x_n)$ может быть представлена в виде суперпозиции (вообще говоря, многократной) уже найденных функций и функций x_1x_2, $x_1 \vee x_2$. Функция $x_1 \vee x_2$ в свою очередь может быть построена с помощью суперпозиции функций $\bar{x}_1$, $\bar{x}_2$ и x_1x_2 в соответствии со вторым правилом де Моргана (см. § 4).

Итак, для завершения доказательства теоремы о функциональной полноте нам остается получить функцию x_1x_2. Для этой цели возьмем нелинейную функцию $f_4(x_1, \ldots, x_p)$. Строя для этой функции канонический полином алгебры Жегалкина (см. § 4), мы обязательно найдем в этом полиноме член, содержащий произведение каких-либо двух переменных x_i и x_j, так как в противном случае функция f_4 оказалась бы линейной. Используя, в случае необходимости, суперпозицию функции f_4 с функциями x_i, x_j, x_1 и x_2, мы можем считать, что $x_i = x_1$, а $x_j = x_2$.

Все члены (канонические произведения) канонического полинома функции f_4 можно разбить на 4 группы следующим образом: в первую группу отнести все произведения, содержащие как x_1, так и x_2, во вторую отнести произведения, содержащие только x_1, но не x_2, в третью — произведения, содержащие только x_2, но не x_1, и, наконец, в четвертую группу — все произведения, не содержащие ни x_1, ни x_2 (некоторые из этих групп, кроме, разумеется, первой, могут оказаться и пустыми). После этого функция f_4 приобретает вид

$$f_4(x_1, \ldots, x_p) = x_1x_2\psi_1(x_3, \ldots, x_p) + x_1\psi_2(x_3, \ldots, x_p) + x_2\psi_3(x_3, \ldots, x_p) + \psi_4(x_3, \ldots, x_p).$$

Функция ψ_1 не равна тождественно нулю. Выбирая значения переменных $x_3, \ldots, x_p$ так, чтобы обратить эту функцию в 1, то есть осуществляя подходящую суперпозицию функции f_4 с константами мы придем к функции

$$\xi(x_1, x_2) = x_1x_2 + \alpha x_1 + \beta x_2 + \gamma,$$

где α, β, γ — константы 0 или 1. В нашем распоряжении для образования суперпозиций имеются функции x_1, x_2, $\bar{x}_1 = x_1 + 1$ и $\bar{x}_2 = x_2 + 1$. Мы можем, следовательно, образовать функцию

$$\xi(x_1+\beta,\ x_2+\alpha) = (x_1+\beta)(x_2+\alpha) + \alpha(x_1+\beta) + \\ + \beta(x_2+\alpha) + \gamma = x_1x_2 + \alpha x_1 + \beta x_2 + \alpha\beta + \alpha x_1 + \alpha\beta + \\ + \beta x_2 + \alpha\beta + \gamma = x_1x_2 + \delta,$$

где $\delta = \alpha\beta + \gamma$.

Если $\delta = 0$, то нами построена требуемая функция x_1x_2, если же $\delta = 1$, то построенная функция равна $x_1x_2 + 1 = \overline{x_1x_2}$. Образуя суперпозицию этой функции с уже построенным отрицанием $\bar{x}_1$, мы придем к требуемой функции x_1x_2. Тем самым теорема о функциональной полноте полностью доказана.

В случае наличия констант 0 и 1 в нашем распоряжении оказывается функция, не сохраняющая 0 (функция 1), функция, не сохраняющая 1 (функция 0), и несамодвойственная функция (любая из функций 0 или 1). Вместе с тем константы являются, очевидно, линейными и монотонными. Отсюда непосредственно (на основе теоремы о функциональной полноте) вытекает следующий результат.

7.10. Теорема об ослабленной функциональной полноте. *Для того чтобы система булевых функций была ослабленно функционально полной (полной при наличии констант), необходимо и достаточно, чтобы эта система содержала хотя бы одну нелинейную и хотя бы одну немонотонную функцию.*

Естественно называть функционально полную систему булевых функций *несократимой*, если из нее нельзя исключить ни одной функции так, чтобы оставшаяся после исключения система функций снова была бы функционально полной. Любую функционально полную систему булевых функций можно привести к несократимому виду, выбрасывая из нее лишние функции. Как вытекает из теоремы о функциональной полноте, в любой несократимой функционально полной системе булевых функций содержится не более 5 функций. В действительности, их число всегда может быть сокращено до 4, поскольку функция f, не сохраняющая 0, либо не сохраняет 1, либо

(если $f(1,1,\ldots,1)=1$) является несамодвойственной. Таким образом, кроме этой функции достаточно оставить в системе лишь три функции: нелинейную, немонотонную и либо функцию, не сохраняющую 1 (если $f(1,1,\ldots,1)=1$), либо несамодвойственную функцию (если $f(1,1,\ldots,1)=0$).

Вместе с тем нетрудно привести примеры функционально полных несократимых систем, состоящих из четырех функций. Такой системой является, например, система функций:

$$f_1=0,\quad f_2=1,\quad f_3=x_1x_2,\quad f_4=x_1+x_2+x_3.$$

Из этой системы нельзя исключить ни одной функции, поскольку функция f_1 является единственной из функций этой системы, не сохраняющей константу 1, f_2 — единственной функцией, не сохраняющей константу 0, f_3 — единственной нелинейной функцией, а f_4 — единственной немонотонной функцией (несамодвойственными здесь являются три функции: f_1, f_2 и f_3).

Таким образом, мы приходим к следующему предложению.

7.11. *Максимальное возможное число функций в несократимой функционально полной системе булевых функций равно четырем.*

Доказательство теоремы о функциональной полноте дает общий конструктивный прием синтеза любых правильных комбинационных схем в двоичном структурном алфавите из произвольных логических элементов, реализующих функционально полную систему булевых функций.

Сущность этого приема состоит в том, что сначала из данной системы логических элементов выбирают пять типов элементов, реализующих булевы функции, не сохраняющие 0 и 1, несамодвойственную, нелинейную и немонотонную функции. Затем, осуществляя правильную композицию элементов первых трех типов, методами, описанными в ходе доказательства теоремы о функциональной полноте, строят правильные комбинационные схемы, реализующие константы. Полученные две схемы принимаются за новые логические элементы. Из этих элементов и элементов, реализующих немонотонную и нелинейную функцию, строят комбинационные схемы, реализующие отрицание и конъюнкцию. После этого

строится дизъюнкция и применяется общий метод представления произвольных булевых функций с помощью совершенных нормальных форм (дизъюнктивных или конъюнктивных).

Тем самым получается одно из возможных решений общей задачи синтеза комбинационных схем в двоичном структурном алфавите, поскольку уже в классе правильных комбинационных схем можно реализовать любые выходные функции. Разумеется, это решение имеет скорее чисто теоретический, чем практический интерес, так как описанный метод синтеза приводит, как правило, к излишне усложненным схемам. Практически более целесообразные приемы синтеза комбинационных схем строятся обычно для каждой системы логических элементов отдельно.

При построении функционально полных систем логических элементов мы будем связывать с этими элементами понятия, относящиеся к реализуемым этими элементами булевым функциям. Например, нелинейными логическими элементами будем называть логические элементы, реализующие нелинейные булевы функции. Естественно не различать между собой логических элементов, отличающихся лишь способом обозначения входных сигналов. Таким образом, логические элементы находятся во взаимно однозначном соответствии не с самими булевыми функциями, а с классами булевых функций в смысле § 3.

Как отмечалось в конце § 3, имеется 12 различных классов булевых функций двух переменных и, следовательно, 12 различных логических элементов в двоичном структурном алфавите, имеющих не более двух элементарных входных сигналов каждый. Перечислим все эти элементы, исключая из рассмотрения т р и в и а л ь н ы й э л е м е н т, реализующий функции x_1 или x_2 (в электронных схемах тривиальным элементам соответствует просто кусок проводника, соединяющего два узла схемы).

Остальные 11 логических элементов представляют собою н у л е в о й и е д и н и ч н ы й элементы, реализующие функции-константы, и н в е р т о р, реализующий отрицание $\bar{x}$, с о в п а д е н и е, реализующее конъюнкцию (произведение) x_1x_2, р а з д е л е н и е, реализу-

ющее дизъюнкцию $x_1 \vee x_2$. Далее идут совпадения с одним и с двумя запретами, реализующие функции $\bar{x}_1 x_2$ и $\bar{x}_1 \bar{x}_2$ соответственно, разделение с одним запретом (функция $\overline{\bar{x}_1 \vee x_2}$), разделение с двумя запретами, или штрих Шеффера (функция $\overline{\bar{x}_1 \vee \bar{x}_2}$). Наконец, имеется еще элемент равнозначности, реализующий функцию $x_1 x_2 \vee \overline{x_1 x_2}$, и элемент неравнозначности (двоичный полусумматор), реализующий функцию $x_1 + x_2$.

Разобьем эти элементы на группы, имея целью облегчить применение к системам этих элементов теорему о функциональной полноте.

К группе *немонотонных элементов* относятся: инвертор, совпадение с одним запретом, совпадение с двумя запретами, разделение с одним запретом, разделение с двумя запретами, элемент равнозначности и элемент неравнозначности — всего 7 логических элементов.

К группе *нелинейных элементов* относятся: совпадение, разделение, совпадения и разделения с одним и с двумя запретами — всего 6 логических элементов.

К группе *несамодвойственных элементов* относятся: нулевой и единичный элементы, совпадение и разделение, совпадение и разделение с одним и с двумя запретами, а также элементы равнозначности и неравнозначности — всего 10 элементов (то есть все элементы, за исключением инвертора и тривиального элемента x).

К группе *элементов, не сохраняющих константу* 0, относятся: единичный элемент, инвертор, совпадение с двумя запретами, разделение с одним и с двумя запретами, а также элемент равнозначности — всего 6 элементов.

Наконец, к группе *элементов, не сохраняющих константу* 1, относятся: нулевой элемент, инвертор, совпадение с одним и с двумя запретами, разделение с двумя запретами и элемент неравнозначности — всего 6 элементов.

Эти результаты удобно свести в табл. III. 10, в которой знаком + отмечена принадлежность элемента к той или иной из введенных 5 групп.

Таблица III.10

Тип элемента	Функция	Группы элементов				
		немонотонные	нелинейные	несамодвойственные	не сохраняющие 0	не сохраняющие 1
Тривиальный . .	x	−	−	−	−	−
Нулевой	0	−	−	+	−	+
Единичный	1	−	−	+	+	−
Совпадение	x_1x_2	−	+	+	−	−
Разделение	$x_1 \vee x_2$	−	+	+	−	−
Инвертор	$\bar{x}$	+	−	−	+	+
Совпадение с запретом	$\bar{x}_1x_2$	+	+	+	−	+
Разделение с запретом	$\bar{x}_1 \vee x_2$	+	+	+	+	−
Совпадение с двумя запретами . .	$\bar{x}_1\bar{x}_2$	+	+	+	+	+
Разделение с двумя запретами . . .	$\bar{x}_1 \vee \bar{x}_2$	+	+	+	+	+
Элемент равнозначности . . .	$x_1x_2 \vee \bar{x}_1\bar{x}_2$	+	−	+	+	−
Элемент неравнозначности . . .	$x_1 + x_2$	+	−	+	−	+

Из табл. III.10 видно, что существуют два двухвходовых элемента, а именно совпадение и разделение с двумя запретами ($\bar{x}_1\bar{x}_2$, $\bar{x}_1 \vee \bar{x}_2$), каждый из которых составляет функционально полную систему логических элементов, то есть дает возможность построения на его основе комбинационных схем, реализующих любые функции в двоичном структурном алфавите. При наличии констант этим свойством обладают еще два логических элемента, а именно совпадение и разделение с одним запретом ($\bar{x}_1x_2$, $\bar{x}_1 \vee x_2$).

Построенная таблица позволяет находить также всевозможные другие функционально полные системы логических элементов. Признаком функциональной полноты системы элементов является, очевидно, наличие плюса в каждом столбце таблицы хотя бы для одного из составляющих систему элементов.

Легко устанавливается, например, функциональная полнота следующих систем логических элементов:

1) совпадение и инвертор;
2) разделение и инвертор;
3) совпадение, элемент неравнозначности и единичный элемент;
4) разделение с запретом и нулевой элемент;
5) совпадение с запретом и единичный элемент.

§ 8. Канонические уравнения структурных схем в двоичном структурном алфавите

В § 2 настоящей главы был решен вопрос о сведении общих проблем структурного синтеза автоматов к проблемам синтеза комбинационных схем. В случае двоичного структурного алфавита можно внести дополнительные уточнения в канонический метод структурного синтеза автоматов, рассмотренный в § 2.

Первое уточнение состоит в том, что в случае двоичного структурного алфавита в качестве запоминающих элементов используются обычно автоматы с двумя внутренними состояниями, обладающие полной системой переходов и полной системой выходов. Можно провести полную классификацию такого рода автоматов, получив при этом все наиболее употребительные в настоящее время запоминающие элементы логических схем автоматов.

Для каждого из полученных в результате проведенной классификации запоминающих элементов можно указать специальные приемы получения канонических уравнений автомата, более удобные по сравнению с общим приемом, описанным в § 2. В построении таких специальных приемов будет состоять второе уточнение, вносимое в настоящем параграфе в канонический метод синтеза автоматов.

Приступая к классификации автоматов с двумя внутренними состояниями, заметим прежде всего, что в таблицах переходов таких автоматов не может быть более четырех различных строк. Ясно также, что нет смысла различать входные сигналы в автомате Мура, если им соответствуют одинаковые строки в таблице переходов автомата.

Мы будем поэтому рассматривать лишь такие автоматы с двумя состояниями, число входных сигналов которых не превосходит четырех. Обозначим состояния автоматов через 0 и 1 и введем специальные буквенные обозначения для входных сигналов, которым соответствует та или иная фиксированная строка таблицы переходов автомата, при условии, что первый столбец таблицы обозначается состоянием 0, а второй — состоянием 1. Через x условимся обозначать входной сигнал, которому соответствует строка (00), через y — входной сигнал со строкой (11), через z — входной сигнал со строкой (01), через u — входной сигнал со строкой (10).

Полнота системы переходов в автомате означает, очевидно, что в каждом столбце таблицы переходов автомата должны встречаться символы всех его состояний. С учетом этого обстоятельства мы получаем следующие возможные типы запоминающих элементов.

1) Элемент с таблицей переходов III. 11, называемый обычно *элементом задержки*. Для этого элемента применяется система кодирования входных сигналов $x=0$, $y=1$.

Таблица III.11

	0	1
x	0	0
y	1	1

Таблица III.12

	0	1
z	0	1
u	1	0

Таблица III.13

	0	1
z	0	1
x	0	0
y	1	1

2) Элемент с таблицей переходов III. 12, называемый обычно *триггером со счетным входом*, или *счетчиком по модулю* 2. Сигнал z в этом случае отождествляется с нулевым сигналом, сигнал u — с единичным.

3) Элемент с таблицей переходов III. 13, называемый обычно *триггером с раздельными входами*. Для него употребляется система кодирования входных сигналов $z=(00)$, $x=(10)$, $y=(01)$ (элемент имеет два элементарных входных канала).

4) Элемент с таблицей переходов III. 14, который естественно назвать *комбинированным триггером* (он полу-

Таблица III.14

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

Таблица III.15

	0	1
z	0	1
u	1	0
x	0	0

Таблица III.16

	0	1
z	0	1
u	1	0
y	1	1

чается в результате комбинирования в одном элементе двух триггеров — триггера со счетным входом и триггера с раздельными входами). Для этого элемента употребляется следующая система кодирования входных сигналов:

$$z=(00),\quad u=(11),\quad x=(10),\quad y=(01).$$

Кроме перечисленных элементов, находящих широкое применение в схемах современных электронных цифровых автоматов, теоретически возможными (хотя и малоупотребительными на практике) являются еще три автомата с двумя состояниями, обладающие полной системой переходов:

5) Элемент с таблицей переходов III.15.
6) Элемент с таблицей переходов III.16.
7) Элемент с таблицей переходов III.17.

Таблица III.17

	0	1
u	1	0
x	0	0
y	1	1

Таблица III.18

	0	1
0	0	0
1	1	1

Таблица III.19

	0	1
0	0	1
1	1	0

Эти три элемента не имеют специальных названий.

Следует иметь в виду, что приведенная классификация основана на трактовке исследуемых элементов как абстрактных автоматов. Возможны поэтому различные модификации каждого из этих элементов с помощью применения систем кодирования входных сигналов, отличных

от рассмотренных выше. То же самое относится и к выходным сигналам; хотя из условия полноты системы выходов вытекает взаимно однозначное соответствие между состояниями автоматов и их выходными сигналами, система кодирования выходных сигналов может отличаться от принятой системы кодирования состояний.

Так, в различного рода триггерах (элементы 2, 3, 4) обычно имеется два элементарных выходных канала: сигнал на одном из этих каналов повторяет символ a состояния автомата, а на втором совпадает с его отрицанием $\bar{a}$. Что же касается элемента задержки, то он имеет один элементарный выходной канал, сигнал на котором совпадает с внутренним состоянием элемента.

Для описания упрощенных приемов получения канонических уравнений введем понятия абстрактной и структурной таблиц переходов автомата. *Абстрактной таблицей переходов* автомата мы будем называть обычную его таблицу переходов, в которой состояния автомата обозначаются просто буквами того или иного алфавита. В *структурной таблице переходов* состояния автомата обязательно кодируются наборами букв структурного алфавита (в рассматриваемом нами частном случае алфавита — в двоичном алфавите — кодирование производится наборами из нулей и единиц).

Входные сигналы автомата также предполагаются закодированными в двоичном структурном алфавите. Заменяя в таблице переходов автомата каждое его структурное (закодированное) состояние структурным входным сигналом памяти автомата, вызывающим указанный в таблице переход, мы приходим к понятию *таблицы возбуждений* автомата. Таблица возбуждений задает векторную функцию возбуждений автомата (см. § 2). По ней, а также по структурной таблице выходов автомата непосредственно строится его функциональная таблица (см. § 2), являющаяся исходным пунктом для построения комбинационной части схемы данного автомата.

Для двоичного структурного алфавита, при использовании описанных выше запоминающих элементов, существуют относительно простые приемы построения таблиц возбуждений. К описанию этих приемов мы сейчас и переходим.

8.1. *При использовании в качестве запоминающих элементов элементов задержки (с двумя состояниями) таблица возбуждений автомата совпадает с его структурной таблицей переходов.*

Чтобы доказать предложение 8.1, достаточно обратить внимание на то, что в элементе задержки переход из любого состояния в состояние 0 осуществляется с помощью входного сигнала 0, а переход из любого состояния в состояние 1 осуществляется с помощью входного сигнала 1. Такое свойство элемента задержки легко усмотреть непосредственно из его структурной таблицы переходов III.18.

8.2. *В случае использования в качестве запоминающих элементов триггеров со счетным входом для получения таблицы возбуждений автомата достаточно к каждой строке этой таблицы прибавить (покоординатно) по модулю 2 строку структурных состояний, последовательно обозначающих столбцы таблицы переходов.*

Действительно, если обозначить через x входной сигнал, переводящий триггер со счетным входом из состояния a_1 в состояние a_2, то из таблицы переходов триггера III.19 непосредственно усматриваем, что $a_1+x=a_2$. Прибавляя к обеим частям этого равенства по a_1 и используя соотношение (26) из § 4, мы приходим к соотношению $x=a_1+a_2$, чем и завершается доказательство предложения 8.2.

Например, если структурная таблица переходов автомата имеет вид, показанный в табл. III. 20, то в случае

Таблица III.20

	00	01	10
0	10	00	10
1	01	10	01

Таблица III.21

	00	01	10
0	10	01	00
1	01	11	11

использования в качестве запоминающих элементов триггеров со счетными входами мы приходим, пользуясь предложением 8.2, к таблице возбуждений III.21.

Для случая триггеров с раздельными входами и комбинированных триггеров положение усложняется благо-

даря тому, что для кодирования их состояний используется одна буква (цифра) двоичного алфавита, а для кодирования их структурных входных сигналов необходим уже двухбуквенный код.

В силу этого обстоятельства в структурной таблице переходов синтезируемого автомата с памятью каждый символ (0 или 1), обозначающий состояние, в которое переходит один из соответствующих триггеров, заменяется двумя символами x_1x_2, означающими сигналы, которые надо подать на входы триггера, чтобы вызвать переход из состояния a в состояние b (a — символ, обозначающий столбец структурной таблицы переходов, в котором находится выделенный символ b). При этом каждый из символов x_1x_2 может принимать кроме значений 0 и 1 еще и безразличное значение, которое мы будем обозначать черточкой. В случае, когда тот или иной символ принимает безразличное значение, это означает, что он может, в действительности, принять любое из двух значений 0 или 1 без нарушения правильности соответствующего перехода.

В рассматриваемом случае оказывается целесообразным записывать закон преобразования таблицы переходов в таблицу возбуждений с помощью системы подстановок вида $\binom{a}{b} \to (x_1x_2)$. Такая запись означает, что элемент b таблицы переходов, стоящий в столбце таблицы, обозначенном символом a, заменяется структурным входным сигналом x_1x_2. Назовем указанную систему подстановок *системой подстановок для получения таблицы возбуждений*.

Задание такой системы подстановок позволяет по любой заданной таблице переходов получить соответствующую ей таблицу возбуждений.

Таблица III.22

	00	10
0	10	10
1	00	10

Таблица III.23

	0	0	1	0
0	10	—1	0—	—1
1	—1	—1	0—	—1

Например, если таблица подстановок для получения таблицы возбуждений имеет вид $\binom{0}{0}\to(-1),\binom{0}{1}\to(10),\binom{1}{1}\to$ $\to(0-)$, то из таблицы переходов III. 22 мы получим таблицу возбуждений III. 23, что приводит в свою очередь к функциональной таблице III. 24.

Таблица III.24

	x_1	x_2	x_3	x_4
000	1	0	—	1
010	0	—	—	1
100	—	1	—	1
110	0	—	—	1

Таблица III.25

	0	1
00	0	1
10	0	0
01	1	1

Укажем теперь системы подстановок для триггеров с раздельными входами и для комбинированных триггеров.

8.3. *В случае использования в качестве запоминающих элементов триггеров с раздельными входами система подстановок для получения таблицы возбуждений имеет вид*

$$\binom{0}{0}\to(-0),\quad \binom{0}{1}\to(01)\quad \binom{1}{0}\to(10),\quad \binom{1}{1}\to(0-).$$

Доказательство предложения 8.3 вытекает из рассмотрения структурной таблицы переходов триггера с раздельными входами (табл. III. 25).

Необходимо подчеркнуть, что при выбранной системе кодирования в любой паре (x_1x_2), составляющей структурный входной сигнал триггера с раздельными входами, первый элемент пары (x_1) означает элементарный входной сигнал на нулевом входном канале триггера (передача сигнала 1 по этому каналу устанавливает триггер в состояние 0), а второй элемент пары (x_2) означает элементарный входной сигнал на единичном входном канале триггера (передача сигнала 1 по этому каналу устанавливает триггер в 1).

8.4. *В случае использования в качестве запоминающих элементов комбинированных триггеров первого рода система подстановок для получения таблицы возбуждений*

имеет вид

$$\binom{0}{0} \to (-0),\ \binom{0}{1} \to (-1),\ \binom{1}{0} \to (1-),\ \binom{1}{1} \to (0-).$$

Доказательство предложения 8.4 вытекает непосредственно из рассмотрения структурной таблицы переходов комбинированного триггера первого рода (табл. III.26).

Таблица III.26

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

Таблица III.27

	0	1
000	0	1
100	0	0
010	1	1
001	1	0

При выбранной системе кодирования первый элемент структурного входного сигнала триггера означает сигнал на его нулевом входном канале, а второй элемент — сигнал на его единичном входном канале. Одновременная передача сигналов 1 по обоим входным каналам вызывает работу триггера в режиме счетного входа; иначе говоря, в этом случае триггер работает так, как работал бы триггер со счетным входом при передаче по его единственному входному каналу сигнала 1.

Заметим, что на практике для комбинированного триггера часто используют три входных канала, двумя из которых являются каналы для установки триггера в состояния 0 и 1, а третьим — канал, отличающийся тем, что передача по нему сигнала 1 вызывает работу триггера в режиме счетного входа; этот последний канал естественно называть с ч е т н ы м. При этом сигнал 1 в каждый данный момент может передаваться не более чем по одному из трех указанных каналов. Если первый элемент структурного входного сигнала такого триггера обозначает сигнал на его нулевом входном канале, второй элемент — сигнал на его единичном входном канале, а третий элемент — сигнал на его счетном входном канале, то структурная таблица переходов рассматриваемого триггера будет иметь вид, представленный в табл. III.27.

Комбинированные триггеры с тремя входными каналами,— при условии, что входные сигналы кодируются по рассмотренной выше системе — мы будем называть *комбинированными триггерами второго рода* (за рассматривавшимися ранее комбинированными триггерами с двумя входными каналами мы закрепим название *комбинированных триггеров первого рода*).

8.5. *Система подстановок для получения таблицы возбуждений в случае комбинированных триггеров второго рода может быть записана следующим образом:*

$$\binom{0}{0} \to (-00),\ \binom{0}{1} \to (0\delta\bar{\delta}),\ \binom{1}{0} \to (\varepsilon 0\bar{\varepsilon}),\ \binom{1}{1} \to (0-0),$$

где через δ и ε обозначен любой из символов 0 или 1, а через $\bar{\delta}$ и $\bar{\varepsilon}$ — их отрицания.

Указанная система подстановок получается непосредственно из рассмотренной выше структурной таблицы переходов комбинированного триггера второго рода.

Следует отметить, впрочем, что в задачах синтеза автоматов при использовании комбинированных триггеров второго рода эти триггеры предпочитают рассматривать в одних случаях как триггеры со счетным входом, а в других — как триггеры с раздельными входами, применяя при построении их таблиц возбуждений предложения 8.2 и 8.3, а не предложение 8.5.

Построение таблицы возбуждений автомата является главной составной частью процесса получения канонических уравнений синтезируемого автомата. Эти уравнения могут быть получены, например, в результате записей функций возбуждения, представленных таблицей возбуждений автомата, в совершенной нормальной форме (дизъюнктивной или конъюнктивной). Мы будем употреблять для этой цели совершенную дизъюнктивную нормальную форму.

Рассмотрим в качестве примера получение канонических уравнений для абстрактного автомата Мура A, синтезированного в примере 2, рассматривавшегося в § 8 главы 2. Отправляясь от отмеченной таблицы переходов этого автомата A (табл. III. 28) и применяя естественную систему кодирования его состояний

Таблица III.28

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

в двоичном алфавите (состоящую в том, что в качестве кода состояния употребляется просто номер этого состояния в двоичной системе счисления), мы получим структурную таблицу переходов P автомата (табл. III.29).

Таблица III.29

	000	001	010	011	100	101	110	111
0	010	111	111	110	101	110	111	111
1	001	100	011	111	111	111	111	111

Условимся через x обозначать входной сигнал автомата A, а через (z_1, z_2, z_3) — его структурное (закодированное) состояние. Рассмотрим случай, когда в качестве запоминающих элементов выбираются элементы задержки, и обозначим через (u_1, u_2, u_3) структурный входной сигнал памяти автомата A (u_i — входной сигнал i-го элемента памяти автомата, где $i=1, 2, 3$). В силу предложения 8.1 таблица возбуждений автомата совпадает с его структурной таблицей переходов.

Функции возбуждений автомата $f_i(x, z_1, z_2, z_3)$ задают зависимость входных сигналов в каждый момент от входного сигнала и структурного состояния автомата A:

$$u_i = f_i(x, z_1, z_2, z_3) \qquad (i = 1, 2, 3). \qquad (1)$$

Эти зависимости и являются искомыми каноническими уравнениями автомата A. Для явного выражения правых частей уравнений (1) представим их в совершенной дизъ-

юнктивной нормальной форме, фиксируя наборы, на которых значения u_l равны единице. Таким образом, мы легко найдем, например, выражение для u_1:

$$u_1 = \bar{x}\bar{z}_1\bar{z}_2z_3 \vee \bar{x}\bar{z}_1z_2\bar{z}_3 \vee \bar{x}z_1\bar{z}_2z_3 \vee x\bar{z}_1\bar{z}_2\bar{z}_3 \vee x\bar{z}_1\bar{z}_2z_3 \vee$$
$$\vee x\bar{z}_1z_2\bar{z}_3 \vee x\bar{z}_1z_2z_3 \vee xz_1\bar{z}_2z_3 \vee$$
$$\vee xz_1\bar{z}_2z_3 \vee xz_1\bar{z}_2\bar{z}_3 \vee xz_1z_2\bar{z}_3 \vee xz_1z_2z_3.$$

Найденное выражение достаточно громоздко. Используя методы минимизации, излагаемые в следующем параграфе, его можно, правда, существенно упростить. Сейчас, однако, мы не будем заниматься его упрощением, а укажем лишь один более простой метод задания совершенных дизъюнктивных нормальных форм, часто применяющийся на практике. Суть этого метода заключается в том, что вместо явного выписывания конституент единицы, составляющих дизъюнктивную нормальную форму, выписываются лишь номера этих конституент, то есть номера тех наборов, на которых данные конституенты обращаются в единицу. При этом, разумеется, необходимо фиксировать некоторый определенный порядок следования переменных, от которых зависят функции возбуждения.

Мы условимся считать первыми переменные, составляющие структурный входной сигнал автомата, причем сами эти переменные выписываются в том порядке, в котором они вписаны в структурной таблице переходов автомата. После этих переменных выписываются (тоже в том порядке, который принят в структурной таблице переходов) переменные, составляющие структурное состояние автомата.

Используя принятую систему обозначений, канонические уравнения автомата в рассматриваемом случае можно записать в следующем виде:

$$\left.\begin{aligned} u_1 &= [1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 13, 14, 15], \\ u_2 &= [0, 1, 2, 3, 5, 6, 7, 10, 11, 12, 13, 14, 15], \\ u_3 &= [1, 2, 4, 6, 7, 8, 10, 11, 12, 13, 14, 15]. \end{aligned}\right\} \quad (2)$$

Найдем теперь канонические уравнения того же самого автомата в предположении, что в качестве запоминающих

элементов используются триггеры со счетными входами. Применение предложения 8.2 приведет к таблице возбуждений автомата A III.30; а это значит, что канонические

Таблица III.30

	000	001	010	011	100	101	110	111
0	010	110	101	101	001	011	001	000
1	001	101	001	100	011	010	001	000

уравнения (в сокращенной записи) представятся в виде

$$\left.\begin{aligned} u_1 &= [1,\ 2,\ 3,\ 9,\ 11], \\ u_2 &= [0,\ 1,\ 5,\ 12,\ 13], \\ u_3 &= [2,\ 3,\ 4,\ 5,\ 6,\ 8,\ 9,\ 10\ \ 12,\ 14]. \end{aligned}\right\} \quad (3)$$

В случае использования в качестве запоминающих элементов триггеров с раздельными входами мы с помощью предложения 8.3 можем найти таблицу возбуждений рассматриваемого автомата A (табл. III. 31).

Таблица III.31

	000	001	010	011	100	101	110	111
0	—001—0	01010—	010—01	010—10	0— —001	0—0110	0—0—01	0—0—0—
1	—0—001	01—010	—00—01	010—0—	0— 0101	0—010—	0—0—01	0—0—0—

В этом случае структурный входной сигнал памяти автомата имеет не три компоненты, как в ранее разобранных случаях, а шесть — по две компоненты на каждый триггер. Обозначим через u_i сигнал на нулевом входном канале, а через v_i — сигнал на единичном входном канале i-го триггера ($i=1, 2, 3$). Тогда согласно принятым условиям (переменный) структурный входной сигнал памяти автомата будет иметь вид $(u_1v_1u_2v_2u_3v_3)$. Рассматриваемый

случай отличается также наличием безразличных значений переменных, обозначаемых черточками.

Поскольку в качестве этих значений могут быть выбраны как 0, так и 1, соответствующие им конституенты единицы могут быть по нашему желанию либо введены в совершенную дизъюнктивную нормальную форму, либо исключены из нее. Мы будем называть такие конституенты безразличными и при выписывании канонических уравнений автомата описанного выше типа помещать их в круглые скобки.

Следует подчеркнуть, что введение в совершенную дизъюнктивную нормальную форму некоторых безразличных конституент может способствовать, как это будет показано в следующей главе, более полной минимизации этой формы. Именно поэтому целесообразно в записи канонических уравнений на первых порах сохранять все безразличные конституенты с тем, чтобы в результате последующего анализа отбросить те из них, которые не способствуют минимизации, и включить в совершенную дизъюнктивную нормальную форму безразличные конституенты, способствующие минимизации этой формы

Применяя введенный способ обозначений, мы получим — для случая использования в качестве запоминающих элементов триггеров с раздельными входами — следующие канонические уравнения автомата A:

$$\left.\begin{aligned}
u_1 &= [(0,\ 8,\ 10)],\\
v_1 &= [1,\ 2,\ 3,\ 9,\ 11,\ (4,\ 5,\ 6,\ 7,\ 12,\ 13,\ 14,\ 15)],\\
u_2 &= [(4,\ 8,\ 9)],\\
v_2 &= [0,\ 1,\ 5,\ 12,\ 13,\ (2,\ 3,\ 6,\ 7,\ 10,\ 11,\ 14,\ 15)],\\
u_3 &= [3,\ 5,\ 9\ (0)],\\
v_3 &= [2,\ 4,\ 6,\ 8,\ 10,\ 12,\ 14,\ (1,\ 7,\ 11,\ 13,\ 15)].
\end{aligned}\right\} \quad (4)$$

В уравнениях с левыми частями u_1 и u_2 все конституенты, стоящие в правых частях, являются безразличными. Поэтому можно принять, что $u_1 = u_2 = 0$. Для остальных переменных получаются уравнения с нетривиальными правыми частями.

ГЛАВА IV

МИНИМИЗАЦИЯ БУЛЕВЫХ ФУНКЦИЙ

§ 1. Сокращенные и минимальные дизъюнктивные нормальные формы

Под *минимизацией булевой функции* понимается нахождение наиболее простого представления этой функции в виде суперпозиции функций, составляющих какую-нибудь фиксированную функционально полную систему S булевых функций. Наиболее простым считается обычно представление, содержащее наименьшее возможное число суперпозиций. Понимаемая в таком смысле задача минимизации булевых функций всегда разрешима, поскольку мы будем рассматривать лишь конечные функционально полные системы. В этом случае можно организовать последовательный перебор сначала всех функций системы S, затем всевозможных их парных суперпозиций, тройных суперпозиций и т. д., пока не будет получена суперпозиция, равная заданной булевой функции. Разумеется, такой метод последовательного перебора всех представлений является весьма громоздким и не может применяться на практике без существенных усовершенствований. На практике пользуются обычно специальными приемами минимизации, разрабатываемыми применительно к каждой отдельной функционально полной системе S булевых функций. Наиболее детально такие приемы разработаны для случая, когда система S состоит из дизъюнкции, конъюнкции и отрицания. В этом случае задача минимизации булевой функции сводится к нахождению выражения булевой алгебры, представляющего заданную функцию и построенного с помощью наименьшего возможного числа операций.

Мы ограничимся еще более частной задачей минимизации, а именно задачей представления заданной булевой функции дизъюнктивной нормальной формой, содержащей наименьшее возможное число букв. Условимся называть эту задачу *канонической задачей минимизации булевых функций*. Каноническая задача минимизации играет основную роль при синтезе комбинационных схем в случае наличия на входах этих схем наряду с любым данным сигналом x также и сигнала $\bar{x}$. Подобная ситуация имеет, например, место при синтезе цепей обратной связи автоматов, использующих в качестве запоминающих элементов одну из разновидностей триггеров, описанных в § 8 предыдущей главы.

При решении канонической задачи минимизации существенную роль играют понятия *импликанты* и *простой импликанты* булевой функции [1]).

1.1. *Булева функция $g=g(x_1,\ldots,x_n)$ называется импликантой булевой функции $f=f(x_1,\ldots,x_n)$, если на любом наборе значений переменных $x_1,\ldots,x_n$, на котором значение функции g равно единице, значение функции f также равно единице.*

Простой импликантой функции f называется всякое элементарное произведение $g=\tilde{x}_{i_1}\tilde{x}_{i_2}\ldots\tilde{x}_{i_k}$, являющееся импликантой функции f и такое, что никакая его собственная часть (то есть произведение, получающееся из произведения g выбрасыванием одного или нескольких сомножителей $\tilde{x}_{i_\mu}$) уже не является импликантой функции f.

Для правильного понимания сформулированного определения необходимо напомнить, что, согласно принятому в предыдущей главе условию, элементарное произведение, состоящее из пустого множества сомножителей, принимается равным единице. Таким образом, константа 1 является собственной частью любого элементарного произведения. В силу сделанного замечания состоящее из одного сомножителя (буквы) элементарное произведение, являясь импликантой какой-либо булевой функции, не равной тождественно единице, будет всегда простой импликантой этой функции.

[1]) См. W. V. Quine, The problem of simplifying of truth functions. Amer. Math. Monthly, v. 59, № 8, 1952, p. 521—531.

Для произвольных импликант имеет место следующее предложение, справедливость которого вытекает непосредственно из определения 1.1.

1.2. *Дизъюнкция любого множества импликант одной и той же булевой функции является импликантой этой функции.*

Предположим, что на некотором наборе α значений переменных некоторая булева функция f принимает значение, равное единице. Условимся говорить, что эта единица накрывается *импликантой* g функции f, если на наборе α импликанта g также обращается в единицу. Условимся называть систему S импликант функции f *полной*, если любая единица из таблицы значений функции f накрывается хотя бы одной импликантой системы S.

Справедливо следующее предложение.

1.3. *Дизъюнкция всех импликант булевой функции f, входящих в какую-либо полную систему S импликант этой функции, совпадает с функцией f.*

В самом деле, в силу предложения 1.2 и определения полноты системы импликант, дизъюнкция g всех импликант системы S является импликантой функции f, накрывающей все единицы этой функции. Будучи импликантой функции f, функция g должна обращаться в нуль на всех наборах, на которых обращается в нуль функция f. Вместе с тем, на любом наборе, на котором функция f обращается в единицу, функция g также должна обращаться в единицу, так как она накрывает все единицы функции f. Следовательно, значения функций f и g совпадают на всех наборах, то есть функции f и g равны друг другу, что и требовалось доказать.

Для данной булевой функции f можно построить, вообще говоря, не одну, а много различных полных систем импликант. Одной из таких систем является система, состоящая лишь из самой функции f. Другой полной системой импликант функции f является, очевидно, система всех конституент единицы, представляющих собою импликанты данной функции. Эти две системы различны для функций, таблицы значений которых насчитывают более одной единицы.

Для минимизации булевых функций существенное значение имеет следующее предложение.

1.4. *Система всех простых импликант булевой функции является полной системой.*

В самом деле, рассмотрим произвольную булеву функцию f. Если f — тождественный нуль, то доказывать нечего. Если же функция f отлична от нуля, то рассуждение поведем так. Рассмотрим произвольный набор, на котором эта функция принимает значение, равное 1. Соответствующая этому набору конституента единицы $K=\tilde{x}_1\tilde{x}_2\ldots\tilde{x}_n$ является, очевидно, импликантой функции f. Некоторая часть p элементарного произведения K (быть может, совпадающая с ним самим) обязательно будет простой импликантой функции f. Действительно, если элементарное произведение само не является простой импликантой, то некоторая его собственная часть K_1 будет импликантой функции f. Если импликанта K_1 также непростая, то можно выделить ее собственную часть K_2, снова являющуюся импликантой функции f. Продолжая подобным образом далее, придем к простой импликанте p.

Благодаря способу своего определения, простая импликанта p обращается в единицу на наборе α. Следовательно, для любой единицы из таблицы значений булевой функции f найдется накрывающая ее простая импликанта этой функции. Предложение 1.4 тем самым полностью доказано.

Из предложений 1.3 и 1.4 непосредственно вытекает следующий результат.

1.5. *Дизъюнкция всех простых импликант булевой функции совпадает с этой функцией.*

Особого рассмотрения в высказанном предложении требует лишь случай функции, тождественно равной нулю.

В этом случае функция не имеет, очевидно, ни одной простой импликанты. Однако, согласно принятому в предыдущей главе условию, дизъюнкция пустого множества членов принимается равной нулю. Так что теорема остается справедливой и в этом случае.

Введем следующее определение.

1.6. *Дизъюнкция всех простых импликант булевой функции называется сокращенной дизъюнктивной нормальной формой этой функции.*

Сокращенная дизъюнктивная нормальная форма является, вообще говоря, более экономным способом представления булевой функции, чем совершенная дизъюнктивная нормальная форма. Однако в большинстве случаев она допускает дальнейшие упрощения за счет того, что некоторые из простых импликант могут поглощаться дизъюнкциями других простых импликант. Например, в сокращенной дизъюнктивной нормальной форме $f=\bar{x}\bar{y}\vee\bar{x}\bar{y}\vee xz\vee yz$ простая импликанта yz поглощается дизъюнкцией остальных членов формы, так что

$$\bar{x}\bar{y}\vee x\bar{y}\vee xz\vee yz=\bar{x}\bar{y}\vee x\bar{y}\vee xz.$$

Справедливость этого утверждения легко проверить, замечая, что импликанта yz обращается в единицу лишь на двух наборах $\alpha=(011)$ и $\beta=(111)$. В то же время на наборе α обращается в единицу импликанта $\bar{x}\bar{y}$, а на наборе β — импликанта xz.

В связи со всем сказанным естественно ввести следующее определение.

1.7. *Система S простых импликант булевой функции f называется приведенной, если эта система полна, а никакая ее собственная часть не является полной системой импликант функции f. Дизъюнкция всех простых импликант, составляющих систему S, называется приведенной или тупиковой дизъюнктивной формой функции f.*

Из предложения 1.3 мы непосредственно выводим, что всякая тупиковая дизъюнктивная нормальная форма любой булевой функции совпадает с этой функцией.

Вместе с тем, в отличие от сокращенной дизъюнктивной нормальной формы, однозначно определяемой функцией f, для одной и той же булевой функции может существовать несколько различных тупиковых дизъюнктивных нормальных форм. Так, например, функция, задаваемая сокращенной д. н. ф. (дизъюнктивной нормальной формой) $f=\bar{x}\bar{y}\vee x\bar{y}\vee xz\vee yz$, имеет две тупиковых д. н. ф.: $f_1=\bar{x}\bar{y}\vee x\bar{y}\vee xz$ и $f_2=\bar{x}\bar{y}\vee x\bar{y}\vee yz$.

Естественно оценивать сложность дизъюнктивных нормальных форм общим числом входящих в них букв, считая различными буквами вхождения одной и той же буквы в различные дизъюнктивные члены формы. В соответствии

с этим условием мы должны считать, что рассмотренная выше форма f содержит 8 букв, а формы f_1 и f_2— по 6 букв каждая.

Введем следующее определение.

1.8. *Минимальной дизъюнктивной нормальной формой булевой функции называется дизъюнктивная нормальная форма* (д.н.ф.) *этой функции, состоящая из наименьшего числа букв.*

Нетрудно видеть, что минимальная д. н. ф. f любой булевой функции представляет собою дизъюнкцию некоторых простых импликант этой функции. В самом деле, представим форму f в виде $p \vee P$, где p — одно из составляющих форму f элементарных произведений, а P — дизъюнкция всех остальных членов этой формы. Отождествив форму f с представляемой ею булевой функцией, мы сразу заметим, что p является импликантой этой функции. Действительно, из определения дизъюнкции непосредственно следует, что функция f равна 1 во всех точках, в которых p равняется 1. Аналогично, функция P также является импликантой функции f. Поскольку $f=p \vee P$, то система импликант p и P — полная.

Если элементарное произведение p не является п р о с т о й импликантой функции f, то найдется элементарное произведение q, составленное лишь из части множителей (букв), составляющих произведение p, и являющееся также импликантой функции f. Ввиду свойств произведения, импликанта q обращается в единицу на всех наборах, на которых равна единице импликанта p. Но тогда, очевидно, система импликант q и P является полной системой функции f и, в силу теоремы 1.3, оказывается, что $f=q \vee P$.

Выражение $q \vee P$ является поэтому дизъюнктивной нормальной формой функции f, причем формой, содержащей меньшее число букв, чем исходная д.н.ф. $p \vee P$. Поскольку эта последняя форма была минимальной, мы приходим к противоречию, источником которого является, очевидно, предположение о том, что импликанта p — непростая. Ввиду произвольности выбора p, мы приходим к выводу, что минимальная д. н. ф. любой булевой функции представляет собою дизъюнкцию некоторого множества S ее простых импликант. Из минимальности формы

непосредственно следует также, что система S простых импликант непременно приведенная. Таким образом, мы приходим к следующему предложению [1]).

1.9. *Любая минимальная дизъюнктивная нормальная форма булевой функции представляет собою дизъюнкцию некоторой приведенной системы простых импликант этой функции. Иными словами, всякая минимальная д. н. ф. является тупиковой д.н.ф.*

Заметим, что одна и та же булева функция может обладать несколькими различными минимальными д.н.ф. Существуют также тупиковые д.н.ф., не являющиеся минимальными д. н. ф. Соответствующие примеры будут приведены в последующих параграфах. Нетрудно проверить, впрочем, что рассмотренные выше тупиковые д. н. ф. f_1 и f_2 являются вместе с тем и минимальными.

Теорема 1.9 составляет основу общей схемы решения канонической задачи минимизации булевых функций. Из теоремы 1.9 вытекает, что эта схема должна состоять из двух основных этапов. На п е р в о м этапе находятся все простые импликанты заданной булевой функции или, что фактически то же самое — ее сокращенная д.н.ф. На в т о р о м этапе ищутся приведенные системы простых импликант, строятся тупиковые д.н.ф., из числа которых отбираются минимальные д.н.ф.

В последующих параграфах мы опишем несколько методов решения задач первого этапа, а сейчас остановимся более подробно на втором этапе. Основным аппаратом для решения задач второго этапа служит так называемая *импликантная таблица* булевой функции [2]).

Импликантная таблица любой данной булевой функции f представляет собою прямоугольную таблицу с двумя входами, строки которой обозначаются различными простыми импликантами функции f, а столбцы — наборами значений переменных (или соответствующими им конституентами единицы), на которых функция обращается в единицу. Если какая-либо простая импликанта обращается в единицу на некотором наборе α, обозначающем

[1]) W. V. Q u i n e, The problem of simplifying of truth functions. Amer. Math. Monthly, v. 59, № 8, 1952, p. 521—531.

[2]) Импликантные таблицы булевых функций введены К в а й н о м. См. его статью, указанную в примечании выше.

какой-либо столбец импликантной таблицы, то на пересечении p-й строки и α-го столбца таблицы ставится звездочка. Если же импликанта p на наборе α обращается в нуль, то пересечение p-й строки и α-го столбца в импликантной таблице оставляется пустым.

При обозначении столбцов импликантной таблицы конституентами единицы имеет силу, как это очевидно, следующее правило заполнения импликантной таблицы.

1.10. *На пересечении p-й строки и K-го столбца импликантной таблицы тогда и только тогда ставится звездочка, когда импликанта p составляет некоторую часть конституенты K* (быть может, совпадающую со всей конституентой).

Рассмотрим в качестве примера импликантную таблицу для булевой функции $f=\bar{x}\bar{y}\vee x\bar{y}\vee xz\vee yz$, уже рассматривавшейся по другому поводу выше. Эта функция имеет четыре простые импликанты $\bar{x}\bar{y}$, $x\bar{y}$, xz, yz и обращается в единицу на пяти наборах (010), (011), (100), (101), (111), которым соответствуют конституенты единицы $\bar{x}\bar{y}\bar{z}$, $\bar{x}yz$, $x\bar{y}\bar{z}$, $x\bar{y}z$, xyz. В соответствии с приведенным выше правилом, импликантная таблица рассматриваемой функции будет иметь следующий вид:

Таблица IV.1

	$\bar{x}\bar{y}\bar{z}$	$\bar{x}yz$	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz
$\bar{x}\bar{y}$	*	*			
$x\bar{y}$			*	*	
xz				*	*
yz		*			*

Выделение приведенных систем простых импликант может быть проведено непосредственно по импликантной таблице. Для этой цели нужно выбрать минимальные системы строк таблицы так, чтобы для каждого столбца среди выбранных строк нашлась хотя бы одна строка, содержащая в этом столбце звездочку. Ясно, что в любую

такую систему должны входить все строки, в которых содержится звездочка, являющаяся единственной звездочкой в своем столбце. Простые импликанты, обозначающие строки с указанным свойством, составляют так называемое *ядро булевой функции*. Они входят во всякую приведенную систему простых импликант этой функции. В рассмотренном выше случае ядро составляют простые импликанты $\bar{x}y$ и $x\bar{y}$ (им соответствуют «однозвездочные» столбцы $\bar{x}yz$ и $x\bar{y}\bar{z}$).

Простые импликанты, входящие в ядро, накрывают часть единиц булевой функции. Применительно к импликантной таблице более удобно говорить, что простые импликанты накрывают соответствующие этим единицам конституенты функции. Ясно, что накрытыми окажутся все те и только те конституенты, которые обозначают столбцы импликантной таблицы, содержащие звездочки в строках, обозначения которых принадлежат ядру. В нашем случае это будут конституенты $\bar{x}y\bar{z}$, $\bar{x}yz$, $x\bar{y}z$, $xy\bar{z}$.

Исключая из импликантной таблицы столбцы, обозначенные уже накрытыми конституентами, мы методом перебора сможем найти минимальные накрытия оставшихся конституент. В рассматриваемом нами случае единственная остающаяся конституента xyz может быть накрыта как импликантой xz, так и импликантой yz, в соответствии с чем мы получаем две приведенные системы простых импликант:

$$(\bar{x}y,\ x\bar{y}\ \ xz),\quad (\bar{x}y,\ x\bar{y},\ yz)$$

и две тупиковые д.н.ф.:

$$f_1 = \bar{x}y \vee x\bar{y} \vee xz,$$
$$f_2 = \bar{x}y \vee x\bar{y} \vee yz.$$

Следует отметить, что метод перебора всевозможных минимальных накрытий конституент непосредственно по импликантной таблице оказывается практически применимым для относительно простых импликантных таблиц; он может быть применен также в том случае, когда нужно найти не все, а лишь одно из минимальных накрытий (приведенных систем простых импликант). Для нахождения всех минимальных накрытий в случае сложных

таблиц можно применять естественный алгебраический метод, предложенный Петриком [1]).

Суть этого метода состоит в том, что по импликантной таблице булевой функции f строится некоторое выражение, называемое *конъюнктивным представлением* этой таблицы. Для этой цели производится обозначение всех простых импликант функции f различными буквами (обычно для этой цели используют большие латинские буквы $A, B, C, \ldots$), после чего для каждого столбца α импликантной таблицы строится дизъюнкция $q_\alpha = A_{l_1} \vee A_{l_2} \vee \ldots \vee A_{l_k}$ всех букв, обозначающих строки импликантной таблицы, на пересечении которых с рассматриваемым столбцом α стоят звездочки. Беря произведение построенных дизъюнкций q_α для всех столбцов α импликантной таблицы, мы придем к конъюнктивному представлению импликантной таблицы.

Обозначая в рассмотренном выше примере импликанту $\bar{x}y$ буквой A, импликанту $x\bar{y}$ — буквой B, импликанту xz — буквой C, а импликанту yz — буквой D, мы получим следующее конъюнктивное представление импликантной таблицы:

$$\varphi = A\,(A \vee D)\,B\,(B \vee C)\,(C \vee D).$$

Если в конъюнктивном представлении импликантной таблицы в соответствии с законом дистрибутивности раскрыть все скобки, то возникнет некоторая дизъюнкция произведений букв, обозначающих простые импликанты. Назовем эту дизъюнкцию *дизъюнктивным представлением* исходной импликантной таблицы. Как непосредственно вытекает из закона раскрытия скобок, в каждое элементарное произведение, составляющее дизъюнктивное представление, обязательно входит символ простой импликанты, накрывающей конституенту единицы, соответствующую любому наперед заданному столбцу импликантной таблицы. Иначе говоря, *простые импликанты, символы которых входят в любой фиксированный терм дизъюнктивного представления импликантной таблицы булевой функции f, составляют полную систему простых импли-*

[1]) S. R. Petrick, A direct decomposition of the irredundant forms of a boolean function from the set of prime-implicants. Tech. reports Air Force Cambridge Research Center, 1956, p. 56—110.

кант функции f. Назовем эту систему системой, соответствующей данному терму.

Нетрудно видеть, что при пользовании одним лишь дистрибутивным законом без каких-либо дополнительных упрощений в дизъюнктивном представлении импликантной таблицы функции f будут встречаться термы, которым соответствуют любые заданные полные системы простых импликант функции f. Для получения одних лишь приведенных систем простых импликант в дизъюнктивном представлении импликантной таблицы нужно, очевидно, произвести исключение тех термов, которые содержат в своем составе другие термы того же представления. Например, при наличии терма AB нужно исключить термы ABC, ABD, $ABCD$ и т. п. Операция исключения более сложных термов при наличии более простых называется обычно операцией *элементарного поглощения* термов.

Операция элементарного поглощения соответствует тождеству $A \vee AB = A$ булевой алгебры. Если использовать дизъюнктивное представление импликантной таблицы лишь для нахождения приведенных систем простых импликант, то наряду с этим тождеством можно, разумеется, употреблять для упрощения выражения также и другие тождества булевой алгебры, не содержащие отрицаний и констант. Иначе говоря, можно пользоваться законами ассоциативности и коммутативности для умножения и дизъюнкции, а также тождествами $AA = A$, $A \vee A = A$.

Выполняя в дизъюнктивном представлении импликантной таблицы все элементарные поглощения и устраняя все повторения в соответствии с тождествами $AA = A$ и $A \vee A = A$, мы приходим к так называемому *приведенному дизъюнктивному представлению импликантной таблицы*. Термам этого представления соответствуют все приведенные системы простых импликант рассматриваемой булевой функции.

В процессе получения приведенного дизъюнктивного представления импликантной таблицы из конъюнктивного представления можно применять различные упрощающие преобразования с помощью формул булевой алгебры еще до получения обычного (неприведенного) дизъюнктивного представления. Такая возможность обусловливается тем, что термы приведенного дизъюнктивного представления

можно рассматривать как простые импликанты булевой функции (от переменных $A, B, C, \ldots$), задаваемой исходным конъюнктивным представлением, а само приведенное дизъюнктивное представление — как сокращенную дизъюнктивную нормальную форму этой функции. Обоснование этого факта будет дано в одном из следующих параграфов при рассмотрении того способа минимизации булевых функций, который носит название метода Нельсона. А пока достаточно сослаться лишь на то, что тождественные преобразования любой функции в булевой алгебре не меняют ее сокращенную д. н. ф.

Использование промежуточных упрощений позволяет в большинстве случаев существенно сократить процесс построения приведенного дизъюнктивного представления импликантной таблицы. В возможности таких упрощений как раз и заключается преимущество рассматриваемого алгебраического метода нахождения приведенных систем простых импликант по сравнению с методом перебора всех вариантов накрытий непосредственно по импликантной таблице.

Вернувшись к рассматривавшемуся выше примеру, мы видим, что

$$\varphi = A(A \vee D)\, B(B \vee C)\,(C \vee D) = (A \vee AD)\,(B \vee BC)\,(C \vee D) = \\ = AB(C \vee D) = ABC \vee ABD.$$

Таким образом, в полном соответствии с тем, что было получено ранее, мы находим две приведенные системы простых импликант:

$$(A,\ B,\ C),\quad (A,\ B,\ D).$$

Им соответствуют две тупиковые д. н ф. исходной функции:

$$f_1 = \bar{x}y \vee x\bar{y} \vee xz, \quad f_2 = \bar{x}y \vee x\bar{y} \vee yz.$$

Заметим, что все проведенные нами рассуждения и выводы остаются в силе, если импликантную таблицу функции f строить не для простых импликант, а для произвольных импликант этой функции. В частности, алгебраический метод позволяет находить все приведенные системы таких импликант, определяемые аналогично тому,

как были определены приведенные системы простых импликант (см. определение 1.7).

Отметим один частный случай, при котором система всех простых импликант булевой функции является приведенной системой. Это — случай, когда все простые импликанты не содержат отрицаний переменных. Действительно, в этом случае все простые импликанты, отличные от любой данной импликанты p, должны обязательно содержать буквы, не входящие в импликанту p. Пусть $x_1, x_2, \ldots, x_k$ — все буквы (аргументы рассматриваемой функции), не входящие в импликанту p. Импликанта p накрывает конституенту единицы $K = p\bar{x}_1\bar{x}_2\ldots\bar{x}_k$. Любая же другая простая импликанта содержит хотя бы одну из букв $x_1, \ldots, x_k$ без отрицания и не может, следовательно, накрывать эту конституенту. Таким образом, система всех простых импликант оказывается приведенной системой.

Аналогичная ситуация будет иметь, очевидно, место и в том случае, когда все переменные входят в простые импликанты непременно с отрицаниями. Более общо: будем говорить, что в системе простых импликант все переменные *разделены*, если любая переменная x_i не может входить в одну из импликант системы с отрицанием, а в другую — без отрицания. Как и выше, мы легко установим, что полная система простых импликант с разделенными переменными является непременно приведенной системой. Иначе говоря, справедливо следующее предложение.

1.11. *Если сокращенная дизъюнктивная нормальная форма булевой функции не содержит никакой буквы, входящей в нее одновременно с отрицанием и без отрицания, то эта форма является минимальной дизъюнктивной нормальной формой.*

Заметим, что монотонная булева функция f не может иметь ни одной простой импликанты, содержащей отрицания переменных. Действительно, предположим противное: пусть p — простая импликанта функции f, содержащая отрицания переменных. Обозначим через q произведение всех переменных, входящих в p без отрицаний, а через α — набор, отличающийся тем, что в нем всем переменным, входящим в q, приписано значение 1, а всем остальным переменным — значение 0. Импликанта p, а значит, и

функция f обращаются на этом наборе в единицу. С другой стороны, произведение q обращается в единицу только на наборах $\beta \geqslant \alpha$ (в том числе, и на наборе α). Но на всех таких наборах функция β, в силу определения свойства монотонности, также равна 1. Следовательно, q — импликанта функции f, а импликанта p — не простая, что противоречит сделанному предположению. Таким образом, простые импликанты монотонной функции не содержат отрицаний переменных. Объединяя этот результат с предложением 1.11, мы приходим к следующему предложению [1]).

1.12. *Сокращенная дизъюнктивная нормальная форма монотонной булевой функции не содержит отрицаний переменных и является минимальной дизъюнктивной нормальной формой этой функции.*

В дополнение к теореме 1.12 отметим, что возможность представления булевой функции дизъюнктивной нормальной формой; не содержащей отрицаний переменных, означает монотонность этой функции, поскольку, как было доказано в § 7 предыдущей главы, суперпозиция монотонных функций является всегда монотонной функцией. Таким образом, *булева функция тогда и только тогда монотонна, когда ее сокращенная д. н. ф. не содержит отрицаний переменных.*

В заключение настоящего параграфа разберем вопрос о *простых импликантах* для случая *частичных* булевых функций, то есть таких булевых функций, значения которых определены не на всех наборах.

Простой импликантой частичной булевой функции f мы будем называть всякую простую импликанту p функции f^*, получающейся из функции f в результате доопределения этой последней значениями, равными единице, на всех наборах, на которых ранее функция f не была определена. Простую импликанту функции f мы будем называть *существенной*, если она накрывает хотя бы одну единицу этой функции.

Система простых импликант функции f называется *полной*, если любая единица функции f накрывается хотя

[1]) См. С. В. Яблонский, Функциональные построения в k-значной логике, Тр. Матем. ин-та им В. А. Стеклова, т. 51 1958, стр. 29.

бы одной импликантой этой системы. Полная система простых импликант называется *приведенной* системой, если из нее нельзя исключить ни одну импликанту без потери свойства полноты.

Условимся дизъюнкцию всех простых импликант, принадлежащих какой-либо приведенной системе простых импликант частичной булевой функции f, называть *тупиковой дизъюнктивной нормальной формой* функции f, а дизъюнкцию всех существенных простых импликант функции f — ее *сокращенной дизъюнктивной нормальной формой*.

Во всех точках, в которых значения частичной булевой функции f определены, эти значения совпадают, очевидно, со значениями ее сокращенной д. н. ф. и со значениями любой из ее тупиковых д. н. ф. Минимальная д. н. ф. частичной булевой функции f определяется как д. н. ф., имеющая наименьшее число букв среди всех д. н. ф., значения которых совпадают со значениями функции f в области ее определения. Подобно тому как это было сделано выше для всюду определенных булевых функций, легко показать, что все минимальные д. н. ф. частичной булевой функции находятся среди ее тупиковых д. н. ф.

§ 2. Метод Квайна — Мак-Класки

В настоящем параграфе мы изложим метод нахождения простых импликант булевой функции, предложенный Квайном[1]) и усовершенствованный Мак-Класки[2]). В методе Квайна простые импликанты находятся по совершенной д. н. ф. булевой функции в результате последовательного применения к ней операции, которую мы назовем *операцией неполного склеивания*, и операции *элементарного поглощения*.

Для того чтобы уяснить себе суть этих операций, рассмотрим процесс *развертывания* сокращенной д. н. ф. в совершенную д. н. ф. Пусть, например, мы имеем

---

[1]) W. V. Quine, The problem of simplifying of truth functions, Amer. Math. Monthly, v. 59, № 8, 1952, p. 521—531.

[2]) Mc. Cluskey, Minimizations of boolean functions. Bell System Techn. J., v. 35, № 6, 1956, p. 1417—1444.

сокращенную д. н. ф.

$$f_1 = xy \vee yz \vee z\bar{u}.$$

Для преобразования ее в совершенную д. н. ф. мы к каждому элементарному произведению добавляем множители, представляющие собой, каждый, дизъюнкцию буквы, отсутствующей в произведении, и отрицания этой буквы; затем мы производим раскрытие скобок. Применительно к форме f_1 этот процесс дает на первом шаге выражение

$$f_2 = xy(z \vee \bar{z}) \vee (x \vee \bar{x})yz \vee (x \vee \bar{x})z\bar{u} = \\ = xyz \vee xy\bar{z} \vee xyz \vee \bar{x}yz \vee xz\bar{u} \vee \bar{x}z\bar{u};$$

на втором шаге получается выражение

$$f_3 = xyz(u \vee \bar{u}) \vee xy\bar{z}(u \vee \bar{u}) \vee xyz(u \vee \bar{u}) \vee \bar{x}yz(u \vee \bar{u}) \vee \\ \vee x(y \vee \bar{y})z\bar{u} \vee \bar{x}(y \vee \bar{y})z\bar{u} = xyzu \vee xyz\bar{u} \vee xy\bar{z}u \vee xy\bar{z}\bar{u} \vee \\ \vee xyzu \vee xyz\bar{u} \vee \bar{x}yzu \vee \bar{x}yz\bar{u} \vee xyzu \vee x\bar{y}zu \vee \bar{x}yzu \vee \bar{x}\bar{y}z\bar{u}.$$

Последнее из полученных выражений представляет собой искомую совершенную д. н. ф., у которой некоторые из конституент повторены несколько раз (например, конституента $xyzu$ входит в выражение f_3 трижды).

Легко видеть, что процесс перехода от формы f_1 к форме f_3 обратим: можно от формы f_3 перейти к форме f_2, а затем и к форме f_1 с помощью операции *склеивания*, осуществляемой на основе формулы $Ax \vee A\bar{x} = A$. Условимся (применительно к этой формуле) говорить, что члены Ax и $A\bar{x}$ склеиваются по x, давая в результате член A.

В результате склеивания по u первые два члена выражения f_3 дадут член xyz, представляющий собой первый член выражения f_2. Склеивая по u третий и четвертый члены из f_3, мы получим второй член $xy\bar{z}$ выражения f_2, и т. д., пока не получим все члены выражения f_2. Применяя подобный же прием к выражению f_2, мы придем к выражению f_1, представляющему собой дизъюнкцию всех простых импликант функции f_3.

Ясно, что приведенные рассуждения имеют вполне общее значение; именно, с помощью многократного применения операции склеивания мы из совершенной д. н. ф.,

отличающейся тем, что в ней некоторые конституенты выписаны более одного раза, можем получить сокращенную д. н. ф. Недостатком этого метода является то, что мы не знаем заранее, какие именно конституенты и в каком числе необходимо вписать в исходную совершенную д. н. ф. Нетрудно, однако, внести такое изменение в метод, которое делает многократное выписывание конституент излишним. С этой целью достаточно, очевидно, при применении операции склеивания не вычеркивать склеиваемые члены. Иначе говоря, вместо формулы

$$Ax \vee A\bar{x} = A$$

следует пользоваться формулой

$$Ax \vee A\bar{x} = A \vee Ax \vee A\bar{x}.$$

Тождественное преобразование, выражаемое последней формулой, мы будем называть операцией *неполного склеивания*.

Операция неполного склеивания отличается тем, что члены, уже подвергавшиеся склеиванию, могут принимать участие в дальнейших склеиваниях столько раз, сколько это окажется необходимым. В силу этого исключается необходимость предварительного многократного выписывания некоторых членов совершенной д. н. ф. Для того чтобы прием, основанный на применении операции неполного склеивания, давал точно такой же результат, как и прием, описанный ранее, следует позаботиться еще об удалении членов, подвергшихся склеиванию, после того как с ними произведены в с е возможные склеивания. Операция удаления склеенных членов носит название операции *элементарного поглощения*. Смысл этой операции состоит в последовательном применении формулы элементарного поглощения $Au \vee A = A$.

Необходимо подчеркнуть при этом, что исключить с помощью операции элементарного поглощения можно лишь те члены, к которым были применены в с е возможные для них склеивания. Нарушение этого условия может привести к ошибкам. С целью исключить возможность подобных ошибок, операции неполного склеивания и элементарного поглощения необходимо производить

в определенном порядке, указываемом следующим *алгоритмом Квайна*:

1. *Минимизируемая булева функция f от произвольного числа n переменных записывается в совершенной д. н. ф. f_0.*

2. *Отправляясь от д. н. ф. f_0, строим последовательность д. н. ф.: $f_0, f_1, \ldots, f_l, \ldots$ до тех пор, пока две какие-либо д. н. ф. f_k и f_{k+1} не совпадут между собой. Переход от формы f_i к форме f_{i+1} $(i = 1, \ldots, k)$ производится по следующему правилу: в форме f_i выполняются все операции неполного склеивания, применимые к элементарным произведениям длины $n - i$, после чего исключаются все те элементарные произведения длины $n - i$, которые могут быть исключены на основании формулы элементарного поглощения.*

3. *Результатом применения алгоритма Квайна к функции f является заключительная д. н. ф. f_k.*

Легко видеть, что, применяя алгоритм Квайна, мы придем к определенному результату (д. н. ф. f_k) не более чем за n шагов. Рассуждения, проведенные выше, доказывают справедливость следующей т е о р е м ы К в а й н а.

2.1. *Для любой булевой функции f результатом применения алгоритма Квайна к совершенной д. н. ф. этой функции будет сокращенная д. н. ф. функции f.*

Иными словами, с помощью последовательного (многократного) применения операций неполного склеивания и элементарного поглощения в конечное число шагов совершенная д. н. ф. любой булевой функции преобразуется в сокращенную д. н. ф. этой функции.

Продемонстрируем работу алгоритма Квайна на примере булевой функции f, заданной своей совершенной д. н. ф.:

$$xyz \vee \bar{x}yz \vee x\bar{y}z \vee xy\bar{z} \vee \bar{x}\bar{y}\bar{z}.$$

Операции неполного склеивания можно применить к первому и второму члену формы f_0, к первому и третьему, а также к первому и четвертому ее членам. В результате мы приходим к форме:

$$f_0' = yz \vee xz \vee xy \vee yz \vee \bar{x}yz \vee x\bar{y}z \vee xyz \vee \bar{x}\bar{y}\bar{z}.$$

После четырехкратного применения операции элементарного поглощения форма f_0' преобразуется в форму

$$f_1 = yz \vee xz \vee xy \vee \bar{x}\bar{y}\bar{z}.$$

Поскольку операция неполного склеивания далее неприменима, форма f_2 совпадает с формой f_1. В силу алгоритма Квайна и теоремы 2.1 мы заключаем, что f_1 — сокращенная д. н. ф. функции f. Таким образом, функция f имеет четыре простые импликанты.

Для булевых функций от большого числа переменных применение алгоритма Квайна в описанном выше виде становится затруднительным. Затруднения возникают в связи с громоздкостью записи конституент и возможностью ошибок при отыскании склеивающихся членов в формах, содержащих большое количество членов. С целью обойти указанные затруднения, Мак-Класки [1]), а также Абрахам и Нордаль [2]) предложили несколько иное, практически более удобное, оформление алгоритма Квайна. Алгоритм Квайна в этом оформлении мы будем называть (усовершенствованным) *алгоритмом Мак-Класки.*

В отличие от алгоритма Квайна, усовершенствованный алгоритм Мак-Класки основывается на задании конституент единицы не в их натуральном виде, а в виде условных чисел, называемых *номерами соответствующих конституент.* Чтобы получить номер конституенты, нужно набор, на котором она обращается в единицу, рассматривать как запись этого номера в двоичной системе счисления. При этом конституента $\bar{x}_1\bar{x}_2\ldots\bar{x}_n$ получает номер 0, а конституента $x_1x_2\ldots x_n$ — номер $2^n - 1$. Для случая трех переменных конституенте $\bar{x}\bar{y}\bar{z}$ приписывается номер 0, конституенте $x\bar{y}\bar{z}$ — номер 4, конституенте $xy\bar{z}$ — номер 6 и т. д.

Индексом целого числа мы условимся называть число единиц в двоичном представлении этого числа. Например,

---

[1]) Mc. Cluskey, Minimizations of boolean functions, Bell. System Techn. J., v. 35, № 6, 1956, p. 1417—1444.

[2]) См. S. H. Caldwell, Switching circuits and logical design. John. Willey and Son inc., 1958.

число 7 = 111 имеет индекс 3, число 12 = 1100 — индекс 2 и т. д. Ясно, что индекс 1 имеют все числа, являющиеся степенью двойки, и только такие числа.

Нетрудно усмотреть справедливость следующего предложения.

2.2. *Для того чтобы два числа m и n являлись номерами двух склеивающихся между собой конституент единицы, необходимо и достаточно, чтобы их индексы различались точно на единицу, чтобы сами числа отличались друг от друга на степень двойки и чтобы число с большим индексом было больше числа с меньшим индексом.*

Действительно, необходимость условий, приведенных в теореме 2.2, очевидна, поскольку двоичные представления номеров двух склеивающихся конституент имеют вид $m = \alpha 0 \beta$ и $n = \alpha 1 \beta$ (α и β — какие-то наборы нулей и единиц). В целях доказательства достаточности указанных условий заметим, что прибавление к любому целому (неотрицательному) числу r числа, являющегося степенью двойки $2^k (k = 0, 1, \ldots)$, только тогда увеличивает индекс этого числа точно на единицу, когда в $(k + 1)$-м разряде двоичного представления числа r стоит нуль. В самом деле, если бы в $(k + 1)$-м разряде стояла единица, то после прибавления числа 2^k в этом разряде появился бы нуль, что вызвало бы уменьшение индекса числа на единицу. Возникающий при этом перенос в старшие разряды осуществляет преобразование некоторого числа $p \geqslant 0$ единиц старших разрядов в нули и лишь одного нуля — в единицу.

Таким образом, в результате сложения индекс числа уменьшился бы на p единиц вместо того, чтобы увеличиться на единицу. Но из условий, сформулированных в теореме, вытекает, что большее из чисел m и n (скажем, для определенности, число m) получается из меньшего числа (n) прибавлением к нему числа, являющегося некоторой степенью двойки 2^k, т. е. $m = n + 2^k$. При этом индекс числа m точно на единицу больше индекса числа n. Как было только что показано, это возможно лишь в том случае, когда двоичное представление числа n имеет нуль в $(k+1)$-м разряде, и, значит, двоичные представления чисел m и n имеют вид $m = \alpha 1 \beta$, $n = \alpha 0 \beta$. Очевидно, что конституенты, соответствующие этим

номерам, различаются отрицанием только у одной из составляющих их букв и, следовательно, могут быть склеены. Тем самым предложение 2.2 полностью доказано.

Элементарное произведение, получившееся в результате склеивания двух конституент с номерами m и n, обозначается в алгоритме Мак-Класки упорядоченной парой (n, m) этих номеров (первым выписывается меньший номер). Член, получающийся при склеивании двух таких произведений (p, q) и (r, s), обозначается четверкой номеров (p, q, r, s), расположенных в порядке их возрастания. Произведения еще меньшей длины будут обозначаться восьмеркой чисел и т. д. При таком способе обозначения склеиванию двух элементарных произведений соответствует объединение обозначающих их множеств номеров (с условием последующей записи этих номеров в порядке возрастания).

Нетрудно показать справедливость следующего предложения.

2.3. *Элементарное произведение, обозначенное множеством номеров P, входит в качестве составной части во все конституенты единицы, номера которых входят в множество P. Элементарное произведение, обозначенное множеством номеров R, тогда и только тогда поглощает элементарное произведение, обозначенное множеством номеров S, когда множество S является подмножеством множества R: $S \subset R$.*

Остается еще научиться распознавать те элементарные произведения (заданные множествами номеров), которые можно склеивать между собой. С этой целью каждому элементарному произведению, кроме множества номеров конституент, сопоставляется еще набор чисел, называемых *разностями*.

Если x_1, x_2, ..., x_n — все аргументы минимизируемой булевой функции, с которыми имеют дело, а рассматриваемое произведение не содержит (ни в прямом, ни в инверсном виде) переменных x_{i_1}, x_{i_2}, ..., x_{i_k}, но содержит все остальные переменные (или их отрицания), то ему сопоставляется набор разностей 2^{n-i_1}, 2^{n-i_2}, ..., 2^{n-i_k}. В частности, любой конституенте единицы сопоставляется пустое множество разностей.

Справедливо следующее предложение.

2.4. *Два элементарных произведения p и q тогда и только тогда склеиваются между собой, когда соответствующие им наборы разностей одинаковы, а наименьшие номера m и n в обозначающих их множествах P и Q являются номерами склеивающихся между собой конституент единицы. В результате склеивания получается элементарное произведение r, обозначаемое объединением множеств P и Q. Набор разностей элементарного произведения r получается из набора разностей любого из произведений p или q добавлением к нему модуля разности номеров m и n.*

Действительно, конституенты единицы с номерами m и n получаются, в силу теоремы 2.3, из произведений p и q в результате умножения их на элементарные произведения s_1 и s_2, состоящие из отрицаний букв, не вошедших, соответственно, в p и q. Теперь ясно, что произведения p и q склеиваются между собой (то есть имеют вид lx_l и $l\bar{x}_l$) тогда и только тогда, когда $s_1 = s_2$ и когда конституенты ps_1 и qs_2 склеиваются между собой. Если произведения p и q склеиваются между собой по переменной x_l, то номера m и n будут иметь (в двоичной записи) вид $\alpha_1\alpha_2\ldots\alpha_{l-1}\ 0\alpha_{l+1}\ \ldots\alpha_n$ и $\alpha_1\alpha_2\ldots\ \alpha_{l-1}\ 1\alpha_{l+1}\ldots\alpha_n$. Модуль их разности равняется 2^{n-l}. Но в силу определения понятия набора разностей именно эту разность надо добавить к элементарному произведению l, получающемуся в результате склеивания элементарных произведений p и q, поскольку в произведении l содержатся все те переменные, которые содержатся в произведениях p и q, исключая переменную x_l.

Поскольку объединение обозначающих множеств при склеивании элементарных произведений предусмотрено в самом их определении, то предложение 2.4 полностью доказано.

Предложения 2.2, 2.3 и 2.4 дают возможность осуществлять алгоритм Квайна для получения сокращенных д. н. ф. из совершенных д. н. ф. способом, при котором конституенты единицы заданы номерами, а произвольные элементарные произведения — множествами номеров. При этом для удобства проверки конституенты группируются по их индексам так, чтобы проверка на склеивание осуществлялась в применении к соседним группам номеров

(то есть к группам номеров, индексы которых отличаются на единицу). Элементарные произведения, исключающиеся с помощью применения операции поглощения, принято обозначать звездочкой. Звездочкой обозначаются также конституенты, соответствующие наборам, на которых заданная функция не определена. Простыми импликантами булевой функции будут все те элементарные произведения (обозначенные множествами номеров), которые после выполнения всех операций окажутся не отмеченными звездочкой и которые отличаются тем, что обозначающие их множества содержат хотя бы один номер, не отмеченный звездочкой.

Последовательность выкладок, выполняемых в рассмотренном алгоритме (алгоритме Мак-Класки), проще всего уяснить на примерах. Заметим, что при применении алгоритма Мак-Класки булевы функции принято задавать множествами номеров конституент единицы их совершенных д. н. ф. В это множество включаются обычно также номера конституент, соответствующих наборам, на которых функция не определена. Эти номера обозначаются звездочками.

Пример 1. Найти все простые импликанты частичной булевой функции f от четырех переменных, заданной множеством номеров своих конституент единицы:

$$f=(1,\ 3,\ 4,\ 5,\ 9,\ 15,\ 10^*,\ 12^*).$$

Решение. Группируем номера конституент в порядке роста их индексов, отделяя друг от друга группы номеров, имеющих одинаковый индекс, вертикальными чертами:

$$|1,4|3,5,9,10^*,12^*\ ||\ 15|.$$

В первую группу попали номера, имеющие индекс 1: $1 = 0001$, $4 = 0100$, во вторую группу — номера, имеющие индекс 2: $3 = 0011$, $5 = 0101$, $9 = 1001$, $10 = 1010$, $12 = 1100$, и, наконец, в последнюю группу попал один номер, имеющий индекс 4: $15 = 1111$.

В силу теоремы 2.2 склеивание возможно только между номерами первой и второй группы. Поскольку индексы номеров второй и третьей группы различаются на двойку, склеивание номеров этих групп невозможно. Это обсто-

ятельство обозначается тем, что вторая и третья группы отделены друг от друга не одной, а двумя вертикальными чертами.

Для выполнения склеиваний нужно, в соответствии с теоремой 2.2, находить пары номеров такие, чтобы номер из второй группы был на некоторую степень двойки больше номера из первой группы. Таким образом, мы приходим к следующему множеству пар:

$$(1,3),\ (1,5),\ (1,9),\ (4,5),\ (4,12\ *).$$

Разности, соответствующие этим парам, принято выписывать рядом с ними:

$$(1,3)(2),\ (1,5)(4),\ (1,9)(8),\ (4,5)(1),\ (4,12\ *)(8).$$

Элементарные поглощения приведут к тому, что исходное множество конституент представится в виде

$$|1^*,\ 4^*\ |3^*,\ 5^*,\ 9^*,\ 10^*,\ 12^*\|15|.$$

Поскольку дальнейшие склеивания полученных пар оказываются невозможными, алгоритм заканчивает свою работу. В результате его выполнения мы получили 6 простых импликант исходной булевой функции f:

$$f_1=(15),\quad f_2=(1,\ 3),\quad f_3=(1,\ 5),\quad f_4=(1,\ 9),$$
$$f_5=(4,\ 5)\quad f_6=(4,\ 12).$$

Предположим, что переменными, от которых зависит функция f, будут переменные x, y, z, u, так что первая импликанта f_1 (конституента с номером 15) будет иметь вид: $f_1 = xyzu$. Для нахождения в явном виде остальных импликант воспользуемся следующим очевидным приемом: 1) выписываем для каждой простой импликанты P какую-нибудь конституенту единицы K, номер которой входит в обозначающее множество импликанты P; 2) по разностям, соответствующим импликанте P, определяем переменные, которые отсутствуют в представляющем ее элементарном произведении, и вычеркиваем эти переменные из конституенты K.

Применяя этот прием к нашему случаю, мы получим, что импликанты f_2, f_3 и f_4 могут быть получены из конституенты единицы $xyzu$ с номером 1 в результате вычер-

кивания из нее букв z, y и x соответственно, а импликанты f_5 и f_6 получаются из конституенты единицы $\bar{x}\bar{y}\bar{z}\bar{u}$ с номером 4 в результате вычеркивания из нее букв u и x соответственно. Выполняя указанные вычеркивания, мы получим:

$$f_2=\bar{x}\bar{y}u,\quad f_3=\bar{x}\bar{z}u,\quad f_4=\bar{y}\bar{z}u,\quad f_5=\bar{x}y\bar{z},\quad f_6=y\bar{z}\bar{u},$$

что и дает искомое решение (вместе с выписанной ранее простой импликантой $f_1=xyzu$).

Пример 2. Найти минимальную д. н. ф. частичной булевой функции:

$$f(x\ y,\ z,\ u)=(0,\ 1,\ 2,\ 3,\ 4,\ 6,\ 8,\ 9,\ 11,\ 14,\ 7^*,\ 10^*).$$

Решение. Расположим выкладки так, как это обычно принято при применении алгоритма Мак-Класки, помещая все импликанты (элементарные произведения)

Таблица IV.2

Индекс				
0	0 *	(0,1) (1) *		
		(0,2) (2) *		
		(0,4) (4) *	(0, 1, 2, 3) (1) (2) *	
	1 *	(0,8) (8) *	(0, 1, 8, 9) (1) (8) *	
1	2 *			
	4 *		(0, 2, 4, 6) (2) (4)	
	8 *	(1,3) (2) *	(0, 2, 8, 10) (2) (8) *	
		(1,9) (8) *		
		(2,3) (1) *		
	3 *	(2,6) (4) *	(1, 3, 9, 11) (2) (8) *	
		(2,10) (8) *		
	6 *	(4,6) (2) *	(2, 3, 6, 7) (1) (4)	
2		(8,9) (1) *		
	9 *	(8,10) (2) *		
			(2, 6, 10, 14) (4) (8)	
	10 *	(3,11) (8) *		
		(3,7) (4) *		
	11 *	(6,7) (1) *	(2, 3, 10, 11) (1) (8) *	(0,1,2,3,8,9,10,11) (1)(2)(8)
		(6,14)(8)*		
3	7 *	(9,11) (2) *		
		(10,11) (1) *	(8, 9, 10, 11) (1) (2) *	
	14 *	(10,14) (4) *		

функции f, имеющие одну и ту же длину, в свой собственный столбец и разделяя группы номеров, имеющих одинаковые индексы, горизонтальными чертами (табл. IV.2).

В результате получаем четыре простые импликанты, обозначаемые множествами (0,2,4,6), (2,3,6,7), (2,6,10,14) и (0,1,2,3,8,9,10,11). Обозначая эти импликанты соответственно буквами A, B, C и D и применяя прием, описанный при рассмотрении примера 1, мы получим, что $A = \bar{x}\bar{u}$, $B = \bar{x}z$, $C = z\bar{u}$, $D = \bar{y}$.

Импликантная таблица функции f может быть легко найдена с помощью теоремы 2.3: всякая импликанта, в обозначающем множестве которой содержится номер какой-либо конституенты единицы, обращается в единицу на наборе, соответствующем этой конституенте. Для рассматриваемой функции мы получаем импликантную табл. IV.3.

Таблица IV.3

	0	1	2	3	4	6	8	9	11	14
A	*		*		*	*				
B			*	*		*				
C			*			*				*
D	*	*	*	*			*	*	*	

Импликанты A, C и D составляют ядро функции f и накрывают, как легко видеть, все единицы этой функции. Поэтому функция f обладает единственной тупиковой, а следовательно, и единственной минимальной д.н.ф.:

$$A \vee C \vee D = \bar{x}\bar{u} \vee z\bar{u} \vee \bar{y}.$$

Полученная форма и дает искомое решение.

Заметим, что в только что рассмотренном примере при склеивании всех элементарных произведений, за исключением конституент, один и тот же результат получался при нескольких склеиваниях (хотя выписывался, разумеется, только один раз). Нетрудно показать, что это не является случайностью, а представляет собою общее правило.

Заметим, что в ряде случаев оказывается более целесообразным применять не усовершенствованный, а обычный алгоритм Мак-Класки. В этом алгоритме элементарные произведения обозначаются наборами нулей, единиц и черточек. Такая система обозначений основывается на заранее фиксированной нумерации булевых переменных, с которыми имеют дело.

Предположим, что имеется n таких переменных: $x_1, x_2, \ldots, x_n$. Тогда любое элементарное произведение, составленное из этих переменных и их отрицаний, обозначается конечной упорядоченной последовательностью $\alpha_1 \alpha_2 \ldots \alpha_n$ длины n, состоящей из нулей, единиц и черточек. Последовательность эта строится по следующему правилу: *i-й элемент α_i последовательности $\alpha_1\alpha_2 \ldots \alpha_n$, обозначающий любое данное элементарное произведение p, равняется нулю, если произведение p содержит отрицание i-й переменной x_i; он равняется единице, если произведение p содержит переменную x_i без отрицания; он заменяется черточкой, если произведение p вообще не содержит переменной x_i ни в прямом, ни в инверсном виде.*

Например, при $n=4$ элементарное произведение $x_1\bar{x}_3$ получит обозначение 1—0—, элементарное произведение $\bar{x}_1x_2\bar{x}_3\bar{x}_4$ — обозначение 0000, элементарное произведение x_1 — обозначение 1— — —, и т. д.

Используя эти обозначения, можно осуществлять минимизацию булевых функций с помощью тех же приемов, которые применяются в описанном выше (усовершенствованном) алгоритме. При этом отпадает, разумеется, необходимость в выписывании разностей. Ясно, что при принятой системе обозначений наборы разностей окажутся одинаковыми у тех и только тех элементарных произведений, которые имеют черточки на одних и тех же местах. Расположение же конституент в порядке их индексов сохраняется и в рассматриваемом случае.

Работу алгоритма Мак-Класки лучше всего уяснить на примере.

Пример 3. Найти минимальную д.н.ф. булевой функции:

$$f = \bar{x}\bar{y}\bar{z}u \vee \bar{x}y\bar{z}u \vee \bar{x}\bar{y}zu \vee \bar{x}y\bar{z}u \vee x\bar{y}zu \vee \bar{x}yzu \vee xyzu.$$

Таблица IV 4

Индекс			
1	0001 *	0—01 * 00—1 *	
2	0101 * 0011 * 1001 *	—001 * —011 * 10—1 *	0——1 —0—1
3	1011 * 0111 *	01—1 * 0—11 *	——11
4	1111 *	1—11 * —111 *	

Таблица IV.5

Индекс	Числа
1	1, 2, 4, 8, 16, 32, 64
2	3, 5, 6, 9, 10, 12, 17, 18, 20, 24, 33, 34, 36, 40, 48, 65, 66, 68 72, 80, 96
3	7, 11, 13, 14, 19, 21, 22, 25, 26, 28, 35, 37, 38 41, 42, 44, 49, 50, 52, 56, 67, 69, 70 73 74, 76, 81, 82, 84, 88, 97, 98, 100, 104, 112
4	15 23, 27, 29, 30, 39, 43, 45, 46, 51, 53, 54, 57, 58, 60, 71, 75, 77, 78, 83, 85, 86, 89, 90, 92, 99, 101, 102, 105, 106, 108, 113, 114 116, 120
5	31, 47, 55, 59, 61, 62, 79, 87, 91, 93, 94, 103, 107 109, 110, 115 117, 118 121, 122, 124
6	63, 95, 111, 119, 123, 125 126
7	127

Р е ш е н и е. Располагая переменные в следующем порядке: x, y, z, u, мы получим представление конституент функции f в виде: 0001, 0101, 0011, 1001, 1011, 0111, 1111. Применим обычную схему расположения выкладок (см. табл. IV.4)

В результате мы получаем три простые импликанты: $A = \bar{x}u$ (обозначение 0——1), $B = \bar{y}u$ (обозначение —0—1) и $C = zu$ (обозначение ——11). Легко проверить, что все эти импликанты входят в ядро рассматриваемой функции. Таким образом, искомая минимальная д.н.ф. имеет вид: $f = \bar{x}u \vee \bar{y}u \vee zu$.

В заключение параграфа приведем таблицу распределения чисел по заданным индексам для всех чисел, меньших чем 128 (это достаточно для минимизации булевых функций, зависящих не более чем от 7 переменных). Понятно, что единственным числом, имеющим индекс 0, будет само число 0. Для всех же остальных индексов соответствующие им числа представлены в таблице IV.5.

§ 3. Другие методы минимизации булевых функций

Разобранный в предыдущем параграфе м е т о д К в а й н а — М а к-К л а с к и минимизации булевых функций в качестве своей отправной точки имеет совершенную д.н.ф. Однако в ряде случаев приходится осуществлять минимизацию (нахождение простых импликант) булевых функций, заданных произвольными д.н.ф. В этом случае целесообразно использовать метод минимизации, предложенный Б л е й к о м [1]).

Метод Блейка основывается на использовании операции, которую мы будем называть *операцией обобщенного склеивания*. Суть этой операции состоит в применении следующего тождественного соотношения булевой алгебры:

$$AC \vee B\bar{C} = AC \vee B\bar{C} \vee AB. \qquad (1)$$

---

[1]) A. B l a k e, Canonical expression in Boolean Algebra. Dissertation, Chicago, 1937. Метод Блейка восходит к работам русского логика 19 в. П. С. П о р е ц к о г о, в результаты которого Блейк внес усовершенствования.

Для того чтобы убедиться в справедливости этого соотношения достаточно провести следующие тождественные преобразования:

$$AC \vee B\bar{C} = AC \vee ABC \vee B\bar{C} \vee AB\bar{C} =$$
$$= AC \vee B\bar{C} \vee AB(C \vee \bar{C}) = AC \vee B\bar{C} \vee AB.$$

Метод Блейка состоит в том, что в произвольной д.н.ф. заданной булевой функции f осуществляются все допустимые обобщенные склеивания. После устранения элементарных поглощений (то есть удаления дизъюнктивных членов вида AB при наличии дизъюнктивных членов A или B) этот метод приводит к сокращенной д.н.ф.

Сформулируем и докажем соответствующее предложение.

3.1. *Если в произвольной д.н.ф. булевой функции f произвести все возможные обобщенные склеивания и устранить затем все элементарные поглощения, то в результате получится сокращенная д.н.ф. функции f.*

Для доказательства сформулированного предложения достаточно доказать, что в результате многократного применения операции обобщенного склеивания из произвольной д.н.ф. булевой функции может быть получена любая простая импликанта этой функции. В самом деле, в результате применения к д.н.ф. операции обобщенного склеивания мы снова получаем некоторую д. н. ф. Каждый же член д.н.ф. является элементарным произведением и импликантой функции f. Поэтому он поглощается какой-либо простой импликантой функции f. Таким образом, после получения всех простых импликант устранение всех элементарных поглощений обязательно приведет нас к сокращенной д.н.ф.

Для доказательства того, что в результате обобщенных склеиваний из произвольной д.н.ф. булевой функции f могут быть получены в с е ее простые импликанты, проведем индукцию по числу переменных n, от которых зависит функция f. Для $n = 1$ это утверждение очевидно. Предположим теперь, что оно справедливо для всех $n < m$, и докажем его для $n = m$.

Заметим прежде всего, что простая импликанта p, являющаяся конституентой единицы, входит во всякую

д.н.ф. F и, следовательно, получается из нее в результате пустого множества обобщенных склеиваний (напомним, что каждое новое обобщенное склеивание не уничтожает полученных ранее членов). Действительно, в д.н.ф. F обязательно найдется элементарное произведение r, которое обращается в единицу на наборе, соответствующем конституенте p. Но тогда, очевидно, $p = rl$, где l — некоторое элементарное произведение, и, поскольку r является импликантой функции f, а p — ее простой импликантой, то $l = 1$ и, следовательно, $p = r$.

Итак, когда простая импликанта p является конституентой единицы, она обязательно войдет в любую д.н.ф. функции f, в том числе и в ту д.н.ф., которая получается в результате применения операций обобщенного склеивания к любой данной исходной д.н.ф. F функции f. Предположим теперь, что p не является конституентой единицы; тогда в p не входит хотя бы одна из переменных, от которых зависит функция f; этой переменной может быть, например, переменная x. В этом случае представим д.н.ф. F в виде

$$F = Ax \vee B\bar{x} \vee C,$$

группируя члены, содержащие x и $\bar{x}$, и вынося x и $\bar{x}$ за скобки. Являясь импликантой функции F и не завися от x, элементарное произведение p будет, очевидно, являться импликантой функций, которые получаются из функции F в результате приравнивания x нулю и единице. Иными словами, p является импликантой функций $A \vee C$ и $B \vee C$. Но тогда p будет, очевидно, импликантой и произведения этих функций

$$(A \vee C)(B \vee C) = AB \vee C.$$

Обозначим это произведение через F_1. Отметим, что F_1 является импликантой функции F, поскольку, применяя к F операцию обобщенного склеивания, мы получим, что

$$F = Ax \vee B\bar{x} \vee C = Ax \vee B\bar{x} \vee AB \vee C = Ax \vee B\bar{x} \vee F_1.$$

Теперь ясно, что никакая собственная часть элементарного произведения p не может быть импликантой функции F_1, потому что в противном случае она была бы импликантой функции F, что исключено ввиду простоты импликанты p.

Следовательно, p представляет собой простую импликанту функции F_1. Поскольку функция F_1 зависит от меньшего числа переменных, чем функция F, к ней можно применить индуктивное предположение. Поэтому можно считать, что простая импликанта p получается из любой д.н.ф. функции F_1 в результате некоторого числа обобщенных склеиваний. Но формула

$$Ax \vee B\bar{x} \vee C = Ax \vee B\bar{x} \vee F_1$$

показывает, что какая-то д.н.ф. функции F_1 получается в результате обобщенных склеиваний (по x) из исходной д.н.ф. F. Следовательно, простая импликанта p возникает из д.н.ф. F функции f в результате применения операции обобщенного склеивания, повторенной некоторое число раз. Ввиду произвольности выбора p и F, теорема доказана.

Покажем работу алгоритма Блейка на примере булевой функции, заданной д.н.ф.

$$f = x\bar{y} \vee \bar{x}yz \vee yz.$$

Легко видеть, что первый и второй члены формулы f допускают обобщенные склеивания как по x, так и по y, но возникающие в результате склеивания произведения равны нулю, так как они обязательно содержат одну из букв (x или y) вместе с ее отрицанием. Нетривиальное обобщенное склеивание возможно для первого и третьего члена формы f. Выполняя это склеивание, мы приходим к д.н.ф.

$$f_1 = x\bar{y} \vee \bar{x}yz \vee yz \vee xz.$$

В этой форме нетривиальное обобщенное склеивание можно применить к первому и третьему, а также к второму и четвертому члену. Однако оба эти склеивания приводят к членам, уже имеющимся в форме f_1. Таким образом, можно считать, что в форме f_1 выполнены все возможные в ней обобщенные склеивания.

Производя затем исключение элементарного поглощения (член $\bar{x}yz$ поглощается членом yz), мы придем к сокращенной д.н.ф.:

$$f_2 = x\bar{y} \vee yz \vee xz.$$

Приведем еще один пример, рассматривая который, мы будем производить все необходимые преобразования без каких-либо дополнительных пояснений:

$$f = xy\bar{z} \vee xz \vee \bar{x}y = xy\bar{z} \vee xz \vee \bar{x}y \vee xy \vee y\bar{z} \vee yz =$$
$$= xz \vee \bar{x}y \vee xy \vee y\bar{z} \vee yz \vee y = xz \vee y.$$

Таким образом, сокращенная д.н.ф. функции f имеет вид $xz \vee y$.

Заметим, что в этом примере мы произвели часть элементарных поглощений еще до того, как были закончены все операции обобщенного склеивания. Ясно, что таким, упрощающим выкладки приемом можно пользоваться всякий раз, когда исключаемые члены не могут дать (в результате обобщенного склеивания) никаких новых членов.

Из теоремы 3.1 нетрудно получить другое доказательство теоремы 1.11 из § 1. Действительно, к д.н.ф., не содержащей никакой буквы, которая входила бы в нее одновременно с отрицанием и без отрицания, неприменима, как это очевидно, операция обобщенного склеивания. То же самое относится к дизъюнкции любой части членов этой формы. Если исходная д.н.ф. была сокращенной, то она должна была бы восстанавливаться с помощью операции обобщенного склеивания по минимальной д.н.ф., составляющей некоторую ее часть. В силу сказанного выше, это возможно лишь в том случае, когда минимальная д.н.ф. совпадает с сокращенной д.н.ф.

Используя операцию обобщенного склеивания, в ряде случаев удается относительно просто устанавливать возможность выбрасывания из сокращенной д.н.ф. (или из некоторой ее части, представляющей заданную функцию) некоторых простых импликант. Соответствующую методику можно уяснить себе из рассмотрения следующего примера.

Пусть $f = x\bar{y} \vee \bar{x}z \vee \bar{y}z$. Легко видеть, что третий член формы f может быть получен в результате применения операции обобщенного склеивания к первым двум членам:

$$x\bar{y} \vee \bar{x}z = x\bar{y} \vee \bar{x}z \vee \bar{y}z.$$

Полученное соотношение дает возможность упростить исходное представление булевой функции f, заменив его представлением:

$$f = \bar{x}\bar{y} \vee \bar{x}z.$$

Разберем еще один метод минимизации булевых функций, предложенный Нельсоном [1]). Этот метод основывается на следующей теореме.

3.2. *Если в произвольной конъюнктивной нормальной форме булевой функции раскрыть все скобки в соответствии с дистрибутивным законом и устранить все элементарные поглощения, то в результате получится сокращенная д.н.ф. этой функции.*

Как и в случае предыдущей теоремы, для доказательства теоремы 3.2 достаточно установить, что при раскрытии скобок в произвольной конъюнктивной нормальной форме $F = K_1K_2 \ldots K_m$ булевой функции f может быть получена любая наперед заданная простая импликанта p этой функции. В силу принятого нами определения конъюнктивных нормальных форм (см. гл. III, § 5), можно считать, что функция f не равна тождественно единице, поскольку в противном случае ее конъюнктивная нормальная форма тривиальна (т. е. равна единице). Можно исключить также случай, когда функция f тождественно равна нулю, поскольку в этом случае она не имеет ни одной простой импликанты. Предполагая, таким образом, что функция f не является константой, мы можем считать простую импликанту p этой функции произведением непустого множества букв или их отрицаний:

$$p = \tilde{x}_{i_1}\tilde{x}_{i_2} \ldots \tilde{x}_{i_k} \quad (k \geqslant 1).$$

В силу простоты импликанты p, полагая в функции f:

$$\tilde{x}_{i_2} = 1, \ldots, \tilde{x}_{i_k} = 1,$$

мы не можем обратить эту функцию в тождественную единицу, поскольку иначе произведение $\tilde{x}_{i_2}\tilde{x}_{i_3} \ldots \tilde{x}_{i_k}$ также было бы импликантой функции f. Это означает, очевидно,

---

[1]) R. J. Nelson, Simplest normal truth functions. J. Symb. Logic., v. 20, № 2, 1955, p. 105—108.

что в форме F найдется такая элементарная дизъюнкция, которая не содержит букв $\tilde{x}_{i_2}, \tilde{x}_{i_3}, \ldots, \tilde{x}_{i_k}$. Не нарушая общности, можно считать, очевидно, что такого рода дизъюнкцией является дизъюнкция K_1. Вместе с тем, полагая $\tilde{x}_{i_1} = \tilde{x}_{i_2} = \ldots = \tilde{x}_{i_k} = 1$, мы обратим, очевидно, функцию f в тождественную единицу (для этого достаточно рассмотреть сокращенную д.н.ф. функции f, имеющую вид $p \vee A$). Это означает, в свою очередь, что все элементарные дизъюнкции $K_1, K_2, \ldots, K_m$ обратятся при такой подстановке в единицу. Отсюда следует, что любая элементарная дизъюнкция K_i $(i = 1, \ldots, m)$ содержит хотя бы один из первичных термов $\tilde{x}_{i_1}, \tilde{x}_{i_2}, \ldots, \tilde{x}_{i_k}$. Поскольку дизъюнкция K_1 не содержит первичных термов $\tilde{x}_{i_2}, \ldots, \tilde{x}_{i_k}$, она обязательно содержит терм $\tilde{x}_{i_1}$. Иными словами: $K_1 = \tilde{x}_{i_1} \vee K_1'$. Аналогичным образом можно установить, что $K_2 = \tilde{x}_{i_2} \vee K_2'$, $K_3 = \tilde{x}_{i_3} \vee K_3'$, $\ldots$, $K_k = \tilde{x}_{i_k} \vee K_1'$. В каждую из остальных элементарных дизъюнкций, согласно сказанному выше, обязательно входит хотя бы один из термов $\tilde{x}_{i_1}, \ldots, \tilde{x}_{i_k}$. После этого становится непосредственно очевидным, что при раскрытии скобок в произведении $K_1 K_2 \ldots K_m$ в числе прочих членов будет получен и член $p = \tilde{x}_{i_1} \tilde{x}_{i_2} \ldots \tilde{x}_{i_k}$. Тем самым теорема 3.2 полностью доказана.

В качестве примера применения метода Нельсона рассмотрим булеву функцию f, заданную в виде конъюнктивной нормальной формы:

$$f = (x \vee \bar{y})\,(\bar{x} \vee z)\,(x \vee y \vee \bar{z}).$$

Раскрывая скобки в этой форме, получим:

$$f = xz \vee xzy \vee \bar{x}\,\bar{y}\,\bar{z} \vee \bar{x}yz.$$

Устраняя элементарные поглощения, мы придем к сокращенной д. н. ф. функции f:

$$f = xz \vee \bar{x}\,\bar{y}\,\bar{z}\,.$$

В § 1 нами был разобран так называемый алгебраический метод нахождения всех приведенных систем простых импликант булевой функции, основанный на получении приведенного дизъюнктивного представления импликантной

таблицы. Из теоремы 3.2 непосредственно следует, что приведенное дизъюнктивное представление импликантной таблицы может рассматриваться как сокращенная д.н.ф. булевой функции от простых импликант $A, B, C, \ldots$, задаваемой конъюнктивным представлением этой таблицы. Поэтому *для нахождения приведенного дизъюнктивного представления импликантной таблицы можно применять все методы, позволяющие находить сокращенные д.н.ф. булевых функций. Можно, в частности, к функции, задаваемой конъюнктивным представлением импликантной таблицы, применять любые тождественные преобразования булевой алгебры.*

Сделанное замечание дает обоснование развитого в § 1 метода нахождения приведенных систем импликант булевой функции. Основной вариант этого метода представляет собою фактически приложение описанного выше метода нахождения сокращенной д.н.ф. булевой функции, принадлежащего Нельсону. Теперь становится ясным, что для этой цели можно применить не только метод Нельсона, но и любой другой прием минимизации, описанный в §§ 2 и 3 этой главы.

Все описанные до настоящего времени методы минимизации булевых функций являются *аналитическими* методами. На практике для минимизации булевых функций от малого числа переменных оказываются более удобными методы, которые мы будем называть *графическими*. Эти методы основаны на использовании той особенности зрительного восприятия, что с его помощью можно практически мгновенно распознавать те или иные простые конфигурации.

Для того чтобы лучше уяснить основную идею графических методов, рассмотрим сначала простейший случай, а именно случай булевой функции $f(x,y)$, зависящей лишь от двух аргументов x и y.

Как известно, каждая такая функция может быть задана таблицей своих значений (см. табл. IV.6).

Для функции двух переменных существуют простые импликанты лишь двух видов: конституенты единицы $x\bar{y}$, $\bar{x}\bar{y}$, $\bar{x}y$, xy и первичные термы x, $\bar{x}$, y, $\bar{y}$. Каждая из простых импликант первого вида накрывает лишь одну единицу булевой функции, каждая импликанта второго вида —

две единицы. Ясно, однако, что любые две единицы булевой функции не могут, вообще говоря, быть накрыты одной простой импликантой второго вида. Для этой цели необходимо, чтобы эти единицы были специальным образом расположены в таблице функций. Необходимым и достаточным условием для возможности одновременного накрытия двух единиц одной простой импликантой является, очевидно, условие, чтобы эти единицы стояли в тех и только тех местах таблицы, в которых расположены единичные значения какой-либо из простых импликант.

Таблица IV.6

x	y	α
0	0	α_1
0	1	α_2
1	0	α_3
1	1	α_4

Введем следующее определение.

3.3. *При задании булевых функций таблицами их значений (имеющими ту или иную фиксированную форму) правильными конфигурациями в этих таблицах называются всевозможные множества мест в таблицах, состоящие из всех тех и только тех мест, в которых обращается в единицу какое-нибудь элементарное произведение. Если элементарное произведение p имеет длину i, то соответствующая ему правильная конфигурация называется конфигурацией ранга i ($i = 1, 2, \ldots, n$).*

Условимся обозначать правильные конфигурации единицами, проставленными во всех местах соответствующей конфигурации.

Для обычно принятой формы табличного задания значений булевых функций от двух аргументов правильные конфигурации ранга 1 имеют вид, представленный в табл. IV.7.

Таблица IV.7

x	y	$\bar{x}$	x	$\bar{y}$	y
0	0	1		1	
0	1	1			1
1	0		1	1	
1	1		1		1

Таблица IV.8

x	y	f
0	0	1
0	1	1
1	0	1
1	1	0

Запомнив эти конфигурации, легко находить минимальные д.н.ф. булевых функций от двух переменных. Действительно, для этого достаточно, очевидно, найти наиболее экономное накрытие единиц функции правильными конфигурациями. Так, например, все единицы функции f, заданной табл. IV.8, накрываются первой и третьей конфигурацией табл. IV.7. В соответствии с этим минимальная д.н.ф. функции f имеет вид $\bar{x} \vee \bar{y}$.

Определим более точно понятие накрытия и наиболее экономного (минимального) накрытия.

3.4. *Совокупность M правильных конфигураций образует накрытие булевой функции f, заданной таблицей своих значений, если на всех местах, входящих в конфигурации системы M, значения функции f либо равны единице, либо не определены и если любое место, в котором значение функции f равно единице, входит хотя бы в одну конфигурацию системы M. Рангом накрытия называется сумма рангов всех образующих накрытие правильных конфигураций. Накрытие функции называется минимальным, если его ранг не превосходит ранга любого другого накрытия этой функции с помощью правильных конфигураций.*

Каждой правильной конфигурации соответствует некоторое элементарное произведение, единицы которого (в таблице его значений) образуют данную конфигурацию. Нетрудно убедиться в справедливости следующего предложения.

3.5. *Дизъюнкция элементарных произведений, соответствующих всем правильным конфигурациям, которые образуют любое накрытие M булевой функции f, является дизъюнктивной нормальной формой F этой функции. Форма F тогда и только тогда является минимальной д.н.ф. функции f, когда накрытие M минимально.*

Теорема 3.5 сводит вопрос о минимизации булевых функций к нахождению минимальных накрытий этих функций правильными конфигурациями. Для облегчения запоминания правильных конфигураций оказывается целесообразным несколько изменить форму табличного задания булевых функций. Эти изменения включают изменение обычного порядка следования наборов значений переменных, а также использование вместо линейно

расположенных таблиц таблиц, имеющих форму прямоугольников или квадратов.

Преимущества, которые получаются в результате подобных изменений, можно оценить на примере функций двух переменных. Изменив порядок следования наборов значений переменных так, как это показано в табл. IV.9, можно получить более единообразную форму конфигураций ранга 1 (см. табл. IV.9).

Таблица IV.9

x	y	$\bar{x}$	x	$\bar{y}$	y
0	0	1		1	
0	1	1			1
1	1		1		1
1	0		1	1	

Таблица IV.10

x \ y	0	1
0	f_1	f_2
1	f_3	f_4

Если условиться мысленно отождествлять («склеивать») верхний и нижний край этой таблицы, то правильные конфигурации ранга 1 будут составлять все рядом расположенные пары единиц и только такие пары.

Еще более просто запомнить все конфигурации ранга 1 при изображении таблиц значений булевых функций двух переменных в форме квадрата (см. табл. IV.10). Легко проверить, что правильным конфигурациям ранга 1 в такой таблице соответствуют прямоугольники, составленные из любых двух соседних мест таблицы.

Нетрудно понять теперь смысл графических методов минимизации булевых функций: дело сводится к тому, чтобы подыскать такой способ записи таблиц значений булевых функций, при котором было бы наиболее просто запомнить все правильные конфигурации. Удобный способ записи таблиц булевых функций был предложен Вейчем [1]) и Карнау [2]). Таблицы функций, за-

[1]) E. W Veitch, A chart method for simplifying truth functions Proc Ass. Comp. Mach., 1952, May, № 2, 3, p. 127—133.

[2]) M. Karnaugh, The map method for synthesis of combinational logic circuits, Trans. AIEE, v. 72, № 1, 1953, p. 593—599.

писанные таким способом, носят название *карт Карнау* или *диаграмм Вейча*.

Карты Карнау (диаграммы Вейча) для булевых функций от трех и от четырех переменных изображены соответственно на табл. IV.11 и IV.12.

Таблица IV.11

x y \ z	0	1
0 0	f_0	f_1
0 1	f_2	f_3
1 1	f_6	f_7
1 0	f_4	f_5

Таблица IV 12

x y \ zu	00	01	11	10
0 0	f_0	f_1	f_3	f_2
0 1	f_4	f_5	f_7	f_6
1 1	f_{12}	f_{13}	f_{15}	f_{14}
1 0	f_8	f_9	f_{11}	f_{10}

В табл. IV.11 нужно отождествить мысленно ее верхний и нижний край (замкнуть таблицу в кольцо). При этом значения f_0 и f_4, а также значения f_1 и f_5 станут соседними (через f_i мы обозначаем значение функции на i-м наборе). В табл. IV.12 кроме отождествления верхнего и нижнего края необходимо также отождествить между собой левый и правый ее края. Для обозначения конфигураций в этих таблицах будем пользоваться геометрическими терминами. Например, конфигурацию табл. IV.12, образуемую местами f_3, f_7, f_{15} и f_{11}, естественно назвать прямоугольником, или, более точно, вертикальным (4 × 1)-прямоугольником. Конфигурация, образуемая местами f_8 и f_{11}, называется горизонтальным (2 × × 1)-прямоугольником; конфигурация, образуемая местами f_{15}, f_{14}, f_{11}, f_{10} — (2 × 2)-квадратом.

Следует обратить особое внимание на то обстоятельство, что, благодаря проведенным отождествлениям краев таблиц, возникают новые квадратные и прямоугольные конфигурации, которые не являлись таковыми до отождествления. Например, следует считать, что места f_1 и f_9 в таблице IV.12 образуют вертикальный (2 × 1)-прямоугольник, места f_0, f_2, f_8 и f_{10} — (2 × 2)-квадрат, верхняя и нижняя строка вместе взятые образуют горизонтальный (4 × 2)-прямоугольник, и т. д.

Для упрощения формулировок последующих теорем целесообразно называть произведение чисел m и n в любом $(m \times n)$-прямоугольнике *площадью* этого прямоугольника. При этом, разумеется, квадрат рассматривается как частный случай прямоугольника.

Теперь, после того как введены необходимые понятия и сделаны нужные разъяснения, можно сформулировать следующие предложения, справедливость которых нетрудно установить непосредственной проверкой (например, путем перебора конфигураций, соответствующих всем элементарным произведениям от трех и четырех переменных).

3.6. *Правильными конфигурациями ранга i на карте Карнау для булевых функций от трех переменных являются все прямоугольники (вертикальные, горизонтальные и квадратные), имеющие площадь 2^{3-i} $(i = 1, 2, 3)$, и только такие прямоугольники.*

3.7. *Правильными конфигурациями ранга i на карте Карнау для булевых функций от четырех переменных являются все прямоугольники (вертикальные, горизонтальные и квадратные), имеющие площадь 2^{4-i} $(i = 1, 2, 3, 4)$, и только такие прямоугольники.*

Идентичность формулировок теорем 3.6 и 3.7 приводит к мысли о возможности обобщения этих предложений на случай функций от произвольного числа n переменных такого, что конфигурации ранга i совпадают с прямоугольниками, имеющими площадь 2^{n-i} $(i = 1, 2, \ldots, n)$. Ясно, что для $n = 2$, равно как и для $n = 3$ и $n = 4$, этот результат справедлив. Однако уже для $n = 5$ нельзя построить карту Карнау по аналогии с картами Карнау для $n = 3$ и $n = 4$, для которой этот результат имел бы место. Поэтому для случая $n \geqslant 5$ либо прибегают к дальнейшему обобщению понятия прямоугольника, вводя в картах Карнау некоторые дополнительные отношения соседства (кроме соседства верхнего с нижним и левого с правым краев карты Карнау), либо используют пространственные диаграммы, где роль прямоугольников выполняют прямоугольные параллелепипеды. Эти обобщения (после приобретения необходимого навыка) дают возможность относительно просто производить минимизацию булевых функций от пяти и шести переменных.

Мы не будем, однако, выписывать соответствующих карт Карнау для пяти и шести переменных. В случае необходимости их легко может выписать каждый, кто ознакомится с картами Карнау для трех и четырех переменных.

На табл. IV.13 и IV.14 показаны, в виде примера, минимальные накрытия для частичных булевых функций f и g от трех и четырех переменных, заданных картами Карнау (места, в которых значения функций не определены, отмечены черточками).

Таблица IV.13

x	y \ z	0	1
0	0	0	1
0	1	1	—
1	1	1	0
1	0	1	1

Таблица IV.14

x	y \ zu	00	01	11	10
0	0	—	1	1	—
0	1	0	1	0	1
1	1	0	1	0	0
1	0	—	1	0	1

Построив накрытие M булевой функции на карте Карнау, нетрудно найти явные выражения для элементарных произведений, соответствующих всем конфигурациям этого накрытия, и построить определяемую данным накрытием дизъюнктивную нормальную форму. С этой целью достаточно выяснить, значения каких переменных остаются постоянными для всех наборов, определяющих места любой заданной правильной конфигурации из рассматриваемого накрытия. Нетрудно видеть, что все такие и только такие переменные входят в элементарное произведение p, соответствующее конфигурации K. Ясно также, что каждая такая переменная входит в элементарное произведение p в прямом или инверсном виде, соответственно тому, равняется ли ее значение на всех наборах, составляющих конфигурацию K, единице или нулю.

Используя описанный прием, мы легко найдем дизъюнктивные нормальные формы, соответствующие накрытиям, которые показаны в табл. IV.13 и IV.14:

$$f = \bar{x}z \vee yz \vee x\bar{y}, \quad g = x\bar{y} \vee zu \vee \bar{x}\,\bar{z}u \vee \bar{x}\,\bar{u} \vee yu.$$

Очевидно, что области карт Карнау, в которых значение любой переменной остается постоянным, представляют собой прямоугольники, имеющие площадь, равную 4 (для функций от трех переменных), и площадь, равную 8 (для функций от четырех переменных). Для того чтобы облегчить нахождение элементарных произведений, соответствующих правильным конфигурациям, эти области иногда выделяются на картах Карнау явным образом. Карты Карнау для функций от трех и четырех переменных с выделенными явно областями постоянства значений переменных изображены на табл. IV.15 и IV.16.

Таблица IV.15

x \ yz	1	0
0 0	f_1	f_1
0 1	f_2	f_3
1 1	f_6	f_7
1 0	f_4	f_5

Таблица IV.16

x \ zu	00	01	11	10
0 0	f_0	f_1	f_3	f_2
0 1	f_4	f_5	f_7	f_6
1 1	f_{12}	f_{13}	f_{15}	f_{14}
1 0	f_8	f_9	f_{11}	f_{10}

Используя размеченные карты Карнау, нетрудно сразу определить, например, что квадрат, составленный из мест f_1, f_3, f_9, f_{11}, соответствует элементарному произведению yu, первая строка в табл. IV.15 — элементарному произведению $\bar{x}\bar{y}$, и т. д.

§ 4. Проблема факторизации. Минимальные конъюнктивные нормальные формы

Методы минимизации булевых функций, развитые в предыдущих параграфах, ограничиваются решением лишь одной задачи, а именно задачи нахождения минимальной

дизъюнктивной нормальной формы булевой функции. Нетрудно видеть, однако, что минимальная д.н.ф. далеко не во всех случаях дает самое простое выражение для булевой функции в булевой алгебре. В самом деле, легко видеть, что $f = xy \vee xz$ — минимальная д.н.ф. (см. предложение 1.12). Используя ее в качестве формы представления булевой функции f, мы выражаем эту функцию с помощью трех операций (двух умножений и одной дизъюнкции). Вместе с тем, вынося x за скобки, мы находим для той же функции выражение, в котором использовано лишь две операции (дизъюнкция и умножение): $f = x(y \vee z)$.

Проблема построения выражения для булевой функции, использующего наименьшее возможное число операций дизъюнкции и умножения среди всех выражений, представляющих данную функцию и построенных из первичных термов $\bar{x}$, x, $\bar{y}$, y, ... с помощью одних лишь дизъюнкций и умножений, носит название *проблемы факторизации*.

К сожалению, в настоящее время не имеется общих методов решения этой проблемы, существенно более простых, чем метод перебора всех формул булевой алгебры, построенных из первичных термов с помощью одних лишь дизъюнкций и умножений и использующих ограниченное число этих операций, не превосходящее суммарное число дизъюнкций и умножений в минимальной д.н.ф. рассматриваемой булевой функции.

Обычно в случае практического решения проблемы факторизации не ставят целью найти непременно самое экономное выражение для данной булевой функции, а ограничиваются лишь некоторым уменьшением сложности ее минимальной д.н.ф. (в случае, когда это оказывается возможным), используя разложение на множители тех или иных групп членов, составляющих эту форму.

Нетрудно показать, что при использовании лишь обычных приемов разложения на множители (подходящей группировки членов и вынесения общих множителей за скобки) не представляется возможным во всех случаях получить самое экономное выражение для булевой функции, если отправляться от ее минимальной д.н.ф.

Соответствующие примеры были построены Беркхартом [1]). Рассмотрим один из примеров Беркхарта.

Пусть булева функция f задана дизъюнктивной нормальной формой

$$f = xyv \vee xyw \vee xzuv \vee yzuw.$$

Из теоремы 3.1 вытекает, что эта форма является сокращенной, но тогда с помощью теоремы 1.11 мы устанавливаем, что она является минимальной д.н.ф. Непосредственная группировка членов и вынесение общих множителей за скобки приводит к выражению

$$f_1 = xy\,(v \vee w) \vee zu\,(xv \vee yw),$$

содержащему 9 операций умножения и дизъюнкции (6 умножений и 3 дизъюнкции). Нетрудно проверить, что это самое экономное выражение из всех выражений, которые могут быть получены из минимальной д.н.ф. с помощью группировки членов и вынесения общих множителей за скобки.

Нетрудно видеть, однако, что $xy(v \vee w) = xy(xv \vee yw)$. Используя это обстоятельство, можно продолжить разложение на множители и получить для функции f выражение $f_2 = (xy \vee zu)(xv \vee yw)$, содержащее лишь 7 операций дизъюнкции и умножения (2 дизъюнкции и 5 умножений).

В ряде случаев при решении проблемы нахождения абсолютно минимального представления булевой функции целесообразно вместо минимальных дизъюнктивных нормальных форм употреблять минимальные конъюнктивные нормальные формы. Теория минимизации конъюнктивных нормальных форм (к.н.ф.) оказывается вполне параллельной теории минимизации д.н.ф. Введем определения, необходимые для построения этой теории. Прежде всего необходимо определить аналоги понятий импликанты и простой импликанты, которые мы будем называть *имплицентой* и *простой имплицентой*.

4.1. *Имплицентой булевой функции* $f(x_1, \ldots, x_n)$ *называется всякая булева функция* $g(x_1, \ldots, x_n)$, *которая обращается в нуль лишь в тех точках (не обязательно во всех), в которых значения функции* f *равны нулю или не*

[1]) W. H. Burkhart, Theorem minimization, Proc. Ass. Comp. Mach. May, № 2 and 3, 1952, p. 259—263.

определены. Простой имплицентой булевой функции f называется любая элементарная дизъюнкция p, являющаяся имплицентой функции f и такая, что никакая собственная часть этой дизъюнкции (дизъюнкция части членов элементарной дизъюнкции p) не является имплицентой функции f.

Заметим, что к числу элементарных дизъюнкций мы относим также и константу н у л ь (рассматриваемую в качестве дизъюнкции пустого множества первичных термов). Ясно, что константа нуль может рассматриваться как простая имплицента лишь одной единственной булевой функции, а именно функции, тождественно равной нулю. Этот случай мы будем считать тривиальным и в дальнейшем исключать из рассмотрения. Во всех остальных случаях все простые имплиценты булевых функций отличны от нуля.

Нетрудно убедиться в том, что произведение любого множества имплицент булевой функции снова является имплицентой той же самой функции. Если это произведение равно рассматриваемой функции, мы будем говорить, что составляющие его имплиценты образуют *полную систему имплицент* этой функции. Полная система имплицент называется *приведенной*, если никакая собственная часть этой системы не является полной системой имплицент.

Имплиценты булевой функции f, образующие полную систему, *накрывают* все нули этой функции. Иными словами, для любого набора значений переменных, на котором функция f обращается в нуль, найдется имплицента данной системы, принимающая на этом наборе нулевое значение. Ясно, что выполнение этого условия необходимо и достаточно для полноты системы имплицент.

Точно так же, как и в случае простых импликант, доказывается утверждение, гласящее, что *система всех простых имплицент булевой функции представляет собою полную систему*. Этот результат позволяет определить (по аналогии с сокращенной д.н.ф.) *сокращенную конъюнктивную нормальную форму*.

4.2. *Сокращенной к.н.ф. булевой функции называется произведение всех ее простых имплицент. Произведение имплицент булевой функции f, составляющих какую-нибудь приведенную систему простых имплицент этой функции, называется тупиковой к.н.ф. функции f.*

Если назвать *минимальной к.н.ф.* булевой функции такую ее к.н.ф., которая содержит наименьшее возможное число первичных термов (переменных и их отрицаний), то, как и в случае минимальных д.н.ф., нетрудно доказать следующее предложение.

4.3. *Любая минимальная к.н.ф. булевой функции является тупиковой к.н.ф. (но, вообще говоря, не наоборот).*

Как и минимизация д.н.ф., минимизация к.н.ф. обычно проводится в два этапа. На первом этапе находятся все простые имплиценты данной булевой функции или, что то же самое,— ее сокращенная к.н.ф.; на втором этапе из найденных простых имплицент строятся *минимальные накрытия* (приведенные системы), образующие тупиковые к.н.ф., а из тупиковых к.н.ф. выбираются минимальные к.н.ф.

Для решения задачи второго этапа может употребляться *имплицентная таблица*, строящаяся по аналогии с импликантной таблицей, рассмотренной в § 1. Строки имплицентной таблицы обозначаются простыми имплицентами данной булевой функции, а столбцы — наборами значений переменных (или соответствующими им конституентами нуля), на которых эта функция обращается в нуль. При этом используются все простые имплиценты и все наборы с нулевыми значениями функции. На пересечении A-й строки и P-го столбца имплицентной таблицы ставится звездочка, если имплицента A обращается в нуль на наборе P. Если же имплицента A на этом наборе равна единице, то соответствующее место таблицы оставляется пустым.

Точно так же, как и в § 1, строится конъюнктивное представление имплицентной таблицы. Превращая это представление в дизъюнктивное (с помощью раскрытия скобок) и осуществляя упрощения в соответствии с правилами булевой алгебры (элементарные поглощения), мы получаем дизъюнкцию произведений простых имплицент. Каждое из таких произведений является тупиковой к.н.ф. исходной булевой функции, причем таким способом получаются все тупиковые (а, значит, и все минимальные) конъюнктивные нормальные формы.

Заметим, что в соответствии с дуальностью операций дизъюнкции и умножения, имеющий место в булевой ал-

гебре, весьма естественно распространить эту дуальность на процесс раскрытия импликантных и имплицентных таблиц. Иными словами, в данном случае нужно от исходного конъюнктивного представления таблицы переходить к дизъюнктивному представлению, а в другом случае — наоборот. Однако мы сделали отступление от принципа дуальности, что было вызвано главным образом соображениями практического удобства. Дело в том, что переход от конъюнктивного представления к дизъюнктивному использует правила раскрытия скобок, аналогичные соответствующим правилам школьного курса алгебры, обратный же переход менее обычен, в связи с чем, как показывает практика, увеличивается вероятность ошибок при проведении выкладок.

Для нахождения простых имплицент булевой функции можно использовать все методы (соответствующим образом перефразированные) нахождения простых импликант, которые были развиты в двух предыдущих параграфах. Условимся сохранять для получаемых таким образом методов введенные выше названия.

В случае м е т о д а К в а й н а для нахождения простых имплицент вместо операции неполного дизъюнктивного склеивания нужно употреблять операцию *неполного конъюнктивного склеивания*, основанную на тождестве

$$(A \vee x)(A \vee \bar{x}) = A\,(A \vee x)(A \vee \bar{x}).$$

Вместо операции обобщенного (дизъюнктивного) склеивания по м е т о д у Б л е й к а нужно употреблять операцию *обобщенного конъюнктивного склеивания*, задаваемую тождественным соотношением:

$$(A \vee x)(B \vee \bar{x}) = (A \vee x) \setminus B \vee \bar{x})(A \vee B).$$

Справедливыми являются также аналоги теорем К в а й н а, Б л е й к а и Н е л ь с о н а:

4.4. *Применяя к совершенной к.н.ф. булевой функции все возможные операции неполного конъюнктивного склеивания и устраняя элементарные поглощения, мы получаем сокращенную к.н.ф. этой функции.*

4.5. *Применяя к произвольной к.н.ф. булевой функции все возможные операции обобщенного конъюнктивного*

склеивания и устраняя элементарные поглощения, мы приходим к сокращенной к.н.ф. этой функции.

4.6. *Если произвольную д.н.ф. булевой функции преобразовать в конъюнктивную нормальную форму, используя второй дистрибутивный закон* $A \vee BC = (A \vee B)(A \vee C)$, *и устранить элементарные поглощения, то мы приходим к сокращенной к.н.ф. этой функции.*

Элементарное поглощение во всех этих теоремах понимается в конъюнктивном смысле; именно, при использовании тождественного соотношения $A(A \vee B) = A$ говорят, что множитель A осуществляет *элементарное поглощение* множителя $A \vee B$. Доказательства сформулированных теорем мы не будем приводить, поскольку они являются перефразировками доказательств теорем Квайна, Блейка и Нельсона, приведенных в §§ 2 и 3 настоящей главы.

Весьма просто осуществляется также перефразировка (применительно к нахождению минимальных к.н.ф.) графических приемов минимизации, основанных на использовании карт Карнау. Различие состоит в том, что вместо конфигураций единиц рассматриваются конфигурации нулей. Задача минимизации состоит в минимальном накрытии всех нулей заданной булевой функции правильными конфигурациями нулей, соответствующими нулям элементарных дизъюнкций. Как и ранее, нетрудно удостовериться в том, что правильными конфигурациями являются обобщенные прямоугольники (с учетом отождествления противоположных сторон на картах Карнау).

Рассмотрим в качестве примера нахождение (с помощью карты Карнау) минимальной к.н.ф. частичной булевой функции от четырех переменных, заданной табл. IV.17 (для наборов значений переменных, отсутствующих в таблице, значения функции безразличны). Решение задачи дается картой Карнау, изображенной в табл. IV.18.

Из этой таблицы видно, что имеются три простые имплиценты заданной функции, образующие приведенную систему. Первой из этих имплицент является имплицента y, которой соответствует обобщенный прямоугольник, составленный из верхней и нижней строки карты Карнау (заметим, что при нахождении простых импликант, а не простых имплицент этому прямоугольнику соответст-

вовала бы простая импликанта $\bar{y}$; различие получается вследствие того, что мы оперируем теперь с конфигурациями нулей, а не с конфигурациями единиц).

Таблица IV.17

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

Таблица IV.18

x	y \ zu	00	01	11	10
0	0	0	—	—	—
0	1	—	1	0	—
1	1	—	—	1	—
1	0	—	—	0	—

Вторая простая имплицента u соответствует обобщенному прямоугольнику, состоящему из крайних (левого и правого) столбцов карты Карнау. Третья простая имплицента $x \vee \bar{z}$ соответствует (2×2)-квадрату в правом верхнем углу таблицы.

Искомая минимальная к.н.ф. функции f совпадает с произведением всех трех найденных простых имплицент: $f = yu(x \vee \bar{z})$.

Заметим еще, что нахождение минимальных к.н.ф. можно полностью свести к нахождению минимальных д.н.ф. Такое сведение осуществляется в соответствии со следующей теоремой.

4.7. *Если в таблице заданной (частичной) булевой функции заменить все нули единицами, а все единицы нулями, то возникает взаимно однозначное соответствие* φ *между множеством всех минимальных д.н.ф. полученной таким образом (частичной) булевой функции* g *и множеством всех минимальных к.н.ф. функции* f*: если* g_0 *— произвольная минимальная д.н.ф. функции* g*, то* $\varphi(g_0) = f_0$*, причем к.н.ф.* f_0 *получается из д.н.ф.* g_0 *заменой всех дизъюнкций умножениями и всех умножений дизъюнкциями при одновременной замене всех первичных термов* x_i *термами* $\bar{x}_i$*, а первичных термов* $\bar{x}_j$ *— термами* x_j.

В целях доказательства предложения 4.7 заметим, прежде всего, что выражение f_0, получаемое из д.н.ф. g_0 в соответствии с предложением 4.7, представляет собой некоторую к.н.ф. Из предложения 4.6 гл. III вытекает, что функция, задаваемая формой f_0, представляет собою отрицание функции, задаваемой формой g_0. Поскольку g_0 является некоторым представлением функции g, то из способа построения функции g непосредственно следует, что f_0 является некоторым представлением функции f. Будучи к.н.ф., выражение f_0 является к.н.ф. функции f.

Отображение φ может быть определено на всех д.н.ф. функции g. Ясно также, что для этого отображения существует обратное отображение φ^{-1}, которое может быть определено на множестве всех к.н.ф. функции f. Далее, отображения φ и φ^{-1} не меняют количества первичных термов, составляющих нормальную форму. Отсюда непосредственно следует, что эти отображения переводят минимальные нормальные формы (д.н.ф. функции g и к.н.ф. функции f) в минимальные, что и требовалось доказать.

Применяя доказанную теорему, теперь можно иначе интерпретировать проведенное выше построение минимальной к.н.ф. для булевой функции f, заданной табл. IV.16. Для этой цели достаточно на карте Карнау, задаваемой табл. IV. 17, произвести мысленную замену всех единиц нулями, а всех нулей — единицами. Для построенной таким образом новой функции g мы находим минимальную д.н.ф. $g = \overline{y} \vee \overline{u} \vee \overline{x}z$. Используя отображение φ, мы приходим к к.н.ф. $f_0 = yu(x \vee \overline{z})$, которая, в силу теоремы 4.7, является минимальной к.н.ф. исходной функции f.

В заключение решим задачу факторизации для случая булевых функций от трех переменных.

Нетрудно убедиться в том, что найдя для некоторой булевой функции f выражение, построенное с помощью операций дизъюнкции и умножения из наименьшего возможного числа первичных термов, мы можем считать построенными аналогичные выражения для всех функций, получаемых из функции f с помощью всевозможных перестановок переменных и преобразований переменных вида $x_i \to \overline{x}_i$. Иначе говоря, проблему нахождения абсолютно минимального выражения (проблему факторизации

зации) достаточно решать лишь для одного какого-нибудь представителя каждого типа булевых функций.

Как уже отмечалось выше, существует всего 22 различных типа булевых функций от трех переменных. Их нетрудно найти с помощью прямого перебора. Среди этих 22 типов 6 типов составляют функции, которые сводятся к функциям от меньшего числа переменных. Кроме констант 0 и 1, это будут типы, задаваемые следующими своими представителями:

$$x,\ xy,\ x \vee y,\ xy \vee \bar{x}\bar{y} = (x \vee \bar{y})(\bar{x} \vee y).$$

Нетрудно проверить, что выписанные выражения будут абсолютно минимальными представлениями соответствующих функций.

Для остающихся 16 типов (от 7-го до 22-го) выпишем по одному представителю каждого типа, задавая для них все минимальные д.н.ф. и к.н.ф., а также абсолютно минимальные выражения:

7-й тип: $f_7 = xyz.$

8-й тип: $f_8 = xyz \vee \bar{x}\bar{y}\bar{z} = x(y \vee z)(\bar{y} \vee \bar{z}).$

9-й тип: $f_9 = \bar{x}\bar{y}\bar{z} \vee xyz =$
$= (x \vee \bar{y})(\bar{x} \vee z)(y \vee \bar{z}) = (\bar{x} \vee y)(x \vee \bar{z})(\bar{y} \vee z).$

10-й тип: $f_{10} = xy \vee xz = x(y \vee z).$

11-й тип: $f_{11} = \bar{x}\bar{y} \vee xyz =$
$= (\bar{x} \vee y)(x \vee \bar{y})(\bar{y} \vee z) = (\bar{x} \vee y)(x \vee \bar{y})(\bar{x} \vee z).$

12-й тип: $f_{12} = xyz \vee x\bar{y}\bar{z} \vee \bar{x}\,y\bar{z} =$
$= (x \vee y)(y \vee \bar{z})(x \vee \bar{z})(\bar{x} \vee \bar{y} \vee z) = x(yz \vee \bar{y}\bar{z}) \vee \bar{x}\,y\bar{z}.$

13-й тип: $f_{13} = xy \vee yz \vee xz =$
$= (x \vee y)(x \vee z)(y \vee z) = x(y \vee z) \vee yz.$

14-й тип: $f_{14} = xz \vee y\bar{z} = (x \vee \bar{z})(y \vee z).$

15-й тип: $f_{15} = xy \vee xz \vee \bar{x}\,\bar{y}\,\bar{z} =$
$= (x \vee \bar{y})(x \vee \bar{z})(\bar{x} \vee y \vee z) = x(y \vee z) \vee \bar{x}\overline{yz}.$

16-й тип: $f_{16} = xyz \vee x\bar{y}\,\bar{z} \vee \bar{x}\,y\bar{z} \vee \bar{x}\,\bar{y}z =$
$= (x \vee y \vee z)(\bar{x} \vee y \vee \bar{z})(\bar{x} \vee \bar{y} \vee z)(x \vee \bar{y} \vee \bar{z}) =$
$= x(yz \vee \bar{y}\bar{z}) \vee \bar{x}(y\bar{z} \vee \bar{y}z).$

17-й тип: $f_{17}=xy\vee xz\vee yz\vee\overline{\overline{xy}\,\overline{z}}=$
$=(x\vee y\vee\overline{z})\,(\overline{x}\vee y\vee z)\,(x\vee\overline{y}\vee z)=$
$=x\,(y\vee z)\vee yz\vee\overline{x}\,\overline{yz}\,.$

18-й тип: $f_{18}=\overline{x}y\vee x\overline{y}\vee\overline{xz}=$
$=\overline{x}y\vee x\overline{y}\vee y\overline{z}=(x\vee y)\,(\overline{x}\vee\overline{y}\vee\overline{z}).$

19-й тип: $f_{19}=x\vee yz=(x\vee z)\,(x\vee y).$

20-й тип: $f_{20}=\overline{xy}\vee x\overline{z}\vee y\overline{z}=$
$=x\overline{y}\vee\overline{x}z\vee y\overline{z}=(\overline{x}\vee\overline{y}\vee\overline{z})\,(x\vee y\vee z).$

21-й тип: $f_{21}=x\vee\overline{yz}\vee y\overline{z}=(x\vee y\vee z)\,(x\vee\overline{y}\vee\overline{z}).$

22-й тип: $f_{22}=x\vee y\vee z.$

Заметим, что функция f_{20} имеет пять различных тупиковых д.н.ф., из которых только две минимальные. Аналогично функция f_8 имеет пять различных тупиковых к.н.ф., из которых только две минимальные. У всех остальных типов булевых функций от трех переменных все тупиковые д.н.ф. и к.н.ф. являются минимальными. При этом, кроме 9-го и 20-го типов, еще только функция f_{11} имеет две различные минимальные к.н.ф., а функция f_{18} — две различные минимальные д.н.ф. Все остальные минимальные д.н.ф. и к.н.ф. являются единственными.

ГЛАВА V

МЕТОДЫ ПОСТРОЕНИЯ КОМБИНАЦИОННЫХ СХЕМ В ДВОИЧНОМ СТРУКТУРНОМ АЛФАВИТЕ

§ 1. Некоторые методы решения канонической задачи комбинационного синтеза

В главе III (§ 6) было введено понятие канонической задачи комбинационного синтеза. Для случая n переменных $x_1, \ldots, x_n$ в канонической задаче имеется $2n + 2$ входных полюса, на которые подаются сигналы

$$x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, 0, 1.$$

Число выходных полюсов может быть произвольным, однако, разумеется, можно считать, что оно не превосходит числа всех булевых функций от n переменных, то есть 2^{2^n}. В качестве логических элементов в канонической задаче используются лишь двухвходовые совпадения и разделения.

При решении канонической задачи комбинационного синтеза на практике достаточно хорошие результаты получаются при применении так называемого *метода каскадов*, принадлежащего Г. Н. Поварову[1]). Этот метод является уточнением метода, предложенного ранее К. Шенноном[2]). Первоначально он был развит применительно к релейно-контактным схемам. Мы дадим несколько иную интерпретацию этого метода, ориентирующуюся

---

[1]) Г. Н. Поваров, Математическая теория синтеза контактных $(1,k)$-полюсников, ДАН СССР, т. 100, № 5, 1955, стр. 909—912.

[2]) C. E. Shannon, The synthesis of two-terminal switching circuits. Bell. System Tech. J., v. 28, 1949, p. 59—98.

на использование двухвходовых совпадений и разделений.

Метод каскадов основывается на следующей формуле булевой алгебры, имеющей место для произвольной булевой функции $f(x_1, x_2, \ldots, x_n)$:

$$f(x_1, x_2, \ldots, x_n) = x_n f(x_1, x_2, \ldots, x_{n-1}, 1) \vee \bar{x}_n f(x_1, x_2, \ldots, x_{n-1}, 0). \quad (1)$$

Справедливость этой формулы легко проверяется непосредственной подстановкой в ее левую и правую часть значений $x_n = 0$ и $x_n = 1$.

Предположим теперь, что перед нами поставлена задача синтеза комбинационной схемы, выходными функциями которой служат функции системы m булевых функций $f_1, f_2, \ldots, f_m$ от n переменных $x_1, \ldots, x_n$. Применим формулу (1) к каждой из функций f_i ($i = 1, \ldots, m$):

$$f_i(x_1, \ldots, x_{n-1}, x_n) = x_n f_i(x_1, \ldots, \bar{x}_{n-1}, 1) \vee \bar{x}_n f_i(x_1, \ldots, x_{n-1}, 0),$$

где $i = 1, \ldots, n$. Если построить комбинационную схему, выходными функциями которой будут функции от $n-1$ переменных $f_i(x_1, \ldots, x_{n-1}, 1)$, $f_i(x_1, \ldots, x_{n-1}, 0)$, где $i = 1, \ldots, m$, то искомая схема с выходными функциями $f_i(x_1, \ldots, x_{n-1}, x_n)$, где $i = 1, \ldots, m$, может быть легко построена с помощью двухвходовых совпадений и двухвходовых разделений, реализующих функции вида $\varphi_1 x_n \vee \varphi_2 \bar{x}_n$.

Назовем участок схемы, реализующий эти функции последним *каскадом* искомой комбинационной схемы. Описанный прием выделения последнего каскада комбинационной схемы позволяет сводить задачу синтеза этой схемы к задаче синтеза схемы, имеющей в качестве своих выходных функций булевы функции от меньшего числа переменных. Осуществляя такое сведение последовательно $n-2$ раза, мы сведем исходную задачу синтеза к задаче синтеза схемы, реализующей некоторые булевы функции от двух переменных. Эта же последняя задача решается непосредственно с помощью образования совпадений сигналов x_1, $\bar{x}_1$ с сигналами x_2, $\bar{x}_2$ и последующих разделений образующихся таким образом сигналов.

В итоге мы получаем метод решения канонической задачи комбинационного синтеза, представляющий схему, выходные функции которой зависят от n переменных, в виде объединения последовательно включенных $n-1$ каскадов. Заметим, что тот же метод в применении к вентильным (в частности, к релейно-контактным) схемам более естественно интерпретировать таким образом, что соответствующая схема разбивается не на $n-1$, а на n каскадов.

В качестве примера синтеза с помощью метода каскадов рассмотрим синтез комбинационной схемы с выходными функциями:

$$f_n = x_1 + x_2 + \ldots + x_n,$$
$$g_n = x_1 + x_2 + \ldots + x_n + 1.$$

Применяя формулу (1), мы получим, что

$$f_n = g_{n-1}x_n \vee f_{n-1}\bar{x}_n,$$
$$g_n = f_{n-1}x_n \vee g_{n-1}\bar{x}_n,$$

где $f_{n-1} = x_1 + x_2 + \ldots + x_{n-1}$, а $g_{n-1} = x_1 + x_2 + \ldots + x_{n-1} + 1$. Вообще, обозначая через f_i и g_i функции $x_1 + x_2 + \ldots + x_i$ и $x_1 + x_2 + \ldots + x_i + 1$, соответственно, $(i = 2, 3, \ldots, n)$, мы получим следующие соотношения;

$$f_2 = x_1\bar{x}_2 \vee \bar{x}_1 x_2,$$
$$g_2 = \bar{x}_1\bar{x}_2 \vee x_1 x_2,$$
$$f_i = g_{i-1}x_i \vee f_{i-1}\bar{x}_i,$$
$$g_i = f_{i-1}x_i \vee g_i\bar{x}_i,$$

где $i = 3, 4, \ldots, n$.

Таким образом, приходим к схеме из $n-1$ каскадов, каждый каскад которой состоит из четырех двухвходовых совпадений и двух двухвходовых разделений. Вся схема в целом содержит, следовательно, $4(n-1)$ совпадений и $2(n-1)$ разделений.

Основываясь на методе, аналогичном методу каскадов, К. Шеннон[1]) предложил способ оценки сложности

---

[1]) См. его работу, указанную в примечании 2 на стр. 317.

комбинационной схемы, реализующей любую заданную булеву функцию от n переменных. Схема, предложенная Шенноном, основана на объединении двух схем, первая из которых называется *универсальным многополюсником*, а вторая — *обратным логическим деревом*.

Универсальным многополюсником от n переменных мы будем называть любую комбинационную схему, выходными функциями которой служат все булевы функции от n переменных.

Следующий результат представляет собой небольшую модификацию известного результата К. Шеннона[1]).

1.1. *Для любого натурального числа $n \geqslant 2$ можно построить универсальный многополюсник от n переменных $x_1, \ldots, x_n$, использующий в качестве входных сигналов сигналы $0, 1, x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$ и содержащий не более $2 \cdot 2^{2^n} - 28$ двухвходовых совпадений и не более $2^{2^n} - 10$ двухвходовых разделений.*

Доказательство теоремы 1.1 проводится индукцией по числу переменных n. При $n = 2$ мы имеем $2^{2^2} = 16$ различных булевых функций от двух переменных x_1 и x_2. Из них шесть функций, а именно функции $0, 1, x_1, x_2, \bar{x}_1, \bar{x}_2$ совпадают с соответствующими входными сигналами и не требуют, следовательно, для своего построения никаких логических элементов. Остальные десять функций можно представить в виде

$$x_1x_2,\ x_1\bar{x}_2,\ \bar{x}_1x_2,\ \bar{x}_1\bar{x}_2,\ x_1\bar{x}_2 \vee \bar{x}_1x_2,$$
$$\bar{x}_1\bar{x}_2 \vee x_1x_2,\ x_1 \vee x_2,\ \bar{x}_1 \vee x_2,\ x_1 \vee \bar{x}_2,\ \bar{x}_1 \vee \bar{x}_2.$$

Учитывая, что пятая и шестая функции представляются в виде дизъюнкций второй с третьей и первой с четвертой функцией, мы приходим к выводу, что для построения универсального многополюсника в случае двух переменных достаточно четырех (двухвходовых) совпадений и шести (двухвходовых) разделений. Таким образом, при $n = 2$ оценки, приведенные в теореме 1.1, справедливы.

Предположим теперь, что нами уже доказана справедливость оценок $C_n \leqslant 2 \cdot 2^{2^n} - 28$ и $P_n \leqslant 2^{2^n} - 10$ для числа

[1]) См. его работу, указанную в примечании 2 на стр. 317.

совпадений C_n и числа разделений P_n в универсальном многополюснике при $n = k$. Докажем, что эти же оценки имеют место и при $n = k+1$.

Прежде всего заметим, что универсальный многополюсник для $k+1$ переменных можно построить из универсального многополюсника для k переменных с помощью использования формулы

$$f(x_1, \ldots, x_k, x_{k+1}) = x_{k+1} f(x_1 \ldots, x_k, 1) \vee \bar{x}_{k+1} f(x_1, \ldots, x_k, 0),$$

справедливой для каждой из $2^{2^{k+1}}$ булевых функций от $k+1$ переменных. Эта формула показывает, что переход от универсального многополюсника для k переменных к универсальному многополюснику для $k+1$ переменных может быть выполнен с помощью добавления в схему k-го каскада, содержащего $2 \cdot 2^{2^{k+1}}$ (двухвходовых) совпадений и $2^{2^{k+1}}$ (двухвходовых) разделений.

Возможно, однако, и более экономное построение k-го каскада. Действительно, ровно 2^{2^k} булевых функций от $k+1$ переменных может быть представлено в виде $x_{k+1}\varphi(x_1, \ldots, x_k)$, и ровно 2^{2^k} функций в виде $\bar{x}_{k+1}\psi(x_1, \ldots, x_k)$. Ясно, что среди этих функций нет одинаковых. Учитывая наличие функций с такими разложениями, мы можем при построении k-го каскада схемы универсального многополюсника от $k+1$ переменных сэкономить $2 \cdot 2^{2^k}$ совпадений (по одному на каждой функции) и $2 \cdot 2^{2^k}$ разделений.

Таким образом, мы имеем следующие соотношения:

$$C_{k+1} \leqslant C_k + 2 \cdot 2^{2^{k+1}} - 2 \cdot 2^{2^k}, \quad P_{k+1} \leqslant P_k + 2^{2^{k+1}} - 2 \cdot 2^{2^k}. \quad (2)$$

Поскольку $C_k \leqslant 2 \cdot 2^{2^k} - 28$, $P_k \leqslant 2^{2^k} - 10$, то из соотношений (2) мы получаем соотношения:

$$C_{k+1} \leqslant 2 \cdot 2^{2^{k+1}} - 28, \quad P_{k+1} \leqslant 2^{2^{k+1}} - 10.$$

Тем самым совершен требуемый переход от k к $k+1$, и закончено доказательство теоремы [1].

---

[1]) Легко видеть, что при $n > 2$ оценка, полученная нами для числа разделений P_n, является сильно завышенной.

Рассмотрим теперь одно обобщение тождества (1). Пусть $f(x_1, \ldots, x_n)$ — произвольная булева функция от n переменных, а m — произвольное натуральное число, меньшее чем n. Непосредственной проверкой с помощью подстановки различных значений переменных x_{m+1}, x_{m+2}, ..., x_n может быть проверена справедливость следующего соотношения:

$$f(x_1, \ldots, x_n) = \bar{x}_{m+1}\bar{x}_{m+2}\ldots\bar{x}_n f(x_1, \ldots, x_m, 0, 0, \ldots, 0) \vee$$
$$\vee \bar{x}_{m+1}\bar{x}_{m+2}\ldots\bar{x}_{n-1}x_n f(x_1, \ldots, x_m, 0, 0, \ldots, 0, 1) \vee \ldots$$
$$\ldots \vee x_{m+1}x_{m+2}\ldots x_n f(x_1, \ldots, x_m, 1, 1, \ldots, 1). \qquad (3)$$

Для реализации соотношения (3) может быть построена специальная комбинационная схема, называемая *обратным логическим деревом*. Эта схема имеет $n - m$ каскадов. В первом каскаде входными сигналами служат

$$x_{m+1}, \bar{x}_{m+1}, f(x_1, \ldots, x_m, \alpha_{m+1}, \alpha_{m+2}, \ldots, \alpha_n),$$

а выходными сигналами — функции вида

$$\bar{x}_{m+1} f(x_1, \ldots, x_m, 0, \alpha_{m+2}, \ldots, \alpha_n) \vee$$
$$\vee x_{m+1} f(x_1, \ldots, x_m, 1, \alpha_{m+1}, \ldots, \alpha_n).$$

Здесь через α_{m+1}, α_{m+2}, ..., α_n обозначен произвольный набор значений переменных x_{m+1}, x_{m+2}, ..., x_n. Всего в первом каскаде будет содержаться 2^{n-m} двухвходовых совпадений и $\frac{1}{2} \cdot 2^{n-m}$ двухвходовых разделений.

Выходные сигналы первого каскада, число которых равно $\frac{1}{2} \cdot 2^{n-m}$, принимаются вместе с сигналами x_{m+2} и $\bar{x}_{m+2}$ за входные сигналы второго каскада, который строится точно таким же образом, как и первый каскад. Продолжая подобным образом, мы получим на выходе $(n - m)$-го каскада заданную функцию $f(x_1, \ldots, x_n)$. Общее число затрачиваемых на построение всей схемы (состоящей из $n - m$ каскадов) двухвходовых совпадений и разделений равно соответственно

$$2^{n-m} + 2^{n-m-1} + \ldots + 2 \text{ и } \frac{1}{2}(2^{n-m} + 2^{n-m-1} + \ldots + 2).$$

Первое из этих чисел, очевидно, равно $2 \cdot 2^{n-m}-2$, а второе $2^{n-m}-1$.

Функции $f(x_1, \ldots, x_m, \alpha_{m+1}, \ldots, \alpha_n)$, использованные при осуществлении вышеприведенного построения, можно получить в качестве выходных функций (возможно, не всех) универсального многополюсника от m переменных. Объединяя оценки для числа совпадений и разделений в универсальном многополюснике (теорема 1.1) с только что полученными оценками для числа совпадений и разделений в обратном логическом дереве, мы приходим к следующему результату.

1.2. *Любая булева функция $f(x_1, \ldots, x_n)$ от n переменных при $n \geqslant 2$ может быть реализована как выходная функция комбинационной схемы, входными сигналами которой служат* $0, 1, x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$ *и которая содержит не более чем* $2 \cdot (2^{2^m}+2^{n-m})-30$ *двухвходовых совпадений и не более чем* $2^{2^m}+2^{n-m}-11$ *двухвходовых разделений (m — любое натуральное число, такое, что $2 \leqslant m \leqslant n$).*

Заметим, что приведенные в теоремах 1.1 и 1.2 оценки числа совпадений и особенно оценки числа разделений могут быть улучшены. Однако мы не будем стремиться к возможным улучшениям, поскольку нашей целью является, в первую очередь, не получение оценок, а нахождение практически удобных методов синтеза комбинационных схем. Нетрудно убедиться в том, что построения, примененные при доказательстве теорем 1.1 и 1.2, представляют собой разновидности метода каскадов.

Из теоремы 1.2 можно получить принадлежащую К. Шеннону асимптотическую оценку сложности комбинационных схем, реализующих булевы функции от n переменных. Для этой цели заметим, что при $n>4$ оценки, приведенные в теореме 1 2, принимают достаточно хорошие (с точки зрения их величины) значения в том случае, если выбрать значение m равным целой части числа $\log_2(n - 2\log_2 n)$:

$$m=[\log_2 (n - 2 \log_2 n)].$$

В этом случае

$$\frac{1}{2}(n - 2 \log_2 n)<2^m \leqslant n - 2 \log_2 n,$$

из чего следует, что

$$2^{2^m}+2^{n-m}<2^{n-2\log_2 n}+\frac{2^n}{\frac{1}{2}(n-2\log_2 n)}=$$
$$=\frac{2^{n+1}}{n}\left(\frac{1}{n}+\frac{1}{1-\frac{2\log_2 n}{n}}\right)=\frac{2^{n+1}}{n}(1+\varepsilon_n).$$

Здесь через ε_n обозначена величина, стремящаяся к нулю при $n\to\infty$. Используя последнее из выведенных соотношений, а также оценки, полученные в теореме 1.2, мы получим следующую асимптотическую оценку.

1.3. *Любая булева функция $f(x_1, \ldots, x_n)$ от n переменных $(n>4)$ может быть реализована как выходная функция комбинационной схемы, входными сигналами которой служат* $0, 1, x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$ *и которая содержит не более чем* $\frac{2^{n+2}}{n}(1+\varepsilon_n)$ *двухвходовых совпадений и не более чем* $\frac{2^{n+1}}{n}(1+\varepsilon_n)$ *двухвходовых разделений, причем величина* ε_n *стремится к нулю при* $n\to\infty$.

Нетрудно заметить, что оценка, определяемая условиями теоремы 1.3, останется справедливой и в том случае, если из числа входных сигналов будут исключены 0 и 1.

При решении канонической задачи синтеза в ряде случаев оказывается удобным использовать, кроме метода каскадов, также метод, основанный на совместной минимизации систем булевых функций. Имея в виду изложение этого метода, мы введем, прежде всего понятие *простой импликанты системы булевых функций.*

Простой импликантой множества S булевых функций $f_1, \ldots, f_m$ от переменных $x_1, \ldots, x_n$ называется всякое элементарное произведение p, составленное из этих переменных и их отрицаний, которое удовлетворяет следующему условию: произведение p является импликантой некоторых функций $f_{i_1}, \ldots, f_{i_t}$ множества S, а никакая собственная часть этого произведения уже не является импликантой хотя бы для одной из этих функций.

В силу приведенного определения, простые импликанты каждой из функций множества S являются в то же время простыми импликантами этого множества.

Из приведенного определения непосредственно вытекает также следующее, более общее предложение.

1.4. *Простые импликанты всевозможных произведений* $f_{i_1} f_{i_2} \dots f_{i_k}$ $(k \geqslant 1)$ *булевых функций из заданного множества* S *булевых функций, и только они, являются простыми импликантами множества* S.

Предложение 1.4 сводит задачу отыскания простых импликант любого (конечного) множества булевых функций к уже решенной в предыдущей главе задаче отыскания простых импликант одной булевой функции. По аналогии с § 1 предыдущей главы естественно систему простых импликант множества S булевых функций называть полной, если входящие в нее простые импликанты накрывают все единицы функций множества S. Полной будет очевидно, система всех простых импликант множества S.

Для целей синтеза комбинационных схем имеют особо важное значение приведенные системы простых импликант заданного множества S булевых функций, т. е. такие полные системы простых импликант, которые перестают быть полными при выбрасывании из них любой простой импликанты. Если построить комбинационную схему, выходные функции которой совпадают с функциями, составляющими некоторую приведенную систему простых импликант заданного множества S булевых функций, то каждая из функций множества S может быть получена из выходных функций этой схемы с помощью одних лишь разделений.

Комбинационную схему, у которой все совпадения предшествуют разделениям, мы будем называть *канонической двухступенчатой схемой*. Все совпадения образуют при этом первую, а все разделения — вторую ступень этой схемы. При построении канонических двухступенчатых схем на практике часто употребляют не двухвходовые, а многовходовые совпадения и разделения. Синтез таких схем производится в два этапа. На первом этапе выбирают ту или иную полную систему простых импликант множества выходных функций, которые требуется реализовать в синтезируемой схеме, и строят первую ступень схемы (на совпадениях), реализующую все импликанты выбранной системы. На втором этапе строят вторую ступень схемы,

производя необходимые разделения полученных простых импликант.

Нетрудно убедиться в том, что для получения канонических двухступенчатых схем с минимальным числом элементов необходимо пользоваться приведенными системами простых импликант. Как правило, число таких систем бывает достаточно большим, что затрудняет выбор наиболее экономной схемы с помощью простого перебора. Поэтому на практике ограничиваются обычно выбором одной из таких приведенных систем,— именно такой, в которой каждая из входящих в нее простых импликант накрывает наибольшее возможное число единиц (не накрытых другими импликантами) исходного множества S булевых функций.

В ряде случаев ограничиваются системой, состоящей из некоторых приведенных систем простых импликант каждой из функций множества S.

Ясно, что такая система будет непременно полной (хотя и не обязательно приведенной).

Рассмотрим действие описанных методов синтеза на примерах.

Пример 1. Построить комбинационную схему, имеющую в качестве своих выходных функций (частичные) булевы функции f_1 и f_2, задаваемые следующими картами Карнау:

Таблица V.1

x \ yu	00	01	11	10
0 0	—	1	1	1
0 1	1	0	0	—
1 1	1	0	0	—
1 0	—	—	1	0

Таблица V.2

x \ yu	00	01	11	10
0 0	0	1	—	1
0 1	1	1	1	1
1 1	0	0	—	0
1 0	0	1	—	0

Р е ш е н и е. Пользуясь картами Карнау, нетрудно найти все простые импликанты множества булевых функций (f_1, f_2). Мы выпишем лишь существенные простые импликанты, т. е. такие простые импликанты, которые накрывают хотя бы одну единицу (см. гл. IV, § 1):

$$p_1=\bar{z}\bar{u}, \quad p_2=\bar{x}\bar{y},$$
$$p_3=\bar{y}u, \quad p_4=y\bar{u},$$
$$p_5=\bar{x}\,\bar{z}\bar{u}, \quad p_6=\bar{x}y,$$
$$p_7=\bar{x}u, \quad p_8=zu,$$
$$p_9=\bar{x}\,z. \quad p_{10}=\bar{x}\bar{y}z,$$
$$p_{11}=\bar{x}\,y\bar{u}.$$

Импликанта p_1 накрывает две единицы (функции f_1), импликанта p_2 — три единицы (функции f_1), импликанта p_3 — пять единиц (три единицы функции f_1 и две единицы функции f_2), импликанта p_4 — две единицы (функции f_1), импликанта p_5 — три единицы (одну единицу функции f_1 и две единицы функции f_2), импликанта p_6 — четыре единицы (функции f_2), импликанта p_7 — три единицы (функции f_2), импликанта p_8 — одну единицу (функции f_2), импликанта p_9 — три единицы (функции f_2), импликанта p_{10} — три единицы, (две единицы функции f_1 и одну единицу функции f_2), импликанта p_{11} — три единицы (одну единицу функции f_1 и две единицы функции f_2).

В приведенную систему простых импликант естественно включить импликанты p_3 и p_6, накрывающие наибольшее число единиц (соответственно пять и четыре единицы). Остальные импликанты можно выбрать различными способами. Один из наиболее экономных способов приводит к системе, состоящей из простых импликант p_1, p_3, p_5, p_6. Исходные булевы функции представятся тогда в виде

$$f_1=p_1 \vee p_3 \vee p_5, \quad f_2=p_3 \vee p_5 \vee p_6.$$

Из этого приведенного представления следует, что вторая ступень канонической двухступенчатой схемы, имеющей функции f_1 и f_2 в качестве своих выходных функций, может быть построена из четырех двухвходовых

разделений. Поскольку

$$p_1=\bar{z}\bar{u},\quad p_3=\bar{y}u,\quad p_5=\bar{x}z\bar{u},\quad p_6=\bar{x}y,$$

то для построения первой ступени этой схемы достаточно, как нетрудно убедиться, пяти двухвходовых совпадений.

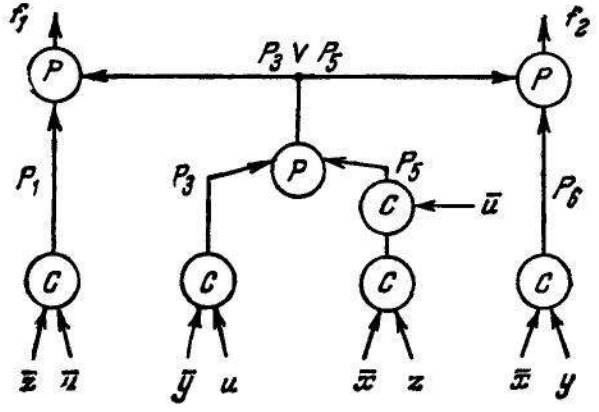

Рис. 11.

Соответствующая схема изображена на рис. 11 (P обозначает разделение, а C — совпадение).

Применение метода каскадов к функциям

$$f_1=\bar{z}\bar{u}\vee\bar{y}u\vee\bar{x}z\bar{u},$$
$$f_2=\bar{y}u\vee\bar{x}y\vee\bar{x}zu$$

приводит к появлению промежуточных функций

$$f_3(x,\ y,\ z)=f_1(x,\ y,\ z,\ 0)=\bar{z}\vee\bar{x},$$
$$f_4(x,\ y,\ z)=f_1(x,\ y,\ z,\ 1)=\bar{y},$$
$$f_5(x,\ y,\ z)=f_2(x,\ y,\ z,\ 0)=\bar{x}(y\vee z),$$
$$f_6(x,\ y,\ z)=f_2(x,\ y,\ z,\ 1)=\bar{x}\vee\bar{y},$$
$$f_7(x,\ y)\ \ =f_6(x,\ y,\ 0)=\bar{x}y.$$

Система уравнений, описывающая схему, построенную по методу каскадов, представится, очевидно, в виде:

$$f_7=\bar{x}y, \qquad f_6=\bar{x}\vee\bar{y},$$
$$f_5=f_7\vee\bar{x}z, \qquad f_4=\bar{y},$$
$$f_3=\bar{x}\vee\bar{z}, \qquad f_2=f_5\bar{u}\vee f_6u,$$
$$f_1=f_3\bar{u}\vee f_4u.$$

Для реализации соответствующей схемы достаточно, как легко проверить, пяти двухвходовых совпадений (для образования функции $\bar{x}y$, $\bar{x}z$, $f_5\bar{u}$, $f_3\bar{u}$, f_4u, f_6u) и пяти двухвходовых разделений. Таким образом, схема, построенная по методу каскадов, в данном случае оказалась менее экономной, чем ранее построенная каноническая двухступенчатая схема.

Пример 2. Построить комбинационную схему, имеющую в качестве своих выходных функций булевы функции

$$f_1=x+y+z=\bar{x}\bar{y}\bar{z}\vee\bar{x}yz\vee\bar{x}\bar{y}z\vee xyz,$$
$$f_2=xy\bar{z}\vee x\bar{y}z\vee\bar{x}yz\vee xyz=xy\vee xz\vee yz.$$

Решение. Простыми импликантами системы S булевых функций f_1 и f_2 будут все простые импликанты функции f_1 (т. е. $p_1=xy\bar{z}$, $p_2=x\bar{y}z$, $p_3=\bar{x}\bar{y}z$, $p_4=xyz$), все простые импликанты функции f_2 (т. е. $p_5=xy$, $p_6=xz$, $p_7=yz$) и все простые импликанты функции f_1f_2. Поскольку последняя функция равна, очевидно, xyz, то ее единственной простой импликантой является уже выписанная ранее импликанта p_4.

Система импликант p_1-p_7 является, как нетрудно видеть, приведенной. Соответствующая ей каноническая двухступенчатая схема может быть построена с помощью 10 двухвходовых совпадений и 5 двухвходовых разделений (одно совпадение экономится за счет того, что при образовании импликанты xyz попутно получается одна из импликант xy, xz, yz).

По методу каскадов получаем следующие промежуточные функции:

$$f_3(x, y)=f_1(x, y, 0)=x\bar{y}\vee\bar{x}y=(x\vee y)(\bar{x}\vee\bar{y}),$$
$$f_4(x, y)=f_1(x, y, 1)=\overline{x\bar{y}\vee\bar{x}y},$$
$$f_5(x, y)=f_2(x, y, 0)=xy,$$
$$f_6(x, y)=f_2(x, y, 1)=x\vee y$$

Так как

$$f_1=f_3\bar{z}\vee f_4 z, \quad f_2=f_5\bar{z}\vee f_6 z,$$

то отсюда следует возможность построения требуемой схемы с помощью 7 двухвходовых совпадений и 5 двухвходовых разделений. Соответствующая схема изображена

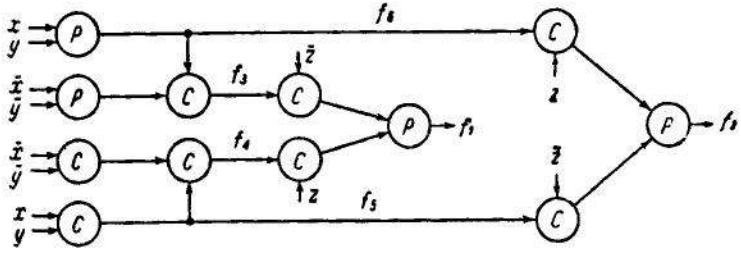

Рис. 12.

на рис. 12. Заметим, что в данном случае схема, построенная по методу каскадов, оказалась более экономной, чем каноническая двухступенчатая схема.

§ 2. Методы канонического синтеза некоторых специальных комбинационных схем

При синтезе цифровых автоматов часто приходится иметь дело с некоторыми специальными типами комбинационных схем. Одним из таких типов являются так называемые *дешифраторные схемы* или просто *дешифраторы*.

Дешифратором для n переменных $x_1, \ldots, x_n$ называется комбинационная схема, выходными функциями которой

служат различные конституенты единицы $\bar{x}_1\bar{x}_2\ldots\bar{x}_n$, $\bar{x}_1\bar{x}_2\ldots$ $\ldots\bar{x}_{n-1}\,x_n,\ldots,\ x_1x_2\ldots x_n$.

В настоящем параграфе мы будем рассматривать только *полные дешифраторы*, т. е. дешифраторы, реализующие все конституенты единицы (для данных переменных $x_1, \ldots, x_n$).

Существует три основных способа построения дешифраторных схем при условии использования в качестве входных сигналов переменных $x_1, x_2, \ldots, x_n$ и их отрицаний $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$. Мы будем называть эти способы соответственно *матричным*, *пирамидальным* и *прямоугольным*.

Матричный дешифратор представляет собою частный случай, описанный в предыдущем параграфе канонической двухступенчатой схемы. Вследствие того, что каждая из выходных функций имеет в этом случае единственную простую импликанту, вторая ступень в схеме отсутствует, а сама схема представляет собою простое объединение не связанных между собою схем, каждая из которых имеет в качестве своей выходной функции какую-либо конституенту единицы.

Поскольку конституента единицы от n переменных требует для своего построения $n-1$ двухвходовых совпадений, а общее число конституент равно 2^n, то мы приходим к следующему выводу.

2.1. *Полный матричный дешифратор для n переменных требует для своего построения* $(n-1)2^n$ *двухвходовых совпадений.*

Пирамидальный дешифратор строится по методу каскадов. Первый каскад пирамидального дешифратора состоит из четырех двухвходовых совпадений, образующих на выходах все четыре конституенты единицы:

$$\bar{x}_1\bar{x}_2,\quad \bar{x}_1x_2,\quad x_1\bar{x}_2,\quad x_1x_2$$

от двух переменных. Выходные сигналы первого каскада служат входными сигналами второго каскада, имеющего, кроме того, входные сигналы x_3 и $\bar{x}_3$. Второй каскад состоит из 8 двухвходовых совпадений, образующих на выходах все 8 конституент единицы от трех переменных. На следующем каскаде число совпадений снова

удваивается, а на его выходах образуются все конституенты единицы от четырех переменных.

Продолжая подобным образом, мы получим на выходах $(n-1)$-го каскада, построенного из 2^n двухвходовых совпадений, все конституенты единицы от n переменных. Поскольку $4+8+\ldots+2^n=2(2^n-2)$, то мы приходим к следующему выводу.

2.2. *Полный пирамидальный дешифратор требует для своего построения* $2(2^n-2)$ *двухвходовых совпадений.*

Прямоугольный способ построения дешифраторов осуществляет сведéние задачи построения дешифратора для n переменных к задаче построения двух дешифраторов для m и для $n-m$ переменных соответственно (где $m=1, 2, \ldots, n-1$). Для осуществления такого сведéния достаточно заметить, что любая конституента единицы от переменных $x_1, \ldots, x_n$ может быть представлена в виде произведения двух конституент единицы от переменных $x_1, \ldots, x_m$ и от переменных $x_{m+1}, \ldots, x_n$ соответственно. Чтобы получить таким способом все 2^n конституент единицы от n переменных, нужно иметь, очевидно, 2^n двухвходовых совпадений и два полных дешифратора для m и $n-m$ переменных соответственно. Сформулируем этот результат в виде следующей теоремы.

2.3. *Полный дешифратор для* n *переменных можно построить из* 2^n *двухвходовых совпадений и двух полных дешифраторов для* m *и для* $n-m$ *переменных соответственно. Здесь* m *— любое натуральное число, меньшее чем* n.

Теорема 2.3 дает возможность осуществлять последовательное сведéние задачи синтеза полного дешифратора к задаче синтеза полных дешифраторов для все меньшего и меньшего числа переменных. Замечая, что полный дешифратор для двух переменных может быть построен на четырех двухвходовых совпадениях, а полный дешифратор для одного переменного вообще не требует для своего построения никаких совпадений, мы — в соответствии с описанной методикой — получаем простой способ подсчета числа совпадений, необходимых для синтеза полного дешифратора в случае любого числа переменных.

Предположим, что нам необходимо синтезировать полный дешифратор для 7 переменных. Применяя теорему

2.3 и обозначив через C_n число двухвходовых совпадений в полном дешифраторе для n переменных, мы получим:

$$C_7=2^7+C_4+C_3,$$
$$C_4=2^4+C_2+C_2=16+4+4=24,$$
$$C_3=2^3+C_2+C_1=8+4+0=12.$$

Таким образом

$$C_7=2^7+24+12=164.$$

Разумеется, число совпадений, из которых строится дешифратор, зависит от способа, которым осуществляется сведéние задачи синтеза дешифратора к задачам синтеза дешифраторов для меньшего числа переменных. Если в рассмотренном только что примере вместо разбиения семерки на сумму 4+3 разбить ее на сумму 5+2, то число совпадений в синтезируемом дешифраторе окажется иным:

$$C_7=2^7+C_5+C_2=132+C_5,$$
$$C_5=2^5+C_3+C_2=\ 36+C_3,$$
$$C_3=2^3+C_2+C_1=12.$$

Отсюда будет следовать, что

$$C_7=132+36+12=180.$$

Нетрудно понять, что самым экономным (с точки зрения необходимого числа совпадений) является такой способ построения дешифратора, при котором число переменных в дешифраторе разбивается каждый раз на два равных слагаемых или на два слагаемых, отличающихся друг от друга на единицу. Именно такой способ построения дешифраторов мы будем называть в дальнейшем п р я м о у г о л ь н ы м. Ясно, что, применяя разбиения числа n переменных на слагаемые 1 и $n-1$, мы придем к описанному выше пирамидальному способу построения дешифраторов.

Для сравнения различных способов построения дешифраторов, описанных в настоящем параграфе, мы приведем таблицу (см. табл. V.3), показывающую зависимость числа двухвходовых совпадений, необходимых для построения

полного дешифратора, от типа дешифратора и от числа переменных.

Таблица V.3

Тип дешифратора \ Число переменных	1	2	3	4	5	6	7	8	9	10
Матричный	0	4	16	48	128	320	768	1792	4096	9216
Пирамидальный	0	4	12	28	60	124	252	508	1020	2044
Прямоугольный	0	4	12	24	48	88	164	304	584	1120

Рассмотрим теперь вопрос о синтезе комбинационных схем, выходными функциями которых служат так называемые симметричные булевы функции. Булева функция называется *симметричной*, если ее значения не меняются при любой перестановке ее аргументов. Непосредственно из этого определения вытекает следующее свойство симметричных булевых функций.

2.4. *Если совершенная д.н.ф. симметричной функции от n переменных содержит конституенту единицы, в которой ровно m переменных $(0 \leqslant m \leqslant n)$ входят с отрицаниями, то она содержит и все другие конституенты единицы, обладающие тем же самым свойством.*

При изучении симметричных булевых функций важную роль играют так называемые *базовые симметричные булевы функции*, то есть такие булевы функции, у которых все конституенты единицы, составляющие совершенную д.н.ф. данной функции, получаются из одной какой-нибудь конституенты с помощью применения к ней всевозможных перестановок переменных. Базовая симметричная функция однозначно определяется двумя параметрами, а именно числом переменных n и числом k переменных, входящих в какую-нибудь конституенту единицы совершенной д.н.ф. базовой функции в обычном (неинверсном) виде. Число k, очевидно, будет одним и тем же для всех конституент единицы, составляющих совершенную д.н.ф. базовой функции, и мы будем называть его *индексом* соответствующей функции.

Условимся через S_k^n обозначать базовую симметричную булеву функцию от n переменных, имеющую индекс k. Для любого данного числа переменных n имеется ровно $n+1$ базовых симметричных булевых функций S_0^n, $S_1^n, \ldots, S_n^n$. Например, для $n = 3$ имеется четыре базовых симметричных функции:

$$S_0^3 = \bar{x}\bar{y}\bar{z},$$
$$S_1^3 = \bar{x}\bar{y}z \vee \bar{x}y\bar{z} \vee x\bar{y}\bar{z},$$
$$S_2^3 = xy\bar{z} \vee x\bar{y}z \vee \bar{x}yz,$$
$$S_3^3 = xyz.$$

Из определения симметричных и базовых симметричных булевых функций немедленно вытекает справедливость следующего предложения.

2.5. *Любая симметричная булева функция представляется в виде дизъюнкции однозначно определяемых этой функцией базовых симметричных булевых функций.*

Действительно, если в совершенной д.н.ф. данной симметричной булевой функции f имеются лишь такие конституенты единицы, которые содержат k_1, k_2, ..., k_{m-1} или k_m неотрицаемых букв, то функция f совпадает, очевидно, с дизъюнкцией базовых симметричных функций, имеющих индексы k_1, k_2, ..., k_m [1]).

Теорема 2.5 позволяет ввести сокращенные обозначения симметричных булевых функций. Именно, условимся через $S^n_{k_1,k_2,\ldots,k_m}$ $(m \geqslant 1)$ обозначать симметричную булеву функцию от n переменных, представляющуюся в виде дизъюнкции базовых симметричных булевых функций $S^n_{k_1}, S^n_{k_2}, \ldots, S^n_{k_m}$. Легко видеть, что функция $S^n_{0,1,\ldots,n}$ тождественно равна единице. Тождественный нуль также принадлежит к числу симметричных булевых функций. Его естественно считать дизъюнкцией пустого множества базовых симметричных булевых функций и обозначать через $S^n\phi$ (ϕ — символ, обозначающий пустое множество). После этого дополнения введенная символика будет охватывать все симметричные булевы функции.

[1]) Тождественный нуль считается при этом дизъюнкцией пустого множества базовых симметричных функций.

Заметим еще, что в том случае, когда это не вызывает недоразумений,— в частности, в случае явного задания аргументов функции — верхний индекс n, обозначающий число переменных в символах симметричных булевых функций, обычно опускается.

Базовым симметричным многополюсником для n переменных $x_1, \ldots, x_n$ мы будем называть многополюсник, имеющий в качестве своих выходных функций все базовые симметричные функции от n переменных, а в качестве своих входных сигналов — сигналы $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$.

Справедливо следующее предложение.

2.6. *Базовый симметричный многополюсник для n переменных может быть построен (для любого $n = 1, 2 \ldots$) из $(n+2)(n-1)$ двухвходовых совпадений и $\frac{1}{2}n(n-1)$ двухвходовых разделений.*

Действительно, предложение 2.6 справедливо при $n = 1$, поскольку обе базовые симметричные функции от одной переменной (x и $\bar{x}$) могут быть реализованы без использования совпадений и разделений. Допустим теперь, что предложение 2.6 справедливо для значения $n = m$, и будем доказывать его справедливость при $n = m+1$. Нетрудно убедиться в справедливости следующих соотношений:

$$S_0^{m+1} = \bar{x}_{m+1} S_0^m,$$
$$S_k^{m+1} = \bar{x}_{m+1} S_k^m \vee x_{m+1} S_{k-1}^m \quad (k = 1, \ldots, m),$$
$$S_{m+1}^{m+1} = x_{m+1} S_m^m.$$

Эти соотношения показывают, что можно построить базовый симметричный многополюсник для $m+1$ переменных из базового симметричного многополюсника для m переменных, присоединяя к нему каскад, состоящий из $2m+2$ двухвходовых совпадений и из m двухвходовых разделений. Поскольку

$$(m+2)(m-1) + 2m + 2 = (m+3)m,$$
$$\frac{1}{2}m(m-1) + m = \frac{1}{2}(m+1)m,$$

то предложение 2.6 справедливо и для $n = m+1$. Тем самым доказательство этого предложения завершено.

Базовый симметричный многополюсник можно применить при построении схемы двоичного сумматора. В каждом разряде сумматора (за исключением самого младшего) имеется три входных переменных:

$$p_{k-1}, \quad x_k, \quad y_k$$

(перенос из предыдущего разряда и два одноименных разряда слагаемых). Задача состоит в построении функций z_k и p_k, то есть в построении соответствующего (k-го) разряда суммы и осуществлении переноса в следующий разряд. Для возможности построения схемы на одних совпадениях и разделениях необходимо образовать также функцию $\overline{p}_k$ (отрицание переноса).

Приступая к решению этой задачи, обратим внимание на то, что в младшем разряде перенос из предыдущего разряда отсутствует, а в старшем разряде отсутствует необходимость строить функции p_k и $\overline{p}_k$. Заметив это обстоятельство, выпишем следующие, легко проверяемые соотношения:

$$\begin{aligned}
&z_1 = S_1(x_1, y_1),\\
&p_1 = S_2(x_1, y_1),\\
&\overline{p}_1 = S_{0,1}(x_1, y_1),\\
&z_k = S_{1,3}(p_{k-1}, x_k, y_k) \qquad (k = 2, 3, \ldots, n)\\
&p_k = S_{2,3}(p_{k-1}, x_k, y_k) \qquad (k = 2, 3, \ldots, n-1),\\
&\overline{p}_k = S_{0,1}(p_{k-1}, x_k, y_k) \qquad (k = 2, 3, \ldots, n-1).
\end{aligned}$$

Учитывая эти соотношения и используя предложение 2.6, мы приходим к заключению, что первый разряд сумматора можно построить из четырех (двухвходовых) совпадений и двух (двухвходовых) разделений. На каждый из следующих $n-2$ разрядов потребуется по 10 совпадений и $\frac{3\cdot 2}{2}+3 = 6$ разделений. Последний (n-й) разряд требует 8 совпадений и 3 разделения.

Теперь нетрудно подсчитать, что n-разрядный двоичный сумматор (для сложения неотрицательных чисел) может быть построен из $10n-8$ двухвходовых совпадений и $6n-7$ двухвходовых разделений.

Заметим, что найденное решение не является самым экономным. Действительно, применяя метод каскадов, можно легко построить схему произвольного (k-го) разряда сумматора, используя лишь 9 двухвходовых совпадений и 6 двухвходовых разделений. Соответствующее построение может быть получено на базе примера 2 предыдущего параграфа, ибо, как нетрудно заметить, рассмотренные в этом примере функции f_1 и f_2 совпадают (с точностью до обозначений переменных) с рассмотренными выше функциями z_k и p_k. Для получения функции $\overline{p_k}$ к схеме, построенной в примере 2, достаточно присоединить два совпадения и одно разделение для реализации функции $a\bar{z} \vee b z$ (в которой функции $a = \bar{x} \vee \bar{y}$ и $b = \bar{x}\bar{y}$ уже были построены в схеме, рассмотренной в примере 2 и изображенной на рис. 12).

§ 3. Общие методы синтеза вентильных схем

Как уже отмечалось в гл. III, *вентилем* называется двухвходовой элемент, реализующий на выходе (логическое) произведение своих входных сигналов. При этом один из входных сигналов (называемый *управляющим*) обязательно должен совпадать с какой-либо из основных переменных $x_1, \ldots, x_n$ или их отрицаний $\bar{x}_1, \ldots, \bar{x}_n$, подаваемых на входные полюсы всей (состоящей из вентилей) схемы. Последнее обстоятельство делает естественным рассмотрение вентиля как устройства, осуществляющего или не осуществляющего передачу сигнала от своего входного узла к выходному, в зависимости от того, равно или не равно единице значение его управляющего входного сигнала.

В соответствии с этим на схемах принято обычно изображать вентиль как элемент с одним входом и одним выходом, а управляющий сигнал относить к характеристике самого вентиля, не указывая явно соответствующий ему входной канал. Тем самым оказывается возможным говорить о вентилях x_i, $\bar{x}_i$ ($i = 1, 2, \ldots$). Обычно используется обозначение вентилей с помощью кружочков, внутри которых проставляются обозначения их управляющих входных сигналов.

Напомним, что обычный (о д н о с т о р о н н и й) вентиль не может пропускать сигналов со своего выхода на вход. В отличие от него, так называемый д в у с т о р о н н и й вентиль может пропускать сигналы как в прямом направлении (от входа к выходу), так и в обратном направлении (от выхода к входу). Закон прохождения сигнала как в прямом, так и в обратном направлении совершенно одинаков: сигнал проходит через вентиль тогда и только тогда, когда значение сигнала, управляющего вентилем, равно единице. Вход и выход двустороннего вентиля оказываются, таким образом, совершенно равноправными, ввиду чего такой вентиль называют иногда также *симметричным*.

В соответствии со сказанным, мы будем при изучении вентильных схем различать два рода сигналов, а именно, *управляющие* и *вентильные* сигналы. Управляющие сигналы совпадают с основными переменными или их отрицаниями; они подаются на входы, управляющие вентилем. Вентильными сигналами называются сигналы, подающиеся на вторые входы вентилей, и сигналы, возникающие на выходах вентилей.

В электронных схемах управляющие сигналы являются обычно сигналами потенциального типа, а вентильные сигналы — сигналами импульсного типа. Мы будем считать в дальнейшем, что для вентильных сигналов имеет место так называемое е с т е с т в е н н о е р а з д е л е н и е (см. гл. III). Иными словами, при одновременном приходе в тот или иной узел схемы нескольких вентильных сигналов фактически возникающий в узле сигнал является дизъюнкцией всех этих сигналов. В и з о л и р о в а н н о м узле (то есть в узле, не подсоединенном ни к чему) возникает всегда нулевой вентильный сигнал.

При построении теории вентильных схем мы будем исключать из рассмотрения у п р а в л я ю щ и е у з л ы, то есть узлы, на которые подаются управляющие сигналы. Все узлы, о которых будет идти речь в этом параграфе, являются в е н т и л ь н ы м и узлами, то есть такими узлами, на которые передаются только вентильные сигналы. Обычная вентильная схема имеет один (в е н т и л ь н ы й) в х о д н о й п о л ю с, на который подается вентильный сигнал, равный (тождественно) единице. Вентиль-

ный сигнал, возникающий в любом (вентильном) узле вентильной схемы, представляет собой некоторую булеву функцию от управляющих сигналов этой схемы. Часть узлов схемы рассматривается в качестве ее в ы х о д н ы х (в е н т и л ь н ы х) п о л ю с о в. Возникающие в этих полюсах вентильные сигналы рассматриваются в качестве в ы х о д н ы х с и г н а л о в схемы, а соответствующие этим сигналам булевы функции (от управляющих сигналов) считаются в ы х о д н ы м и ф у н к ц и я м и схемы.

Все (вентильные) узлы вентильной схемы нумеруются натуральными числами, причем н о м е р е д и н и ц а присваивается обычно тому узлу, который отождествляется с (вентильным) входным полюсом схемы.

Зафиксировав некоторую (любую) пару (i,j) узлов вентильной схемы, рассмотрим множество всех вентилей, (вентильные) входы которых подсоединены к i-му узлу, а выходы — к j-му узлу. Пусть $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_m$ представляют собой управляющие сигналы всех вентилей этого множества ($\tilde{x}_l$, как обычно, обозначает либо x_l, либо $\bar{x}_l$). Булева функция $f_{ij} = \tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_m$ называется *функцией непосредственной связи* от i-го узла к j-му узлу. Функции f_{ii} мы будем, по определению, считать равными единице, а дизъюнкцию пустого множества термов принимать, как всегда, равной нулю.

Если n — число всех (вентильных) узлов схемы, то квадратная матрица n-го порядка $\|f_{ij}\|$ называется *матрицей непосредственных связей* этой схемы.

Если на i-й узел вентильной схемы подать вентильный сигнал, тождественно равный единице, то вентильный сигнал, возникающий на j-м узле схемы, будет некоторой булевой функцией от управляющих сигналов схемы. Эта функция называется *функцией полной связи* от i-го узла к j-му узлу и обозначается через F_{ij}. Квадратная матрица $\|F_{ij}\|$, составленная из функций полной связи между всеми узлами схемы, называется *матрицей полных связей* этой схемы.

Для схем, составленных из двусторонних (симметричных) вентилей, как матрица непосредственных связей, так и матрицы полных связей будут, очевидно, симмет-

ричными. Иными словами, для любых узлов i и j справедливо, что

$$f_{ij} = f_{ji}, \quad F_{ij} = F_{ji}.$$

Для булевых матриц, т. е. для квадратных матриц, элементами которых служат булевы функции, естественным образом можно определить операцию умножения и возведения в степень. Если $A = \|\alpha_{ij}\|$ и $B = \|\beta_{ij}\|$ являются двумя квадратными булевыми матрицами одного и того же порядка n, то их произведением $C = \|\gamma_{ij}\|$ является квадратная булева матрица порядка n, произвольный элемент которой определяется по формуле

$$\gamma_{ij} = \alpha_{i1}\beta_{1j} \vee \alpha_{i2}\beta_{2j} \vee \ldots \vee \alpha_{in}\beta_{nj}.$$

Теперь нетрудно определить степени произвольной квадратной булевой матрицы A:

$$A^2 = AA, \quad A^3 = A^2A, \ \ldots, \quad A^n = A^{n-1}A, \ \ldots$$

Справедливо следующее предложение, принадлежащее А. Г. Лунцу [1]).

3. 1. *Для любой вентильной схемы с n вентильными узлами и $(n-1)$-я степень (а также любая бо́льшая степень) матрицы непосредственных связей этой схемы совпадает с матрицей ее полных связей.*

Доказательство теоремы 3.1 мы начнем с рассмотрения произвольной пары узлов (i, j) заданной вентильной схемы. Сигнал, поданный на i-й узел, может быть передан на j-й узел непосредственно в том случае, если имеется вентиль, входной канал которого подсоединен к i-му, а выходной канал — к j-му узлу схемы. Возможна, однако, и такая передача сигнала от i-го к j-му узлу, при которой этот сигнал проходит через некоторые другие узлы. Ясно, что сигнал, фактически передающийся от i-го к j-му узлу, будет дизъюнкцией всех таких сигналов.

Рассматривая вместо самих сигналов соответствующие им булевы функции, мы приходим к выводу, что функция полной связи F_{ij} от i-го узла к j-му узлу представима

---

[1]) А. Г. Лунц, Приложение матричной булевой алгебры к анализу и синтезу релейно-контактных схем, ДАН СССР, т. 70, № 3, 1950, стр 421—423.

в виде дизъюнкций всевозможных произведений функций непосредственных связей f_{ij}, имеющих вид

$$p = f_{ik_1} f_{k_1 k_2} f_{k_2 k_3} \dots f_{k_{m-1} k_m} f_{k_m j} \qquad (m \geqslant 0).$$

Нетрудно убедиться в том, что можно ограничиться при этом только такими произведениями, которые содержат не более чем $n-1$ сомножителей. Действительно, если произведение содержит n или более сомножителей, то $m \geqslant n-1$, и из $n+1$ индексов $i, k_1, \geqslant \dots, k_{n-1}, j$ хотя бы два индекса будут одинаковыми. Благодаря принятому способу индексации в произведении p найдется в этом случае группа стоящих рядом сомножителей такая, что первый индекс первого из сомножителей группы совпадает со вторым индексом последнего из сомножителей этой же группы (не исключено, что оба эти сомножителя совпадают между собой).

Ясно, что, исключив из произведения p указанную группу сомножителей, мы придем к произведению того же самого типа, что и произведение p, но содержащему меньшее число сомножителей. Если это число превышает $n-1$, то процесс исключения сомножителей можно повторять до тех пор, пока мы не придем к произведению q, длина которого (то есть число его сомножителей) окажется не превосходящей $n-1$. Поскольку произведение q очевидным образом поглощает произведение p (ибо $p \vee q = q$), то в описанной выше дизъюнкции и в самом деле можно ограничиться лишь теми произведениями, длина которых меньше или равна $n-1$.

Поскольку функция f_{ii} тождественно равна единице, то любое произведение $p = f_{ik_1} \dots f_{k_m j}$, имеющее меньше, чем $n-1$ сомножителей, можно заменить равным ему произведением $f_{ii} \dots f_{ii} p$, имеющим ровно $n-1$ сомножителей. Нетрудно доказать посредством индукции, что элементами α_{ij} матрицы A^m, представляющей собою m-ую степень матрицы $A = \|f_{ij}\|$ непосредственных связей любой вентильной схемы, будут всевозможные произведения $f_{ii_2} f_{i_2 i_3} \dots f_{i_m j}$, состоящие из m сомножителей. В силу всего сказанного мы делаем вывод, что матрица A^{n-1} совпадает с матрицей полных связей рассматриваемой вентильной схемы. Поскольку те же самые рассуждения можно повторить для любого показателя $m \geqslant n-1$, теорема 3.1 полностью доказана.

Доказанная теорема позволяет относительно простым способом — с помощью возведения в степень булевых матриц — производить анализ вентильных схем, то есть определять выходные функции этих схем. Действительно, матрица непосредственных связей вентильной схемы является непосредственным отражением структуры этой схемы и может быть найдена по заданной схеме тривиальным образом. Возводя матрицу непосредственных связей в степень $n-1$ (n есть порядок матрицы), мы в числе элементов строки матрицы, полученной в результате возведения в степень, получим все выходные функции схемы.

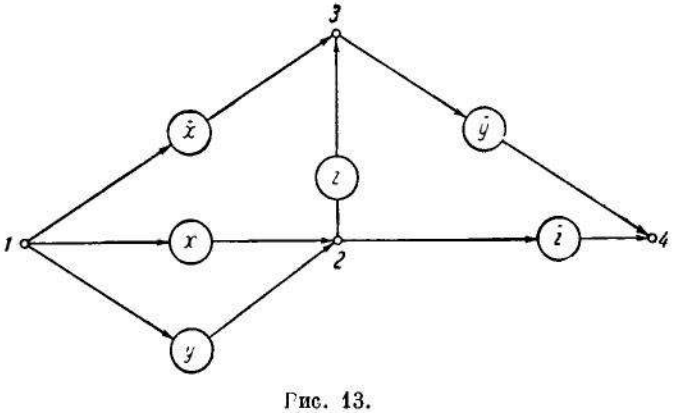

Рис. 13.

Рассмотрим в качестве примера вентильную схему, состоящую из односторонних вентилей, которая изображена на рис. 13.

Из этого рисунка находим матрицу непосредственных связей схемы:

$$A=\begin{Vmatrix} 1 & x\vee y & \bar{x} & 0 \\ 0 & 1 & z & \bar{z} \\ 0 & 0 & 1 & \bar{y} \\ 0 & 0 & 0 & 1 \end{Vmatrix}.$$

Для решения задачи анализа в рассматриваемом случае достаточно найти третью степень матрицы A. Осуществляя последовательное умножение матрицы A на

самое себя, находим:

$$A^2=\begin{Vmatrix}1 & x\vee y & a & b\\ 0 & 1 & z & c\\ 0 & 0 & 1 & \bar{y}\\ 0 & 0 & 0 & 1\end{Vmatrix},$$

где

$$a=\bar{x}\vee(x\vee y)z=\bar{x}\vee z,$$
$$b=(x\vee y)\bar{z}\vee\bar{x}\bar{y}=\bar{z}\vee\bar{x}\bar{y},$$
$$c=\bar{z}\vee\bar{y}z=\bar{z}\vee\bar{y}.$$

$$A^3=A^2\cdot A=\begin{Vmatrix}1 & x\vee y & d & e\\ 0 & 1 & z & f\\ 0 & 0 & 1 & \bar{y}\\ 0 & 0 & 0 & 1\end{Vmatrix},$$

где

$$d=\bar{x}\vee(x\vee y)z\vee a=\bar{x}\vee z,$$
$$e=(x\vee y)\bar{z}\vee a\bar{y}\vee b=\bar{y}\vee\bar{z},$$
$$f=\bar{z}\vee z\bar{y}\vee c=\bar{y}\vee\bar{z}.$$

Если входным полюсом схемы является узел *1*, а выходными полюсами — узлы *3* и *4*, то выходными функциями схемы являются два последних элемента первой строки матрицы полных связей A^3:

$$F_{13}=d=\bar{x}\vee z,\qquad F_{14}=e=\bar{y}\vee\bar{z}.$$

Нетрудно показать, что эти выходные функции можно реализовать с помощью гораздо более простой вентильной схемы, чем схема, изображенная на рис. 13 (сложность вентильной схемы принято оценивать числом входящих в нее вентилей).

Теорема 3. 1 позволяет также осуществить некоторый специальный подход к проблеме синтеза вентильных схем. Действительно, задача синтеза вентильной схемы, имеющей заданные выходные функции $F_1, F_2, \ldots, F_k$ от переменных $x_1, \ldots, x_m$, сводится с помощью теоремы 3.1 к следующей задаче.

Требуется найти булеву матрицу $A=\|f_{ij}\|$ *порядка* $n\geqslant k+1$, *все диагональные элементы которой* (f_{ii}) *равны единице, все остальные ее элементы представляют собою произвольные элементарные дизъюнкции (дизъюнкции пе-*

ременных $x_1, \ldots, x_n$ и их отрицаний), в том числе тождественный нуль, отличающуюся тем, что при возведении этой матрицы в $(n-1)$-ю степень некоторые элементы первой строки матрицы A^{n-1} совпадают с заданными функциями $F_1, F_2, \ldots, F_k$.

Непосредственное решение этой задачи представляет собою значительные трудности. Более удобно поэтому воспользоваться методом *последовательного восстановления узлов*, предложенным Ф. Хоном и Шислером [1]). Для того чтобы лучше понять этот метод, рассмотрим предварительно задачу *последовательного исключения узлов* в вентильных схемах.

Пусть нам дана произвольная вентильная схема, заданная с помощью своей матрицы непосредственных связей $A = \|f_{ij}\|$, и пусть n есть порядок этой матрицы (число узлов схемы). Поставим задачу исключить из схемы какой-либо узел, например m-й, и найти матрицу $B = \|g_{ij}\|$ $(i, j = 1, 2, \ldots, m-1, m+1, \ldots, n)$ непосредственных связей схемы с исключенным m-м узлом.

Произвольный элемент g_{ij} этой матрицы представляет собой булеву функцию, выражающую сигнал в j-м узле при условии, что в i-й узел подается сигнал, тождественно равный единице, а через все остальные узлы, кроме i-го, j-го и исключенного m-го узла, передача информации не производится (этого можно достичь, производя отключение входных и выходных каналов вентилей от всех таких узлов).

Непосредственно из этого определения вытекает соотношение

$$g_{ij} = f_{ij} \vee f_{im} f_{mj}, \qquad (1)$$

решающее задачу исключения из схемы m-го узла.

По аналогии с задачей исключения одного узла вентильной схемы можно поставить задачу исключения произвольного множества узлов $M = (m_1, m_2, \ldots, m_k)$. *Функцией непосредственной* связи i-го узла с j-м узлом вентильной схемы с исключенным множеством M узлов мы будем

[1]) F. E. Hohn, L. R. Schissler, Boolean matrices and the design of combinational relay switching circuits. Bell. System Tecb., v. 34, 1955, p. 177—202.

называть булеву функцию, выражающую сигнал в j-м узле при условии, что в i-й узел подается сигнал, тождественно равный единице, а от всех узлов, кроме i-го, j-го и узлов множества M, отсоединены входные и выходные каналы всех подсоединенных к ним вентилей.

Матрица $\|h_{ij}\|$ порядка $n-k$, где индексы i и j принимают любые значения от единицы до n, не входящие в множество M, называется *матрицей непосредственных связей* рассматриваемой схемы с *исключенным множеством узлов* M.

Нетрудно убедиться в том, что формула (1) справедлива и для того случая, когда $\|f_{ij}\|$ представляет собой матрицу непосредственных связей вентильной схемы с исключенным множеством M узлов, а $\|g_{ij}\|$ является матрицей непосредственных связей той же схемы, из которой исключен, кроме узлов множества M, еще и m-й узел.

Сделанное замечание позволяет использовать формулу (1) для решения задачи анализа вентильных схем. Действительно, после (последовательного) исключения всех узлов любой данной вентильной схемы, кроме какой-либо пары узлов i и j, возникающая матрица (второго порядка) непосредственных связей полученной схемы с исключенными узлами будет, очевидно, иметь в качестве своих недиагональных элементов функции полной связи от i-го узла к j-му и от j-го узла к i-му.

В качестве примера рассмотрим применение метода исключения узлов к задаче нахождения функции полной связи F_{14} от 1-го узла к 4-му в вентильной схеме, изображенной на рис. 13. Матрица непосредственных связей этой схемы имеет вид

$$A=\begin{Vmatrix} 1 & x\vee y & \bar{x} & 0 \\ 0 & 1 & z & \bar{z} \\ 0 & 0 & 1 & \bar{y} \\ 0 & 0 & 0 & 1 \end{Vmatrix}.$$

После исключения второго узла мы приходим к матрице:

$$A_1=\begin{Vmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{Vmatrix},$$

где

$$a = \bar{x} \vee (x \vee y) z = \bar{x} \vee z,$$
$$b = \bar{z} (x \vee y),$$
$$c = \bar{y}.$$

После исключения третьего (в матрице A_1 — второго) столбца мы приходим к матрице:

$$A_2 = \begin{Vmatrix} 1 & b \vee ac \\ 0 & 1 \end{Vmatrix}.$$

Таким образом, искомая функция F_{14} равна

$$b \vee ac = \bar{z} (x \vee y) \vee \bar{x}\bar{y} \vee \bar{y}z = \bar{y} \vee \bar{z}.$$

При исключении узлов на каждом этапе такого исключения осуществляется переход от некоторой булевой матрицы A к булевой матрице B, имеющей порядок на единицу ниже. Нетрудно убедиться в том, что этот переход осуществляется на основе следующего правила.

3.2. *Для получения матрицы B в матрице A вычеркивается строка и столбец, соответствующие исключаемому узлу, а к любому из остающихся элементов f_{ij} добавляется новый дизъюнктивный член, равный произведению элемента исключаемой строки, стоящего в одном столбце с элементом f_{ij}, и элемента исключаемого столбца, стоящего в одной строке с элементом f_{ij}.*

Операция исключения (строки и столбца), описываемая правилом 3.2, применима к любой булевой матрице. Естественно определить обратную ей операцию, которую мы будем называть *операцией расширения*. Операция расширения заключается в окаймлении данной булевой матрицы A новой строкой и новым столбцом так, чтобы после их исключения (по правилу 3.2) получалась снова исходная матрица A.

Операция расширения неоднозначна. Так, например, в результате расширения матрицы

$$\begin{Vmatrix} 1 & xy \vee zu \\ 0 & 1 \end{Vmatrix}$$

могут быть получены существенно различные матрицы:

$$\begin{Vmatrix} 1 & xy & z \\ 0 & 1 & 0 \\ 0 & u & 1 \end{Vmatrix}, \quad \begin{Vmatrix} 1 & zu & x \\ 0 & 1 & 0 \\ 0 & y & 1 \end{Vmatrix}.$$

Применяя к произвольной булевой матрице A последовательно несколько раз операцию расширения, мы всегда можем привести эту матрицу к *элементарно дизъюнктивной форме*, то есть, иными словами, превратить ее в такую матрицу, элементами которой являются элементарные дизъюнкции (включая константу н у л ь) и константа единица.

В самом деле, запишем каждый элемент f_{ij} матрицы в конъюнктивной нормальной форме. Те из элементов, для которых конъюнктивная нормальная форма состоит более чем из одного сомножителя, разобьем на два сомножителя: $f_{ij}=g_{ij}h_{ij}$, каждый из которых представляется конъюнктивной нормальной формой, имеющей меньшее число сомножителей, чем элемент (булева функция) f_{ij}. Теперь нетрудно убедиться в том, что можно произвести расширение матрицы A, заменяя элемент f_{ij} нулем и вводя новую строку, имеющую на i-м месте элемент g_{ij} и новый столбец, имеющий на i-м месте элемент h_{ij}. Все остальные элементы добавляемой строки и добавляемого столбца, за исключением элемента, стоящего на пересечении добавляемой строки и добавляемого столбца, принимаются равными нулю; что же касается этого последнего элемента, то его мы примем равным единице.

Применяя описанное построение для всех элементов матрицы A, мы расширим эту матрицу до матрицы B, у которой максимальное число сомножителей в к. н. ф. для всех элементов матрицы будет меньше соответствующего числа для матрицы A. Если это число (для матрицы B) не равно единице, процесс повторяется; повторение происходит до тех пор, пока все элементы матрицы не превратятся либо в элементарные дизъюнкции (включая н у л ь), либо в константу е д и н и ц а.

Введем следующее определение.

3. 3. *Квадратная булева матрица называется правильной, если все элементы ее главной диагонали равны единице.*

Правильная квадратная булева матрица называется вентильной, если все ее элементы, не стоящие на главной диагонали, являются элементарными дизъюнкциями (включая константу нуль).

Ясно, что всякая вентильная матрица может рассматриваться как матрица непосредственных связей некоторой вентильной схемы, и наоборот.

Проведенное выше построение обеспечивает при каждом расширении получение из правильной матрицы опять-таки правильной матрицы; но тогда ясно, что на заключительном этапе этого построения всякий раз будет получаться вентильная матрица, как только исходная матрица оказывается правильной. Тем самым нами доказано следующее предложение.

3. 4. *В результате конечного числа расширений любая правильная квадратная булева матрица может быть превращена в вентильную матрицу.*

Теорема 3.4 дает возможность построить упомянутый выше алгоритм синтеза вентильных схем методом последовательного восстановления узлов. Первым шагом в этом алгоритме является нахождение матрицы непосредственных связей искомой схемы, из которой исключены все узлы, кроме входного и выходного полюсов. Такая матрица (которую мы обозначим буквой A) может быть построена многими различными способами. Наиболее простой способ состоит в следующем.

Если $f_1, \ldots, f_k$ представляют собой выходные функции, которые должна реализовать искомая схема, то в качестве матрицы A можно выбрать матрицу:

$$\begin{Vmatrix} 1 & f_1 & f_2 & \ldots & f_k \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 1 \end{Vmatrix}.$$

Легко видеть, что эта матрица обладает тем свойством, что при применении к ней операции исключения узлов элементы ее первой строки не меняются, так что в результате исключения всех узлов, кроме входного полюса и i-го

выходного полюса, она превращается в матрицу $\left\|\begin{matrix}1 & f_i\\ 0 & 1\end{matrix}\right\|$. Это означает, очевидно, что схема, соответствующая матрице A, имеет в качестве выходных функций заданные функции $f_1, \ldots, f_k$.

Построенная нами матрица A является правильной квадратной булевой матрицей. Применяя к ней несколько раз операцию расширения, мы на основании теоремы 3.4 через конечное число шагов придем к вентильной матрице B. Принимая матрицу B в качестве матрицы непосредственных связей некоторой вентильной схемы C, мы приходим к выводу, что эта схема имеет своими выходными функциями заданные функции $f_1, \ldots, f_k$. Действительно, в силу описанного построения, оказывается возможным с помощью последовательного исключения всех узлов, кроме входного полюса и любого выходного полюса i, привести эту матрицу к виду $\left\|\begin{matrix}1 & f_i\\ 0 & 1\end{matrix}\right\|$. На основании описанного выше алгоритма анализа это означает, что функция f_i является i-й выходной функцией схемы C.

Процесс нахождения матрицы B с помощью последовательных расширений матрицы A представляет собою второй (заключительный) шаг алгоритма Хона для синтеза вентильных схем.

Легко видеть теперь, что на первом шаге алгоритма в качестве матрицы A можно выбрать любую устойчивую булеву матрицу $(k+1)$-го порядка, т. е. такую правильную квадратную булеву матрицу, первая строка которой состоит из единицы и заданных выходных булевых функций $f_1, \ldots, f_k$ и не меняется при возведении матрицы A в квадрат.

Действительно, если какая-либо строка матрицы не изменилась при возведении матрицы в квадрат, то она сохранится неизменной в более высоких степенях этой матрицы. Но тогда, в силу теоремы 3.1, функции $f_1, f_2, \ldots, f_k$ будут выходными функциями схемы, задаваемой матрицей A.

Продемонстрируем работу описанного алгоритма на примере. Предположим, что нам необходимо построить вентильную схему (из односторонних вентилей), имеющую в качестве своих выходных функций две булевы

функции:

$$f_1 = xy \vee \bar{x}\bar{z},$$
$$f_2 = x\bar{z} \vee \bar{x}y.$$

В соответствии с описанным выше алгоритмом, на первом шаге выпишем какую-нибудь устойчивую булеву матрицу, первая строка которой совпадает со строкой 1, f_1, f_2, например матрицу

$$A = \begin{Vmatrix} 1 & f_1 & f_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{Vmatrix} = \begin{Vmatrix} 1 & xy \vee \bar{x}\bar{z} & x\bar{z} \vee \bar{x}y \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{Vmatrix}.$$

Применяя к матрице A процесс расширения (восстановления узлов), мы приведем ее сначала к виду

$$A_1 = \begin{Vmatrix} 1 & \bar{x}\bar{z} & x\bar{y} & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & y & \bar{z} & 1 \end{Vmatrix},$$

а затем к виду

$$A_2 = \begin{Vmatrix} 1 & 0 & 0 & x & \bar{x} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & y & \bar{z} & 1 & 0 \\ 0 & \bar{z} & y & 0 & 1 \end{Vmatrix}.$$

Матрица A_2 является вентильной. Ей соответствует вентильная схема, изображенная на рис. 14. Символом 0 на этой схеме обозначен входной полюс, а символами *1* и *2* — выходные полюсы, реализующие соответственно функции f_1 и f_2.

Отметим, что в процессе расширения матриц A и A_1 мы осуществляли каждый раз одновременное расщепление двух произведений, что было возможно благодаря наличию у этих произведений общего множителя (в первом случае x, а во втором $\bar{x}$). Ясно, что такое одновременное расщепление приводит к более экономному расходованию вентилей.

При анализе и синтезе схем из симметричных вентилей оказываются применимыми те же самые приемы, что и в случае односторонних вентилей. Необходимо лишь всякий раз дополнительно заботиться о том, чтобы все получающиеся матрицы были симметричными. В случае алгоритма анализа (исключения узлов) из симметричной матрицы могут возникать только симметричные матрицы. Что же

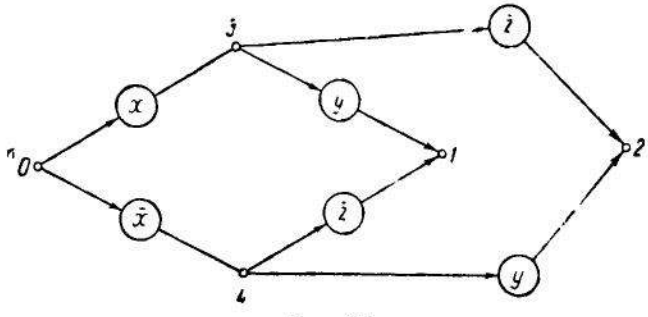

Рис. 14.

касается алгоритма синтеза, то сохранение свойства симметричности матриц в этом случае должно контролироваться на каждом шаге алгоритма. Заметим, что самой простой устойчивой симметричной булевой матрицей, имеющей заданную первую строку $1, f_1, \ldots, f_k$, является матрица

$$\begin{Vmatrix} 1 & f_1 & f_2 & \ldots & f_k \\ f_1 & 1 & 0 & \ldots & 0 \\ f_2 & 0 & 1 & \ldots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ f_k & 0 & 0 & \ldots & 1 \end{Vmatrix}.$$

Применим описанный алгоритм синтеза к синтезу схемы из двухсторонних вентилей, реализующей выходные функции $f_1 = xy \vee z$ и $f_2 = xz \vee y$.

В качестве исходной устойчивой матрицы выберем матрицу

$$A = \begin{Vmatrix} 1 & xy \vee z & xz \vee y \\ xy \vee z & 1 & 0 \\ xz \vee y & 0 & 1 \end{Vmatrix}.$$

Применяя к ней процесс расширения (с учетом необходимости получения симметричной матрицы) мы придем к матрице

$$B=\begin{Vmatrix} 1 & z & y & x \\ & 1 & 0 & y \\ y & 0 & 1 & z \\ x & y & z & 1 \end{Vmatrix}.$$

Матрица B — вентильная. Ей соответствует вентильная схема, изображенная на рис. 15.

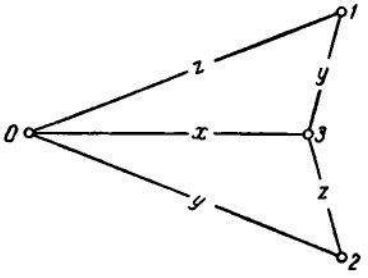

Рис 15.

Символом 0 на этом рисунке обозначен входной полюс, а символами *1* и *2* — выходные полюсы, реализующие соответственно функции f_1 и f_2. Заметим также, что на рис. 15 двусторонние вентили обозначены так, как принято обычно обозначать релейные контакты. Как уже отмечалось выше, электромагнитные реле представляют собой наиболее употребительный вид двусторонних вентилей. Поэтому такое обозначение является вполне оправданным.

§ 4. Некоторые дополнительные приемы синтеза и минимизации вентильных схем

При синтезе вентильных схем оказывается применимым, очевидно, метод каскадов, описанный в § 1 настоящей главы. Возможность его применения обусловливается тем обстоятельством, что в каждом каскаде схемы, построенной с помощью этого метода, применяются лишь такие совпадения, которые осуществляют умножение выходных сигналов предыдущего каскада на сигналы, соответствующие основным переменным или их отрицаниям. Такие совпадения могут быть очевидным образом реализованы с помощью вентилей. Что же касается разделений, то в вентильных схемах необходимость в них отпадает ввиду наличия естественного разделения сигналов.

Таким образом, возникает возможность интерпретации всех результатов, полученных в § 1 настоящей главы, на

языке вентильных схем. Для осуществления такой интерпретации достаточно заменить каждое двухвходовое совпадение соответствующим (односторонним) вентилем, а каждое разделение — простым соединением входных каналов этого разделения в одном узле схемы. Различие получается лишь в первом каскаде: в канонической задаче синтеза (с помощью двухвходовых совпадений и разделений) в первом каскаде схемы реализуются функции двух переменных, например, функции x_1x_2 и $\bar{x}_1 \vee \bar{x}_2$. В случае вентильной интерпретации этот каскад более естественно разбить на два каскада: в первом образуются в виде вентильных сигналов необходимые функции одной переменной (в данном случае функции x_1 и $\bar{x}_1$), во втором, с помощью вентилей x_2 и $\bar{x}_2$, из этих функций образуются необходимые функции от двух переменных (в данном случае x_1x_2 и $\bar{x}_1 \vee \bar{x}_2$).

В случае двусторонних вентилей возникает дополнительная трудность, связанная с возможностью обратной передачи сигналов из последующих каскадов в предыдущие. Необходимо поэтому всякий раз проверять, не приведет ли такая передача к искажению окончательных выходных сигналов схемы.

При построении *универсального вентильного многополюсника* для n переменных (то есть вентильной схемы, имеющей в качестве выходных функций все булевы функции от n переменных) переход от i-го каскада к $(i+1)$-му выполняется точно так же, как и в § 1: в $(i+1)$-м каскаде требуется не более $2 \cdot 2^{2^{i+1}} - 2^{2^i}$ вентилей. Ясно также, что первый каскад требует всего двух вентилей: x_1 и $\bar{x}_1$. Так как $2 = 2 \cdot 2^{2^1} - 6$, то, рассуждая так же, как и в § 1, мы придем к следующему предложению.

4. 1. *Универсальный вентильный многополюсник для n переменных может быть построен с помощью не более чем $2 \cdot 2^{2^n} - 6$ вентилей.*

Заметим, что предложение 4. 1 остается справедливым не только для односторонних, но также и для двусторонних вентилей.

Точно так же, как и в § 1, устанавливается асимптотическая оценка Шеннона.

4. 2. *Любая булева функция от n переменных может быть реализована в качестве выходной функции вентильной*

схемы, состоящей из $\frac{2^{n+2}}{n}(1+\varepsilon_n)$ *вентилей;* ε_n *обозначает здесь положительную величину, стремящуюся к нулю при* $n \to \infty$.

Схема, о которой идет речь в теореме 4.2, получается с помощью объединения универсального вентильного многополюсника для $m (m < n)$ переменных с вентильным (обратным) деревом для $n-m$ переменных, строящимся по аналогии с соответствующей схемой для случая канонической задачи синтеза. Теорема 4. 2 также справедлива не только для односторонних, но и для двусторонних вентилей.

Нетрудно убедиться в том, что с помощью вентилей нельзя реализовать прямоугольный способ построения дешифраторных схем, поскольку при этом необходимо было бы осуществлять совпадение двух сигналов, получающихся на выходах вентилей. Что же касается других способов построения дешифраторов (матричного и пирамидального), то оба они могут применяться и в случае вентильных схем.

При матричном способе каждая конституента единицы строится отдельно и в случае дешифраторов для n переменных требует, очевидно, n вентилей. Поскольку общее число конституент равно 2^n, то для построения вентильного дешифратора от n переменных матричным способом требуется $n2^n$ вентилей.

При пирамидальном способе в первом каскаде схемы строятся конституенты единицы x_1 и $\bar{x}_1$ для одной переменной, во втором каскаде — для двух переменных, и т. д. Число вентилей, используемых в каждом каскаде, равняется при этом числу выходных каналов этого каскада. Таким образом, общее число вентилей, которые требуются для построения дешифратора от n переменных пирамидальным способом, равно $2^1+2^2+2^3+\ldots+2^n=2^{n+1}-2$. В i-м каскаде каждый выходной сигнал предыдущего каскада в этой схеме направляется на два вентиля x_i и $\bar{x}_i$, в результате чего из каждой конституенты единицы от $i-1$ переменных в i-м каскаде образуются две конституенты единицы от i переменных.

Заметим, наконец, что с помощью вентилей нетрудно построить базовый симметричный многополюсник.

Произвольный (i-й) каскад этого многополюсника строится с помощью $2i$ вентилей и имеет $i+1$ выходных функций, которые мы обозначим через $f_0^i, f_1^i, \ldots, f_i^i$. Связи между каскадами задаются следующими уравнениями:

$$\begin{aligned}
f_0^i &= x_i f_0^{i-1}, \\
f_1^i &= \bar{x}_i f_0^{i-1} \vee x_i f_1^{i-1}, \\
f_2^i &= \bar{x}_i f_1^{i-1} \vee x_i f_2^{i-1}, \\
&\ldots\ldots\ldots\ldots \\
f_{i-1}^i &= \bar{x}_i f_{i-2}^{i-1} \vee x_i f_{i-1}^{i-1}, \\
f_i^i &= \bar{x}_i f_{i-1}^{i-1}.
\end{aligned}$$

Поскольку $\sum_{i=1}^{n} 2i = n(n+1)$, то мы приходим к следующему предложению.

4.3. *Для любого натурального числа n можно построить базовый симметричный многополюсник от n переменных в виде вентильной схемы, насчитывающей $n(n+1)$ вентилей.*

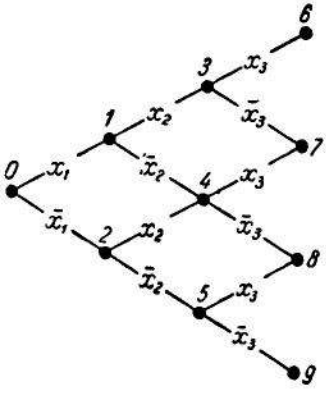

Рис. 16.

На рис. 16 изображен вентильный базовый симметричный многополюсник для случая трех переменных. Из рассмотрения этой схемы нетрудно усмотреть, что сигнал, попавший в какой-либо выходной канал i-го каскада многополюсника, не может попасть отсюда в другой выходной канал того же каскада через предыдущие каскады. Действительно, для этого сигнал должен обязательно пройти последовательно через пару вентилей x_l, $\bar{x}_l$, что невозможно, так как когда один из этих вентилей открыт (пропускает сигнал), другой обязательно закрыт и, следовательно, пропустить сигнал, прошедший через первый вентиль, не может.

Приведенные соображения показывают, что базовые симметричные многополюсники можно строить абсолютно

одинаковым способом как на односторонних, так и на двусторонних вентилях. То же самое относится, как нетрудно проверить, также и к дешифраторным схемам.

Рассмотрим теперь некоторые способы, позволяющие упрощать уже построенные вентильные схемы. В ряде случаев достаточно хорошие результаты дает следующий частный прием упрощения матриц непосредственных связей вентильных схем. Введем в рассмотрение три узла схемы i, j, k и функции непосредственных связей f_{ij}, f_{jk}, f_{ik} между этими узлами. При исключении j-го узла функция f_{ik} заменится функцией $f_{ik} \vee f_{ij} f_{jk}$. Отсюда непосредственно вытекает справедливость следующего предложения.

4.4. *Если функция непосредственной связи f_{ik} i-го узла вентильной схемы с k-м узлом представляется в виде дизъюнкции двух членов $f_{ik}=p_{ik} \vee q_{ik}$, один из которых (например, q_{ik}) является импликантой произведения $f_{ij} f_{jk}$ функций непосредственных связей i-го с j-м и j-го с k-м узлами заданной схемы (j — произвольный узел), то замена функции f_{ik} функцией p_{ik} в матрице непосредственных связей этой схемы не изменит функции полной связи i-го узла с k-м.*

Для быстрого нахождения элементов f_{ij}, f_{jk}, f_{ik} матрицы непосредственных связей можно воспользоваться тем очевидным обстоятельством, что эти три элемента можно рассматривать как три вершины прямоугольника, четвертая вершина которого (противоположная вершине с элементом f_{ik}) расположена на главной диагонали матрицы. Перебирая все такие прямоугольники, мы с помощью теоремы 4.4 можем в ряде случаев осуществить значительное упрощение заданной вентильной схемы.

Рассмотрим в качестве примера схему из односторонних вентилей, заданную матрицей непосредственных связей

$$A=\begin{Vmatrix} 1 & x & y & x \vee y \\ 0 & 1 & 0 & x \\ 0 & 0 & 1 & y \\ 0 & 0 & 0 & 1 \end{Vmatrix}.$$

В функции $x \vee y$ первый член является импликантой произведения $xx=x$, а второй — импликантой произведения $yy=y$. Поскольку тройки $(x, x, x \vee y)$, $(y, y, x \vee y)$

элементов матрицы A образуют, каждая, три вершины прямоугольника с четвертой вершиной на главной диагонали, мы можем, на основании теоремы 4. 4, заменить функцию $x \vee y$ нулем, не изменив функции полной связи первого узла схемы с четвертым.

Заметим, однако, что описанный прием упрощения матриц непосредственных связей применим далеко не всегда. Он не позволяет, например, осуществить упрощение схемы, изображенной на рис. 13, хотя эта схема допускает, как указывалось в предыдущем параграфе, большие упрощения.

Мы опишем поэтому общий прием упрощения матриц непосредственных связей вентильных схем, позволяющий находить в некотором смысле наипростейшие схемы. Этот прием (описанный ранее Хэмфри [1]) основан на последовательной минимизации каждого элемента матрицы непосредственных связей вентильной схемы (обычной или с исключенными узлами).

Обозначим через δ произвольный элемент матрицы непосредственных связей заданной вентильной схемы и поставим задачу минимизировать этот элемент, сохраняя функцию полной связи F между какой-нибудь фиксированной парой узлов. Рассматривая в процессе преобразований элемент δ как отдельную букву, мы всегда можем представить функцию F в виде $F=\alpha\delta\vee\beta$, где α и β — выражения булевой алгебры, не содержащие буквы δ.

Рассматривая теперь снова букву δ в ее первоначальном значении (как функцию основных переменных x_1, x_2, ...), заменим ее новой функцией λ. Условие сохранения значения функции F дает нам: $\alpha\delta\vee\beta=\alpha\lambda\vee\beta$ или $(\alpha\delta\vee\beta)+(\alpha\lambda\vee\beta)=0$, что эквивалентно соотношению $(\alpha\delta\vee\beta)(\overline{\alpha\lambda\vee\beta})\vee\overline{(\alpha\delta\vee\beta)}(\alpha\lambda\vee\beta)=0$. Из последнего соотношения мы получаем, что $\alpha\bar{\lambda}\delta\bar{\beta}\vee\alpha\lambda\bar{\delta}\bar{\beta}=0$, из чего, очевидно, следует справедливость двух равенств:

$$\alpha\bar{\lambda}\delta\bar{\beta}=0 \quad \text{и} \quad \alpha\lambda\bar{\delta}\bar{\beta}=0. \tag{1}$$

Считая, что $0\leqslant 0$, $0\leqslant 1$, $1\leqslant 1$, определим отношение неравенства для пар булевых функций, полагая $f_1\leqslant f_2$ в том и

[1]) W. S. Humphrey, Switching circuits with computer applications. Mc. Grow. Hill Book Co. inc., 1958.

только в том случае, когда это неравенство справедливо для значений функций f_1 и f_2 на любом наборе значений их аргументов. Ясно, что отношение $f_1 \leqslant f_2$ эквивалентно тому, что функция f_1 является импликантой функции f_2.

Заметим теперь, что из равенства нулю произведения $f_1 f_2$ вытекает справедливость неравенства $f_1 \leqslant \bar{f}_2$. Действительно, на всех наборах, на которых значения функции f_1 равны 1, значения функции f_2 должны быть равны 0, и, следовательно, функция $\bar{f}_2$ обращается на всех этих наборах в единицу.

Применяя это замечание к полученным выше соотношениям $\alpha\bar{\lambda}\delta\bar{\beta}=0$ и $\alpha\lambda\bar{\delta}\bar{\beta}=0$ (см. (1)), мы получаем из первого соотношения неравенство $\alpha\delta\bar{\beta} \leqslant \bar{\lambda}=\lambda$, а из второго — неравенство $\lambda \leqslant \overline{\alpha\bar{\delta}\bar{\beta}}=\bar{\alpha}\vee\delta\vee\beta$. Объединяя оба эти неравенства, мы находим пределы, в которых может изменяться рассматриваемый элемент λ матрицы непосредственных связей:

$$\alpha\delta\bar{\beta} \leqslant \lambda \leqslant \bar{\alpha} \vee \delta \vee \beta. \qquad (2)$$

В случае, когда необходимо сохранить не одну функцию полной связи, а несколько, требуется, чтобы функция λ удовлетворяла нескольким неравенствам вида (2),—по одному неравенству на каждую функцию полной связи. Нетрудно проверить, что получаемая таким образом система неравенств эквивалентна одному неравенству вида

$$\vee\alpha\delta\bar{\beta} \leqslant \lambda \leqslant \Pi(\bar{\alpha} \vee \delta \vee \beta). \qquad (3)$$

Дизъюнкция в левой части и произведение в правой части соотношения (3) распространяются на все неравенства системы, заменяемой этим соотношением.

Соотношение (3) *позволяет решать задачу минимизации матриц непосредственных связей вентильных схем* (*обычных или с исключенными узлами*) *при условии сохранения функций полной связи между некоторыми заданными узлами.*

Действительно, используя это соотношение, можно, очевидно, в конечное число шагов привести матрицу непосредственных связей заданной вентильной схемы к такому виду, что замена любого элемента δ этой матрицы функцией λ, требующей для своей реализации меньшего числа

вентилей, чем функция δ, приведет к изменению заданного множества M функций полных связей схемы.

Матрицу непосредственных связей вентильной схемы, удовлетворяющую последнему условию, естественно назвать *тупиковой* (по отношению к множеству M). Ясно, что матрица непосредственных связей, задающая наиболее экономную вентильную схему с заданным множеством M функций полной связи, относится к числу тупиковых. Нахождение такой *минимальной* матрицы связано, однако, с громоздким перебором, в связи с чем на практике при минимизации вентильных схем обычно ограничиваются нахождением какой-либо тупиковой матрицы.

Продемонстрируем процесс минимизации вентильной схемы на примере схемы, изображенной на рис. 13 (см. предыдущий параграф). Мы будем минимизировать эту схему при условии сохранения функций полной связи F_{13} и F_{14} (между первым и третьим и между первым и четвертым узлами).

Минимизируемая схема задается матрицей непосредственных связей:

$$A=\begin{Vmatrix} 1 & x\vee y & \bar{z} & 0 \\ 0 & 1 & z & \bar{z} \\ 0 & 0 & 1 & \bar{y} \\ 0 & 0 & 0 & 1 \end{Vmatrix}$$

Попробуем прежде всего заменить элемент $\delta=x\vee y$ этой матрицы более простым элементом λ. Исключая второй узел, приведем матрицу A к виду:

$$A_1=\begin{Vmatrix} 1 & z\delta\vee\bar{x} & \bar{z}\delta \\ 0 & 1 & \bar{y} \\ 0 & 0 & 1 \end{Vmatrix}.$$

Исключение дополнительно третьего узла приводит к матрице

$$A_2=\begin{Vmatrix} 1 & (\bar{y}\vee\bar{z})\delta\vee\bar{x}\bar{y} \\ 0 & 1 \end{Vmatrix},$$

а исключение четвертого узла — к матрице

$$A_3=\begin{Vmatrix} 1 & z\delta\vee\bar{x} \\ 0 & 1 \end{Vmatrix}.$$

Отсюда следует, что

$$F_{13} = z\delta \vee \bar{x}, \qquad F_{14} = (\bar{y} \vee \bar{z})\,\delta \vee \bar{x}\bar{y}.$$

Вводя обозначения $\alpha_1 = z$, $\beta_1 = \bar{x}$, $\alpha_2 = \bar{y} \vee \bar{z}$, $\beta_2 = \bar{x}\bar{y}$ и применяя соотношение (3), мы получим:

$$\alpha_1\bar{\beta}_1\delta \vee \alpha_2\bar{\beta}_2\delta \leqslant \lambda \leqslant (\bar{\alpha}_1 \vee \beta_1 \vee \delta)\,(\bar{\alpha}_2 \vee \beta_2 \vee \delta)$$

или, после подстановки значений α_1, α_2, β_1, β_2 и δ и упрощений:

$$x \vee y\bar{z} \leqslant \lambda \leqslant 1.$$

Наиболее простое решение будет, очевидно, если придать значение $\lambda = 1$. После этого второй узел в схеме делается излишним. Исключая этот узел, мы получаем схему, задаваемую матрицей

$$B = \begin{Vmatrix} 1 & z \vee \bar{x} & \bar{z} \\ 0 & 1 & \bar{y} \\ 0 & 0 & 1 \end{Vmatrix},$$

(чтобы получить ее, достаточно в матрицу A_1 подставить значение $\delta = 1$).

Обозначая теперь через δ элемент $\bar{y}$ матрицы B, мы получим, что

$$F_{13} = \bar{x} \vee z, \quad F_{14} = (z \vee \bar{x})\,\delta \vee \bar{z} = \delta \vee \bar{z}.$$

В этом случае мы имеем, очевидно:

$$\alpha_1 = 0, \ \beta_1 = \bar{x} \vee z, \ \alpha_2 = 1, \ \beta_2 = \bar{z}.$$

Применение формулы (3) дает:

$$0 \vee z\bar{y} \leqslant \lambda \leqslant 1 \cdot (\bar{y} \vee \bar{z}).$$

Так как в качестве λ нельзя выбрать ни нуль, ни единицу, то дальнейшее упрощение элемента $\bar{y}$ матрицы B оказывается невозможным. Аналогично, обозначая буквой λ значение, которым можно заменить элемент $z \vee \bar{x}$, мы придем к выводу, что $z \vee \bar{x} \leqslant \lambda \leqslant z \vee \bar{x}$, то есть что $\lambda = z \vee \bar{x}$.

Повторяя то же самое для элемента $\bar{z}$, мы придем к соотношению

$$(x \vee y)\,\bar{z} \leqslant \lambda \leqslant \bar{z} \vee \bar{y}\,(z \vee \bar{x}) = \bar{y} \vee \bar{z}.$$

Это соотношение показывает что дальнейшее упрощение (замена константой) элемента $\overline{z}$ также невозможно. Следовательно, матрица B является тупиковой.

Вентильная схема, соответствующая матрице B, состоит из четырех вентилей. Нетрудно показать, что заданные функции F_{13} и F_{14} невозможно реализовать менее чем четырьмя вентилями. Таким образом, найденная вентильная схема не только тупиковая, но и минимальная.

В заключение рассмотрим еще один прием синтеза вентильных схем, принадлежащий Ф. Свободе [1]); этот прием оказывается удобным для реализации с помощью цифровых автоматов. По методу Ф. Свободы каждому узлу схемы сопоставляется функция $F(x_1, \ldots, x_n)$ входных переменных схемы, значениями которой могут служить четыре величины, обозначаемые символами $0, 1, I, \sim$.

Функция F_i узла i принимает значение I на всех тех и только тех наборах значений переменных, при которых происходит передача сигнала от входного полюса схемы к данному узлу. Функция F_i принимает значение 0 на всех тех наборах значений входных переменных $x_1, \ldots, x_n$, при которых осуществляется передача сигнала от данного (i-го) узла хотя бы к одному из выходных полюсов схемы, на котором в этот момент должен возникать нулевой выходной сигнал. Функция F_i принимает значение 1 на всех тех наборах, на которых ее значение отлично от I и от 0 и при которых осуществляется передача сигнала от данного (i-го) узла к одному или нескольким из выходных полюсов схемы, в которых при этих условиях должен возникать единичный выходной сигнал. Функция F_i принимает (безразличное) значение $\sim$ во всех остальных случаях.

Процесс синтеза осуществляется с помощью двух операций, называемых ω_1 и ω_2. Операция ω_1 состоит во введении в схему нового узла, соединяемого с помощью какого-либо вентиля с одним из старых узлов. В случае односторонних вентилей элементарная ветвь должна включаться таким образом, чтобы обеспечивать передачу сигналов от нового узла к старому, а не наоборот.

---

[1]) Ф. Свобода, Синтез релейных схем при помощи машины. Авт. и телемех., т. 18, № 3, 1957, стр. 240—255.

Операция ω_2 состоит в соединении каким-либо вентилем двух уже имеющихся в схеме узлов.

В начальный момент синтеза на схеме изображается только входной полюс и все выходные полюсы синтезируемой схемы, не соединенные никакими вентилями. Функция входного узла (полюса) при этом тождественно равна I, а функции выходных узлов (полюсов) совпадают с заданными выходными функциями схемы. На каждом этапе синтеза выполняется одна из операций ω_1 или ω_2, в результате чего происходит изменение функций некоторых узлов (за исключением функции входного узла, которая остается все время постоянной). Процесс синтеза считается оконченным, когда для всех функций выходных узлов схемы все значения 1 превратятся в значения I.

Для определения порядка выполнения операций и вида присоединяемых вентилей определяются два типа вспомогательных функций входных переменных $x_1, \ldots, x_n$. Функция первого типа, которую мы будем обозначать через φ_i, сопоставляется каждому узлу i схемы. Ее значения определяются исключительно значениями функции соответствующего узла и могут быть найдены с помощью табл. V. 4.

Таблица V. 4

F_i	φ_i
0	0
1	1
I	~
~	~

Функция второго типа ψ_{ij} сопоставляется каждой упорядоченной паре (i, j) узлов схемы. Ее значения определяются по значениям функций узлов i и j в соответствии с табл. V. 5. При этом следует пользоваться двумя различными функциями ψ'_{ij} и ψ''_{ij} соответственно тому, употребляем ли мы для построения схемы односторонние или двусторонние вентили.

Соединять вентилем $\tilde{x}_k$ (x_k или $\bar{x}_k$) i-й узел схемы с j-м узлом можно только в тех случаях, когда на всех наборах, на которых функция ψ_{ij} равна нулю, значение $\tilde{x}_k$ также равно нулю. Вентили, удовлетворяющие этому условию, называются *допустимыми*.

Назовем *индексом* вентиля $\tilde{x}_k$ по отношению к паре узлов (i, j) число всех наборов, на которых функции ψ_{ij} и $\tilde{x}_k$ одновременно обращаются в единицу. Индексом вентиля $\tilde{x}_k$

Таблица V.5

F_iF_j	ψ'_{ij}	ψ''_{ij}	F_iF_j	d'_{ij}	d''_{ij}
00	~	~	*I*0	0	0
01	~	0	*I*1	1	1
0*I*	~	0	*II*	~	~
0~	~	~	*I*~	~	~
10	0	0	~0	~	~
11	1	1	~1	~	~
1*I*	~	1	~*I*	~	~
1~	~	~	~~	~	~

по отношению к одному узлу i называется число всех наборов, на которых функции φ_i и $\tilde{x}_k$ одновременно обращаются либо в нуль, либо в единицу.

Порядок выполнения операций в процессе синтеза определяется следующим правилом. Сначала выполняется операция ω_2 с помощью допустимых вентилей,— причем в первую очередь осуществляются соединения пар узлов вентилями, имеющими наибольший возможный индекс. Лишь только при невозможности выполнения операции ω_2 (то есть при отсутствии новых, еще не включенных допустимых вентилей для всех пар узлов) выполняется операция ω_1. При выполнении этой операции также следует следить за тем, чтобы в первую очередь осуществлялось подсоединение нового узла таким вентилем $\tilde{x}_k$ и к такому узлу i, что индекс вентиля $\tilde{x}_k$ по отношению к этому узлу является максимальным (при нескольких возможностях предпочтение отдается тем вентилям, индекс которых складывается из бо́льшого числа наборов, на которых имеет место совпадение единиц функций φ_i и $\tilde{x}_k$). После выполнения операции ω_1 снова делают все возможные операции ω_2.

Нетрудно усмотреть, что при употреблении в процессе выполнения операции ω_2 лишь допустимых вентилей мы никогда не придем к схеме, дающей единичные выходные сигналы тогда, когда она должна давать нулевые выходные сигналы. Вместе с тем, из определения функций φ_1 и ψ_{ij}

непосредственно следует, что выполнение операций ω_1 и ω_2 приближает нас к построению схемы, реализующей каждый заданный единичный выходной сигнал. Что же касается выбора для операций ω_1 и ω_2 вентилей максимального индекса, то это правило является эмпирическим и может быть оправдано лишь соображениями, носящими чисто качественный характер. Нетрудно понять, например, что выполнение операций ω_1 и ω_2 с вентилями, индекс которых равен нулю, нисколько не приближает нас к построению желаемой схемы.

Продемонстрируем работу описанного алгоритма синтеза на простом примере. Предположим, что нам необходимо построить вентильную схему, состоящую из двусторонних вентилей и реализующую две выходные функции δ и p, значения которых задаются табл. V.6 (такая схема называется вентильным полусумматором).

Таблица V.6

x	y	δ	p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

В процессе синтеза удобно все встречающиеся функции задавать векторами их значений на естественным образом упорядоченных наборах значений входных переменных x и y.

На первом шаге синтеза мы первоначально имеем три узла: входной узел 1 с функцией $(IIII)$ и два выходных узла 2 и 3 с функциями (0110) и (0001) соответственно. Функции ψ_{ij} для пар узлов имеют при этом вид

$$\psi_{12}=(0110),\quad \psi_{13}=(0001),\quad \psi_{23}=(\sim 000).$$

Выпишем еще функции, реализуемые вентилями:

$$x=(0011),\quad \bar{x}=(1100),\quad y=(0101),\quad \bar{y}=(1010).$$

Теперь очевидно, что ни для одной пары узлов ни один из вентилей не является допустимым. Операция ω_2 оказывается поэтому невозможной. Необходимо применить операцию ω_1, то есть произвести вставку нового, четвертого узла. Для выбора окончательного вида операции заметим, что по отношению к узлу 1 все вентили имеют индекс 0. По отношению к узлу 2 все вентили имеют индекс 2, а по отношению к узлу 3 вентили x и y имеют индекс 3, а вентили $\bar{x}$ и $\bar{y}$ — индекс 1. Таким образом, лучше всего новый узел 4 соединить с узлом 3 вентилем x или вентилем y.

Выберем любую из этих возможностей; например, будем вставлять вентиль x. После осуществления этой операции функции узлов представятся в виде

$$F_1=(IIII), \quad F_2=(0110), \quad F_3=(0001), \quad F_4=(\sim\sim 01).$$

Операция ω_2 оказывается возможной в применении к парам узлов (1,4), (4,2) и (3,4). Соответствующие функции имеют вид:

$$\psi_{14}=(\sim\sim 01), \quad \psi_{42}=(\sim\sim 00), \quad \psi_{34}=(\sim\sim 11).$$

Индекс 2 имеет вентиль y по отношению к паре (1,4) и вентиль $\bar{x}$ по отношению к паре (4,2). Первый вентиль оказывается более предпочтительным, так как он осуществляет совпадение единиц, тогда как второй — только совпадение нулей. Однако оказывается возможным после включения первого вентиля включить также и второй.

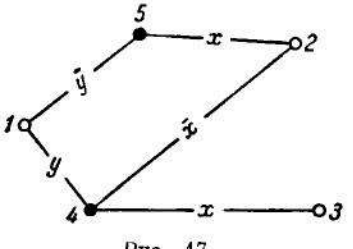

Рис. 17.

После этого функции узлов приобретают вид

$$F_1=(IIII), \quad F_2=(0I10), \quad F_3=(000I), \quad F_4=(0I0I).$$

Операция ω_2 снова оказывается невозможной. Новый узел 5 целесообразно соединить вентилем x с узлом 2, по отношению к которому этот вентиль имеет индекс 2. Осуществляя это соединение, мы получим в качестве функции

пятого узла функцию $F_5=(\sim\sim 10)$. Функции остальных узлов при этом, очевидно, сохраняются.

Функция ψ_{15} имеет вид $\psi_{15}=(\sim\sim 10)$, из чего видно, что вентиль $\overline{y}$ по отношению к паре узлов (1,5) имеет индекс 2. Осуществляя соединение этих узлов вентилем $\overline{y}$, мы придем к следующим функциям узлов:

$$F_1=(IIII),\quad F_2=(0II0),\quad F_3=(000I),$$
$$F_4=(0I0I),\quad F_5=(I\sim I0).$$

Тем самым процесс синтеза оказывается законченным. Полученная в результате этого процесса вентильная схема изображена на рис. 17.

ГЛАВА VI

НЕКОТОРЫЕ ПРОБЛЕМЫ НАДЕЖНОСТИ ЦИФРОВЫХ АВТОМАТОВ

§ 1. Потенциальные и импульсные сигналы. Основные типы схем цифровых автоматов

Целью настоящего параграфа, как и всей гл. VI, является изложение некоторых, самых простейших сведений о возможных искажениях сигналов в схемах цифровых автоматов. Эти сведения ни в коей мере не претендуют на то, чтобы полностью заменить инженерный расчет элементов и схем. Их назначение состоит лишь в том, чтобы дать некоторое представление о проблемах, возникающих после окончания логического структурного синтеза автомата. Заметим, однако, что уже на этом уровне возникает много интересных математических задач, имеющих большое значение для целей практического синтеза цифровых автоматов из реальных (неидеализированных) элементов.

В предыдущих главах мы имели дело с абстрактными алфавитными (в частности, с абстрактными двоичными) сигналами, которые считались возникающими лишь в определенные моменты времени, совокупность которых называлась нами дискретным автоматным временем. В настоящей главе мы будем рассматривать так называемые *абстрактные физические сигналы*, представляющие собой некоторые вещественные функции $f(t)$ обычного (непрерывного) времени, заданные на положительной полуоси $(0, +\infty)$.

Эти функции возникают следующим образом: в каждой точке реальной (физической) схемы автомата, участвующей в передаче или хранении сигналов, эти сигналы предстаавляются в виде той или иной физической величины

(электрическое напряжение, сила тока, напряженность магнитного поля и т. п.). Каждая такая величина, вообще говоря, изменяет свое значение в процессе работы автомата и может, следовательно, быть представлена в виде некоторой функции времени. Эта функция и называется абстрактным физическим сигналом в данной точке. Термином «абстрактный» в данном случае подчеркивается то обстоятельство, что мы отвлекаемся от реальной физической природы сигнала и интересуемся лишь процессом его изменения с течением времени.

В соответствии с видом функций, определяющих абстрактные физические сигналы, эти сигналы подразделяются на два больших класса, которые принято называть *потенциальными* и *импульсными сигналами*.

Опишем сначала так называемые *идеальные импульсные* и *потенциальные сигналы*. В целях осуществления такого описания выделим на оси времени точки, соответствующие последовательным моментам дискретного автоматного времени; эти точки мы будем называть *переходными* или *тактирующими*. Промежуток времени между любыми двумя соседними переходными точками назовем *тактом*.

Идеальным импульсным сигналом называется абстрактный физический сигнал, представляемый функцией, которая равна нулю во всех точках, отличных от переходных точек. Идеальным потенциальным сигналом называется абстрактный физический сигнал, представляемый функцией, которая остается постоянной в течение каждого временно́го такта и значения которой в переходных точках не определены.

При двоичной системе кодирования сигналов (то есть в двоичном структурном алфавите) для идеальных потенциальных сигналов считаются возможными только два каких-нибудь значения a и b, называемые уровнями сигналов. Одно из этих значений отождествляется с нулевым сигналом, другое — с единичным. Идеальные импульсные системы сигналов в двоичном структурном алфавите могут быть двух типов. К первому типу относятся такие системы, в которых нулевому сигналу в любой данный переходной момент времени соответствует нулевое значение функции (отсутствие импульса), а единичному сигналу — некоторое наперед

заданное фиксированное ненулевое значение a (наличие положительного или отрицательного импульса с амплитудой, равной $|a|$). В системах второго типа нулевой сигнал кодируется импульсом одной полярности, а единичный сигнал — импульсом другой полярности (однако, как правило, той же самой амплитуды); иначе говоря, в системах этого типа нулевому сигналу соответствует значение функции с определенным знаком (скажем, с отрицательным), а единичному сигналу — значение функции с противоположным знаком.

Чтобы избежать недоразумений, заметим сразу же, что все сказанное относится к представлению сигналов в каждой отдельной точке схемы. Существуют такие системы кодирования сигналов, при которых нулевой и единичный сигналы кодируются абстрактными физическими сигналами в двух параллельных каналах. Такие системы кодирования естественно называть *бинарными*. В ряде случаев они обладают существенными преимуществами по сравнению с обычными (м о н а р н ы м и) системами кодирования.

В реальных схемах цифровых автоматов не представляется, разумеется, возможным использовать потенциальные или импульсные сигналы, имеющие описанную выше

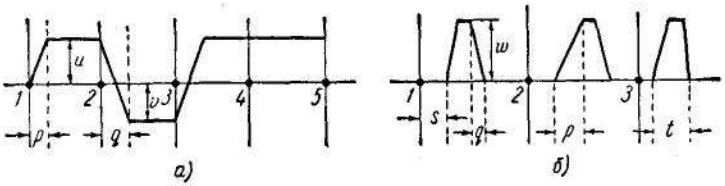

Рис. 18.

идеальную форму. Вместе с тем в большинстве случаев нецелесообразно учитывать абсолютно все возможные отличия реальных сигналов от идеальной формы. В связи с этим в практических приложениях теории обычно предполагается, что абстрактные физические потенциальные и импульсные сигналы имеют вид, указанный соответственно на рис. 18, *а* и *б*.

Временные интервалы p и q на этом рисунке называются соответственно *длиной переднего фронта* и *длиной*

заднего фронта потенциального и импульсного сигнала, интервал t — *длительностью импульса*, а интервал s — *временем запаздывания импульса*. Отношение времени такта к длительности импульса называют *коэффициентом скважности* или просто *скважностью импульса*. Все введенные величины могут изменяться при переходе от одного такта к другому, а также от одной точки схемы к другой. Уровни потенциальных сигналов (u и v) и амплитуда импульсных сигналов w также меняются. Однако все указанные изменения могут происходить только в некоторых, заранее определенных пределах. При выходе за эти пределы сигнал утрачивает свое значение, то есть, иными словами, не отождествляется ни с каким абстрактным алфавитным сигналом.

Проведенная классификация абстрактных физических сигналов позволяет провести классификацию схем цифровых автоматов по характеру сигналов, которые в них используются. По этому признаку можно выделить т р и основных типа схем: *потенциальные*, *импульсные* и *импульсно-потенциальные*. В потенциальных схемах употребляются только сигналы потенциального типа, в импульсных схемах — только сигналы импульсного типа, в импульсно-потенциальных схемах используются как импульсные, так и потенциальные сигналы.

Необходимость сохранения параметров, характеризующих абстрактные физические сигналы, в заданных пределах накладывает ряд ограничений на схемы; часть этих ограничений одинакова для схем всех трех указанных типов. Оказывается прежде всего, что всякий реальный элемент, генерирующий тот или иной выходной сигнал, обладает вполне определенной *нагрузочной способностью*, не позволяющей подсоединять выходной канал этого элемента к неограниченно большому числу входных узлов других элементов. На том уровне, которым ограничивается рассмотрение вопросов надежности в настоящей главе, достаточно в качестве характеристики нагрузочной способности элементов рассматривать так называемый *коэффициент разветвления*.

Под коэффициентом разветвления любого логического элемента в схеме автомата понимается наибольшее количество входных узлов других элементов, которое можно

подсоединить к выходу данного элемента без того, чтобы вызвать искажения сигналов, превышающие заданные границы. Эти границы, а с ними и коэффициент разветвления, обычно зависят от размера такта, увеличиваясь с его увеличением. Таким образом, за счет уменьшения скорости работы схемы, вообще говоря, можно увеличивать коэффициент разветвления.

Заметим еще, что входные каналы различных элементов могут представлять собою различные нагрузки для выходного канала любого данного элемента. В связи с этим при подсчете числа входных каналов, которые допустимо подсоединять к данному выходному узлу, можно употреблять *весовые коэффициенты* (в е с а) этих каналов. Сумма весов входных каналов, подсоединяемых в схеме к выходному узлу, не должна превышать коэффициента разветвления для этого узла.

Необходимость учета коэффициента разветвления вводит дополнительные трудности в задачу синтеза схем. Заметим, например, что в схеме пирамидального дешифратора выходной канал элемента, генерирующего последнюю (n-ю) переменную x_n, нагружается 2^{n-1} входными каналами двухвходовых совпадений или вентилей, составляющими половину последнего каскада схемы. При малом коэффициенте разветвления построение пирамидальных дешифраторов становится невозможным уже для относительно небольшого числа переменных.

Выход из положения находят обычно в том, что в схему автомата, наряду с логическими и запоминающими элементами, вводятся вспомогательные элементы, единственной задачей которых является увеличение коэффициента разветвления. Эти элементы не несут никаких логических функций и лишь повторяют на своих выходных узлах сигналы, подаваемые на их входные узлы. В связи с этим такие элементы называют *повторителями*.

Принято различать три вида повторителей, которые мы будем называть соответственно *простыми повторителями*, *усилителями* и *формирователями*. Простые повторители усиливают лишь мощность сигнала, увеличивая тем самым коэффициент разветвления. Что же касается различных параметров, характеризующих качество повторяемого сигнала (амплитуда или уровень сигнала и

крутизна фронтов), то они либо не изменяются в результате повторения, либо даже ухудшаются. Простые повторители употребляются в схемах (или частях схем), использующих сигналы потенциального типа. Будучи построенными на электронных лампах, они называются обычно *катодными повторителями*, а на транзисторах — *эмиттерными повторителями*. В этих названиях подчеркнуты некоторые особенности схемной реализации соответствующих элементов.

Усилители служат для такого повторения входных сигналов, при котором восстанавливаются до нормальных значений их амплитуды или уровни. Попутно может происходить также увеличение коэффициента разветвления, однако соблюдение этого условия для повторителей рассматриваемого типа не обязательно. В зависимости от вида своих входных и выходных сигналов усилители делятся на *потенциальные* и *импульсные*.

Потенциальные усилители обычно объединяются в одной схеме с инверторами и называются поэтому *инвертирующими усилителями*. При подаче на вход инвертирующего усилителя потенциального сигнала, соответствующего нулю или единице, на выходе его появляется потенциальный сигнал нормального уровня, отвечающий, соответственно, единице или нулю. Потенциальные инвертирующие усилители обычно не увеличивают коэффициент разветвления и употребляются поэтому в комбинации с обычными повторителями.

Импульсные усилители не только восстанавливают амплитуду (импульсных) сигналов до нормальной величины, но и увеличивают, как правило, коэффициент разветвления. Часто импульсные усилители объединяются в одной схеме с вентилями, которые называются в этом случае *усиливающими вентилями*.

Обычные усилители (потенциальные и импульсные) восстанавливают лишь уровень или амплитуду сигнала. Если же усилитель, кроме этого, восстанавливает до нормальной величины длительность переднего и заднего фронтов, то его принято называть формирователем. Название это связано с тем, что формирователь восстанавливает правильную форму сигнала по сигналу того же типа (нулевому или единичному), имеющему

искаженную форму (но остающемуся при этом в допустимых пределах изменений). В электронных схемах обычно употребляются *импульсные формирователи*. Иногда импульсный формирователь объединяется в одном элементе с так называемым дифференцирующим устройством. Этот элемент осуществляет дифференцирование сигналов потенциального типа, формируя на выходе стандартный импульсный сигнал в те моменты времени, когда меняется уровень потенциального сигнала на его входе (при уменьшении уровня входного сигнала формируется отрицательный импульс, а при увеличении — положительный).

Рассмотрим теперь основные типы запоминающих и логических элементов, из которых строятся схемы современных цифровых автоматов. При этом, в соответствии с принятым уровнем абстракции, мы не будем обсуждать здесь фактические способы реализации этих элементов, а лишь характер их функционирования и тип входных и выходных сигналов.

Одним из наиболее употребительных на практике запоминающих элементов является *потенциальный триггер*. По характеру функционирования различают три типа триггеров, а именно: триггеры со счетным входом, триггеры с раздельными входами и комбинированные триггеры. Описание их работы на уровне алфавитных структурных сигналов было дано в гл. III. Теперь необходимо отметить лишь, что все входные сигналы в потенциальных триггерах являются импульсными, а выходные — потенциальными. Как правило, потенциальный триггер обладает двумя выходными каналами — прямым и инверсным, причем сигналы в этих двух каналах всегда имеют противоположные значения (если в нулевом канале возникает нулевой сигнал, то в инверсном в это время возникает единичный сигнал, и наоборот).

Вторым широко употребляющимся на практике запоминающим элементом является элемент задержки. В *потенциальном элементе задержки* как входной, так и выходной сигнал являются сигналами потенциального типа, в *импульсном элементе задержки* оба эти сигнала — импульсные. Потенциальный элемент задержки может иметь либо один, либо два выходных канала. В последнем случае по одному из каналов выдается задержанный на один

такт входной сигнал, а по другому — его инверсия. Вместо простого импульсного элемента задержки часто употребляется *функциональный импульсный элемент задержки*. Такой элемент имеет несколько импульсных входных каналов $x_1, \ldots, x_n$ и один импульсный выходной канал, сигнал на котором возникает в тактирующий момент времени, непосредственно следующий за приходом сигналов $x_1, \ldots, x_n$ (предполагается, что все эти сигналы приходят в течение одного такта). Этот сигнал будет единичным или нулевым в зависимости от того, чему — единице или нулю,— равна некоторая, характеризующая выбранный элемент, булева функция $f(x_1, \ldots, x_n)$. Принято говорить при этом, что соответствующий элемент реализует эту функцию с задержкой на один такт.

Функциональные элементы задержки представляют собой соединение в одном элементе логического и запоминающего элемента. К числу чисто логических элементов, широко употребляющихся на практике, принадлежат потенциальные двухвходовые, трехвходовые — и вообще многовходовые совпадения и разделения. Как входные, так и выходные сигналы у этих элементов являются сигналами потенциального типа. Во многих случаях употребляется также *потенциальный инвертор*, осуществляющий преобразование потенциального входного сигнала в потенциальный выходной сигнал, имеющий противоположное значение (в смысле двоичных сигналов).

Широкое употребление в схемах цифровых автоматов находят также *импульсно-потенциальные вентили*, то есть такие вентили, в которых управляющие входные сигналы являются потенциальными, а вентильные входные и выходные сигналы — импульсными. Существуют также потенциально-потенциальные вентили, у которых все сигналы (как входные, так и выходные) являются потенциальными.

Все описанные элементы допускают самые различные конкретные воплощения. Для начальной ориентировки читателя в этом вопросе отметим, что повторители, усилители, формирователи, потенциальные триггеры и потенциальные инверторы обычно изготовляются на основе ламповых и транзисторных схем, потенциальные совпадения

и разделения реализуются чаще всего с помощью полупроводниковых диодов и сопротивлений. Функциональные импульсные элементы задержки строятся с использованием магнитных сердечников (а в ряде случаев также и электронных ламп и транзисторов), импульсно-потенциальные вентили собираются чаще всего из импульсных трансформаторов и полупроводниковых диодов. Что же касается потенциально-потенциальных вентилей, то в комбинации с потенциальным элементом задержки они воплощаются в обычных электромагнитных реле и их различных электронных аналогах.

Рассмотрим теперь основные типы схем цифровых автоматов, использующих перечисленные элементы, имея в виду выявить возможные источники искажений сигналов в этих схемах. Прежде всего необходимо отметить *чисто потенциальные схемы*. В качестве запоминающих элементов в таких схемах используются обычно потенциальные элементы задержки, все сигналы, в том числе и внешние (входные и выходные), являются потенциальными. Для построения комбинационной части чисто потенциальных схем употребляются два способа: способ, основанный на применении потенциальных совпадений, разделений и инверторов, и способ, основанный на применении потенциально-потенциальных вентилей. Первый способ (он применяется в случае электронных схем) предполагает использование элементов задержки с одним выходом, второй способ (который применяется в случае схем, составленных из электромагнитных реле) предполагает использование элементов задержки с двумя выходами.

Потенциальные схемы могут быть построены как на принципе *естественного тактирования* (*асинхронные схемы*), так и на принципе *принудительного тактирования* (*синхронные схемы*). При естественном тактировании выходы запоминающих элементов (элементов задержки) через схему обратной связи (составленную из логических элементов) управляют своими собственными входами. Это обстоятельство создает возможность неправильного срабатывания схемы за счет явления, получившего в литературе название «гонок» [1]).

[1]) См. S. H. Caldwell, Switching circuits and logical design. John. Willey and son inc., 1958

Суть явления гонок состоит в следующем: элементы задержки, употребляющиеся в схеме в качестве запоминающих элементов, имеют различные (хотя обычно и достаточно близкие друг другу) времена срабатывания. Различны также задержки сигналов (то есть длины передних фронтов), поступающих на входные каналы элементов задержки по логическим цепям, имеющим неодинаковую длину. Если по условиям срабатывания (таблице переходов) автомата в какой-то момент времени должны изменить свое состояние сразу несколько запоминающих элементов, то между элементами начинаются г о н к и. Тот элемент, который в ы и г р а е т эти гонки, т. е. изменит свое состояние раньше, чем другие элементы, может через цепь обратной связи изменить сигналы на входах некоторых запоминающих элементов д о т о г о, как другие участвующие в гонке элементы изменят свое состояние. Это, как нетрудно понять, может вызвать переход автомата совсем не в то состояние, которое предусмотрено таблицей переходов автомата.

Проблема устранения возможности неправильного функционирования схемы за счет описанного явления носит название п р о б л е м ы г о н о к. Проблему гонок можно решить за счет точного согласования времени передачи сигналов по цепям обратной связи с временем полного изменения состояний запоминающих элементов, которое для краткости в дальнейшем будет называться временем их переброса. Более радикальным является такое решение этой проблемы, которое исключает необходимость о д н о в р е м е н н о г о изменения состояний с р а з у у н е с к о л ь к и х запоминающих элементов. С этой целью достаточно, очевидно, применить такое кодирование состояний заданного автомата, при котором (двоичные) коды любых двух с о с е д н и х состояний (состояний, переходящих одно в другое под воздействием какого-нибудь входного сигнала) отличаются друг от друга не более чем одним разрядом. Условимся называть такое кодирование *противогоночным*.

Нетрудно понять, что противогоночное кодирование оказывается возможным далеко не во всех случаях. Действительно, если имеется три различных состояния a, b, c и три (необязательно различных) входных сигнала x, y, z

таких, что $ax=b$, $by=c$, $cz=a$, то противогоночное кодирование состояний a, b, c невозможно, поскольку код состояния c неизбежно будет отличаться от кода состояния a в двух разрядах, если коды состояний a, b и коды состояний b, c отличаются между собой в одном разряде.

Этот пример показывает, что возможность противогоночного кодирования должна обеспечиваться еще на этапе абстрактного синтеза автомата.

Для асинхронных автоматов, то есть таких автоматов, у которых время такта определяется скоростью переброса различных его запоминающих элементов, существенным (для обеспечения правильной работы автомата) является также согласование изменения (потенциальных) входных сигналов с переходами автомата из одного состояния в другое.

Такое согласование осуществляется обычно за счет того, что входной сигнал x заменяется входным сигналом y лишь тогда, когда автомат находится в стабильном, относительно x, состоянии a, то есть когда $ax=a$. Ясно, что в этом случае отпадает необходимость точного согласования момента изменения входного сигнала с переходами автомата из одного состояния в другое, поскольку начиная с некоторого момента времени и вплоть до изменения входного сигнала автомат будет пребывать в одном и том же состоянии a. Назовем описанный прием *стандартным приемом согласования асинхронного автомата с изменениями входных сигналов*.

Необходимость решения проблемы гонок и проблемы согласования переходов автомата из одного состояния в другое с изменениями входных сигналов отпадает при введении *принудительного тактирования автомата*, то есть при превращении автомата в синхронный автомат. Для потенциальных схем принудительное тактирование осуществляется обычно за счет удвоения числа запоминающих элементов и введения специального *тактирующего генератора*.

Предположим, что по условиям работы автомата для него достаточно иметь n запоминающих элементов. В таком случае, кроме этих n элементов, составляющих первую ступень запоминания, в автомат вводятся еще n запоминающих элементов, составляющих

вторую ступень запоминания. Тактирующий генератор вырабатывает два управляющих напряжения, сменяющих друг друга с вполне определенной частотой. Моменты изменения этих напряжений и будут тактирующими (переходными) моментами. Элементы задержки, составляющие обе ступени запоминания, управляются напряжениями, вырабатываемыми тактирующим генератором, и осуществляют задержку сигналов как раз на время одного такта.

Во все нечетные такты (время действия первого управляющего напряжения) осуществляется передача информации (без ее изменения) из первой ступени запоминания во вторую. Выходы элементов второй ступени через комбинационную схему обратной связи связаны со входами элементов первой ступени. Через эту схему во все четные такты (время действия второго управляющего напряжения) осуществляется передача информации (с попутным ее преобразованием) из второй ступени запоминания в первую. Изменение входных сигналов происходит лишь в тактирующие моменты времени.

Очевидно, что описанный способ организации циркуляции информации полностью исключает опасность неправильного срабатывания автомата в результате гонок.

Для асинхронных потенциальных схем, использующих запоминающие элементы с двумя выходами, имеется еще одна опасность неправильного срабатывания, связанная с неодновременностью установления сигналов на прямом и на инверсном выходе после перехода запоминающего элемента из одного состояния в другое. В результате указанной неодновременности в какой-то момент времени может оказаться, что некоторая переменная x и ее инверсия $\bar{x}$ принимают одинаковые значения. Проблема устранения источников возникновения подобных ситуаций носит название *проблемы риска* [1]. Дальнейшее уточнение этой проблемы и разбор методов ее решения будет произведен в следующем параграфе. А сейчас мы отметим еще одну проблему, связанную с надежностью потенциальных

[1]) См. D. A. Huffman, The design and use of hazard-free switching networks. Journ. Ass. Comp. Machinery, v. 4, № 1, 1957, p. 47—62.

схем. Дело заключается в том, что обычно употребляющиеся потенциальные совпадения и разделения снижают на выходе разницу уровней нулевого и единичного сигналов по сравнению с сигналами на своих входах. Поэтому многоступенчатые потенциальные комбинационные схемы, составленные из совпадений и разделений, оказываются относительно мало надежными, в силу чего на практике ограничиваются часто двуступенчатыми схемами, причем первая ступень обычно строится из многовходовых совпадений, выходы которых подсоединяются к входам разделений, составляющих вторую ступень схемы. Нетрудно понять, что подобные схемы соответствуют представлению их выходных функций в дизъюнктивных нормальных формах (каждое совпадение, составляющее первую ступень, реализует одно из входящих в эти формы элементарных произведений). Возможно, разумеется, строить двуступенчатые схемы, в которых разделения предшествуют совпадениям. Такие схемы соответствуют представлению их выходных функций в конъюнктивных нормальных формах.

Переходим теперь к рассмотрению импульсно-потенциальных схем автоматов. В качестве запоминающих элементов в таких схемах употребляются обычно потенциальные триггеры. В наших рассмотрениях мы будем предполагать, что входные сигналы всей схемы в целом являются сигналами потенциального типа. Подобное предположение не нарушает общности рассуждения, поскольку импульсный сигнал всегда можно превратить в потенциальный, подавая его на триггер для запоминания.

Мы будем различать три вида импульсно-потенциальных схем: *схемы с потенциальной логикой*, *схемы с импульсной (вентильной) логикой* и *комбинированные схемы*.

В схемах с потенциальной логикой вся комбинационная часть схемы (реализующая функции возбуждения и выходные функции) строится обычно на потенциальных совпадениях и разделениях с добавлением в ряде случаев еще и потенциальных инверторов. На входах триггеров ставятся вентили, работающие в режиме *преобразователя* потенциального сигнала в импульсный. Это означает, что на их вентильные входы подаются в каждый тактирую-

щий момент импульсы от внешнего источника (генератора синхронизирующих импульсов). На управляющие входы вентилей подаются выходные (потенциальные) сигналы схемы обратной связи автомата. Выходные сигналы схемы являются потенциальными. В случае необходимости эти сигналы можно превратить в импульсные: стоит лишь использовать их в качестве управляющих сигналов вентилей, работающих в режиме преобразования потенциальных сигналов в импульсные.

Учет надежности для схем этого вида, на принятом уровне абстракции, ограничивается соблюдением заданных значений коэффициента разветвления и недопущением превышения заданного числа ступеней в потенциальных комбинационных схемах. Заметим, впрочем, что использование потенциальных инверторов дает возможность восстанавливать нормальную разность уровней нулевого и единичного сигналов и использовать вследствие этого многоступенчатые схемы.

В импульсно-потенциальных схемах с импульсной логикой комбинационная часть схемы строится на импульсно-потенциальных вентилях, управляемых выходными сигналами триггеров и входными сигналами схемы. Выходные (импульсные) сигналы этой схемы направляются непосредственно на входы триггеров и на выходные полюсы автомата. Если выходные сигналы должны быть потенциальными, то полученные импульсные выходные сигналы направляются на входы специально вводимых для этой цели триггеров, которые осуществляют преобразование их в одноименные сигналы потенциального типа. Кроме вентилей в комбинационной части схемы обычно используются также импульсные усилители и формирователи. На вентильный вход комбинационной схемы подаются импульсы от тактирующего генератора.

Учет надежности в данном случае не ограничивается лишь одним соблюдением заданных значений коэффициента разветвления. Необходимо учитывать также еще две возможности неправильного функционирования автоматов. Первая возможность — это *возможность многократного переброса триггера под влиянием импульсных сигналов, прошедших через вентильную схему различными путями, получивших в результате этого различные*

задержки и воспринимающиеся на выходе триггера как несколько пришедших друг за другом импульсов.

Ясно, что такая возможность имеется лишь при работе триггера в режиме счетного входа, так как в триггере с раздельными входами приход нескольких импульсов по одному и тому же входному каналу вызывает тот же эффект, что и приход одного импульса. В случае же триггеров со счетным входом устранение возможности неправильного срабатывания может быть произведено за счет введения в схемы дополнительных вентилей, устраняющих возможность прихода в один и тот же (вентильный) выходной узел нескольких импульсов в течение одного такта. Пример такого устранения будет приведен в следующем параграфе.

Вторая возможность неправильного срабатывания схем с импульсной логикой заключается в неодновременном перебросе триггеров: на некоторые из триггеров импульсы могут прийти по коротким цепям с малой задержкой и вызвать переброс этих триггеров в тот момент времени, когда еще не закончилось прохождение импульсов по более длинным путям; в результате этого может произойти ошибка в передаче этих отставших импульсов по заключительным участкам их путей.

Устранение этой опасности производится с помощью введения дополнительных задержек импульсов на входах триггеров, исключающих возможность переброса триггеров до окончания процесса распространения импульсов по всем цепям рассматриваемой вентильной схемы. Другой способ устранения описанной опасности заключается в удвоении числа триггеров и в образовании двух ступеней запоминания,— подобно тому, как это было сделано выше в случае потенциальных схем. В рассматриваемом случае вентильная схема становится управляемой лишь триггерами второй ступени; при этом в нечетные такты осуществляется передача информации от первой ступени ко второй, а в четные — от выходов второй ступени через вентильную схему к входам первой ступени.

В комбинированных импульсно-потенциальных схемах комбинационная часть выполняется частично на потенциальных элементах (совпадениях и разделениях), частично — на импульсно-потен-

циальных вентилях. Учет надежности в этом случае включает в себя все факторы, которые приходилось учитывать в двух уже описанных видах импульсно-потенциальных схем.

Последний тип схем — это чисто импульсные схемы. Все сигналы в такого рода схемах, в том числе входные и выходные сигналы всей схемы в целом, являются импульсными сигналами. Наиболее часто импульсные схемы строятся из описанных выше функциональных импульсных элементов задержки, представляющих собою объединение в одном элементе как запоминающего, так и логического элемента. Элементы выбираются таким образом, чтобы реализуемые ими булевы функции (без учета задержки) составляли бы функционально полную (см. гл. III) систему булевых функций. Моменты появления выходных сигналов на элементах определяются специальным тактирующим генератором.

Функции возбуждения и выходные функции автомата реализуются в этом случае с помощью суперпозиции выбранных элементов. При этом необходимо следить за тем, чтобы задержки, с которыми реализуются все эти функции, были одинаковыми. Если задержки оказываются различными, то они выравниваются посредством включения в схему дополнительных (нефункциональных) элементов задержки. В соответствии с тем, на сколько тактов задерживается образование сигналов для функций возбуждения автомата, каждый такт работы данного автомата расщепляется на соответствующее число тактов, в течение которых завершается полный цикл обращения информации по цепи обратной связи.

Учет надежности, на принятом уровне абстракции, ограничивается выполнением условия, чтобы коэффициент разветвления в схеме не превышал заданной величины.

§ 2. Проблема риска. Примеры синтеза схем с учетом простейших соображений надежности

Как уже сообщалось в предыдущем параграфе, проблема риска возникает в потенциальных асинхронных схемах,— в частности в схемах, построенных из электромагнитных реле.

Рассмотрим произвольную асинхронную потенциальную схему, имеющую n запоминающих элементов (потенциальных элементов задержки). Пусть $y_i = f_i(x_1, \ldots, x_n, a_1, \ldots, a_m)$, где $i = 1, \ldots, n$, представляют собой канонические уравнения этой схемы. Здесь $a_1, \ldots, a_m$ — входные сигналы автомата, а $x_1, \ldots, x_n$ — выходные сигналы его памяти. Вследствие различной длительности переходных процессов в выходных цепях запоминающих элементов может случиться, что в течение какого-то (обычно весьма короткого) времени на прямом и инверсном выходе одного и того же элемента появятся одноименные сигналы (нулевые или единичные). При этом имеется возможность получения неправильного сигнала на входах запоминающих элементов и связанный с нею риск неправильного срабатывания этих элементов.

Нетрудно понять, что применительно к запоминающим элементам, которые в наступающем такте должны изменить свое состояние, получение неправильного входного сигнала в начале такта не приведет к неправильному срабатыванию. В самом деле, под влиянием этого неправильного сигнала элемент может либо сохранить свое прежнее состояние, либо перейти в противоположное состояние. Поскольку в течение рассматриваемого такта состояние элемента так или иначе должно измениться, то в первом случае произойдет лишь некоторая затяжка этого изменения, а во втором изменение произойдет тогда, когда это требуется, а именно в начале такта.

Таким образом, риск неправильного срабатывания имеет место лишь для тех запоминающих элементов, которые должны сохранять свое состояние в течение рассматриваемого такта. Если это состояние — единичное, то говорят о риске в единице, если нулевое — то о риске в нуле.

Предположим, что схема построена на основе противогоночного кодирования состояний. Иными словами, в течение каждого такта работа схемы изменяет состояние лишь одного элемента. Для проверки наличия риска в такой схеме достаточно, очевидно, подвергнуть все цепи, реализующие входные функции памяти автомата (функции возбуждения)

$$f_i(x_1, \ldots, x_n \; a_1 \; \ldots, a_m) \quad (i = 1 \quad \ldots \quad n),$$

следующей проверке: если функция f_l не меняет своего значения при изменении значения какого-либо своего аргумента x_i или a_j на противоположное (при каких-либо фиксированных значениях остальных аргументов), то проверяется, не изменит ли функция f_l своего значения, если в ее конкретное представление

$$f_l = F_l(x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\ a_1\ \bar{a}_1, \ldots, a_m, \bar{a}_m),$$

реализуемое заданной схемой, подставить одно и то же значение (0 или 1) как для самого изменяющегося аргумента, так и для его инверсии (сохраняя неизменными значения всех остальных переменных). Если такое изменение имеет место, то говорят, что *при принятом представлении функции f_l имеется риск по соответствующему аргументу, причем при изменении значения функции f_l с 0 на 1 говорят о риске в нуле, а при изменении значения f_l с 1 на 0 — о риске в единице. Представление функции f_l называется свободным от риска, если при этом представлении не имеется риска ни по одному из аргументов.*

Ясно, разумеется, что о риске мы будем говорить лишь применительно к тем схемам, которые имеют в качестве своих входных сигналов не только значения соответствующих переменных, но и их отрицаний.

Рассмотрим в качестве примера функцию f от трех переменных, заданную следующей формулой (отражающей структуру соответствующей схемы): $f = xy \vee \bar{x}z$. Применительно к этому представлению речь может итти лишь о риске по переменной x. Нетрудно усмотреть, что функция f сохраняет значение 1 при изменении значения x в случае, когда $y=1$ и $z=1$, и сохраняет значение 0, когда $y=0$, $z=0$. При одновременном обращении в нуль x и $\bar{x}$ выбранное представление дает для функции f нулевое значение для любых значений y и z, в том числе и для значений $y=1$, $z=1$. Иными словами, при принятом представлении имеется риск в единице по переменной x. В то же время при других представлениях этой функции, например при представлении $f = xy \vee \bar{x}z \vee yz$, риск по переменной x (как, впрочем, и по всем остальным переменным) отсутствует.

Нетрудно удостовериться в справедливости следующего предложения.

2. 1. *При представлении булевой функции произвольной дизъюнктивной нормальной формой отсутствует риск в нуле, а при представлении произвольной конъюнктивной нормальной формой — риск в единице.*

В самом деле, для любой переменной x произвольную д.н.ф. заданной функции f можно представить в виде $f=A\bar{x}\vee Bx\vee C$, где A, B и C — булевы функции, не зависящие от x. Если при каких-то фиксированных значениях всех переменных, отличных от x, функция f равна нулю как при $x=0$, так и при $x=1$, то это означает, очевидно, одновременное равенство нулю соответствующих значений функций A, B, C. Но тогда $f=0$ при любом выборе значений для x и $\bar{x}$, в том числе и значений $x=1$, $\bar{x}=1$ или $x=0$, $\bar{x}=0$. Тем самым отсутствие риска в нуле доказано. Доказательство отсутствия риска в единице для случая к.н.ф. проводится точно таким же образом.

Хотя всякое представление булевой функции в виде д.н.ф. свободно от риска в нуле, разобранный выше пример показывает, что остается все же возможность риска в единице. Оказывается, однако, что справедливо следующее предложение.

2. 2. *Представления булевых функций в виде сокращенных д.н.ф. и сокращенных к.н.ф. свободны от риска (как в нуле, так и в единице) по всем переменным.*

Действительно, пусть, независимо от выбора значения какой-либо переменной x, существует такой набор значений остальных переменных, при котором заданная булева функция f равна единице. Это означает, очевидно, что в совершенной д.н.ф. функции f имеются две соседние конституенты единицы $\bar{x}p$ и xp. С помощью метода Квайна или Блейка отсюда непосредственно выводится, что функция f обладает простой импликантой, не зависящей от x. Эта импликанта сохраняет свое значение независимо от выбора значений для x и $\bar{x}$. Поскольку она обязательно содержится в сокращенной д.н.ф. функции f, то эта форма свободна от риска в единице по переменной x. Ввиду произвольности выбора x и в силу предложения 2. 1 очевидно, что сокращенная д.н.ф. свободна от риска (как в нуле,

так и в единице) по всем переменным. Доказательство предложения 2. 2 для случая сокращенной к.н.ф. производится точно таким же способом.

Доказанная теорема вместе с разобранным выше примером показывает, что при синтезе асинхронных потенциальных схем по соображениям надежности иногда целесообразно отказываться от реализации минимальных представлений булевых функций и пользоваться представлениями в виде сокращенных д.н.ф. и к.н.ф.

Рассмотрим теперь примеры синтеза схем автоматов с учетом тех соображений надежности, которые были рассмотрены в настоящем и в предыдущем параграфе.

Пример 1. Построить асинхронную потенциальную схему на двусторонних потенциальных вентилях и потенциальных элементах задержки с двумя выходами для автомата Мура, заданного следующей отмеченной таблицей переходов VI. 1:

Таблица VI. 1

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

Таблица VI. 2

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

Коэффициент разветвления выбранных элементов предполагается настолько большим, что его влияние учитывать не нужно (это всегда можно предполагать в случае, когда схема не слишком велика, а элементами являются электромагнитные реле).

Решение. Применим противогоночное кодирование состояний автомата. Такое кодирование может быть сделано, например, следующим образом: 0=00, 1=01, 2=11, 3=10.

В силу теоремы 8. 1 из § 8 гл. III таблица возбуждений автомата совпадает с его структурной таблицей переходов, то есть, иными словами, имеет вид, показанный в табл. VI. 2.

Обозначая состояния элементов памяти буквами x и y, входной сигнал автомата буквой a, выходной сигнал

буквой z, а функции возбуждения буквами f_1 и f_2, мы получим представления функций z, f_1 и f_2 картами Карнау, показанные соответственно в табл. VI. 3, VI. 4, VI. 5.

Таблица VI. 3

xy \ a	0	1
00	0	0
01	1	1
11	1	1
10	0	0

Таблица VI. 4

xy \ a	0	1
00	0	0
01	1	0
11	1	1
10	0	1

Таблица VI. 5

xy \ a	0	1
00	0	1
01	1	1
11	1	0
10	0	0

С помощью построенных карт находим минимальные дизъюнктивные нормальные формы соответствующих функций:

$$z = y,\ f_1 = y\bar{a} \vee xa,\ f_2 = \bar{x}a \vee y\bar{a}.$$

Для устранения риска необходимо перейти к сокращенным д.н.ф.:

$$z = y,$$
$$f_1 = xy \vee xa \vee y\bar{a} = x(a \vee y) \vee y\bar{a},$$
$$f_2 = \bar{x}y \vee \bar{x}a \vee y\bar{a} = \bar{x}(a \vee y) \vee y\bar{a}.$$

Схема автомата, реализующая найденные представления, изображена на рис. 19.

Построенная вентильная схема имеет шесть вентильных узлов и задается следующей матрицей непосредственных связей:

$$A = \begin{Vmatrix} 1 & x & 0 & y & \bar{x} & 0 \\ x & 1 & a \vee y & 0 & 0 & 0 \\ 0 & a \vee y & 1 & a & 0 & 0 \\ y & 0 & \bar{a} & 1 & 0 & \bar{a} \\ \bar{x} & 0 & 0 & 0 & 1 & a \vee y \\ 0 & 0 & 0 & \bar{a} & a \vee y & 1 \end{Vmatrix}.$$

Исключение второго и пятого узлов приводит к матрице

$$A_1 = \begin{Vmatrix} 1 & x(a \vee y) & \underline{y} & \bar{x}(a \vee y) \\ x(a \vee y) & 1 & \bar{a} & 0 \\ y & \bar{a} & 1 & \bar{a} \\ \bar{x}(a \vee y) & 0 & \bar{a} & 1 \end{Vmatrix}.$$

Продолжая исключение, мы приходим к матрицам

$$A_{13} = \begin{Vmatrix} 1 & F_{13} \\ F_{13} & 1 \end{Vmatrix}, \quad A_{14} = \begin{Vmatrix} 1 & F_{14} \\ F_{14} & 1 \end{Vmatrix}, \quad A_{16} = \begin{Vmatrix} 1 & F_{16} \\ F_{16} & 1 \end{Vmatrix},$$

где

$$F_{13} = x(a \vee y) \vee y\bar{a},$$
$$F_{14} = y \vee x\bar{a}(a \vee y) \vee \bar{x}\,\bar{a}(a \vee y) = y,$$
$$F_{16} = \bar{x}(a \vee y) \vee y\,\bar{a}$$

представляют собой функции полной связи от узла *1* к узлам *3*, *4*, *6*. Поскольку эти функции совпадают с заданными функциями f_1, z, f_2, то схема, изображенная на рис. 19, действительно решает поставленную задачу.

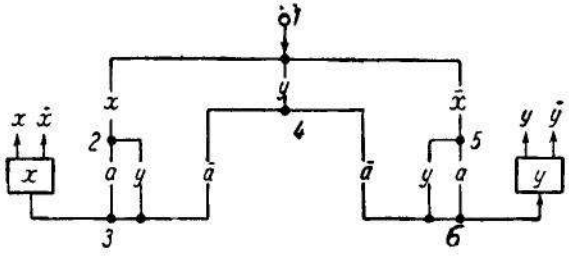

Рис. 19

В случае употребления двусторонних вентилей всегда является желательной проверка, аналогичная только что проведенной, поскольку при синтезе схемы в этом случае могут образоваться ложные пути, приводящие к ошибочным значениям сигналов в выходных узлах.

Посмотрим, как выглядела бы схема только что построенного автомата при реализации ее на потенциальных триггерах со счетными входами, двухвходовых потенци-

альных совпадениях и разделениях и импульсно-потенциальных вентилях.

В силу теоремы 8. 2 из § 8 гл. III таблица возбуждений автомата (при принятой выше системе кодирования) примет вид, показанный на табл. VI. 6,

Таблица VI. 6

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

а карты Карнау для функций возбуждения f_1 и f_2 приобретают вид, представленный в табл. VI. 7 и VI. 8.

Таблица VI. 7

xy \ a	0	1
00	0	0
01	1	0
11	0	0
10	1	0

Таблица VI. 8

xy \ a	0	1
00	0	1
01	0	0
11	0	1
10	0	0

Учитывая, что выходная функция z сохраняет свое старое значение, мы получаем следующие представления для интересующих нас функций:

$$z = y, \quad f_1 = (\bar{x}y \vee x\bar{y})\,\bar{a}, \quad f_2 = (\bar{x}\,\bar{y} \vee xy)\,a.$$

Полученные представления приводят к схеме автомата, изображенной на рис. 20 (на этом рисунке $СИ$ обозначает канал, по которому передаются синхронизирующие импульсы от тактирующего генератора, T_1 и T_2 обозначают триггеры, а B — вентили).

Если допустить использование наряду с совпадениями и разделениями также потенциального инвертора, то в полученной схеме можно заменить два совпадения и

разделения, реализующие функцию $\bar{x}\bar{y} \vee xy$ одним инвертором, поскольку, как нетрудно заметить, $\overline{\bar{x}\bar{y} \vee xy} = \overline{xy \vee x\bar{y}}$.

Пример 2. Построить схему n-разрядного двоичного параллельного сумматора, используя в качестве запоминающих элементов потенциальные триггеры со счетными входами, а в качестве логических элементов — односторонние импульсно-потенциальные вентили.

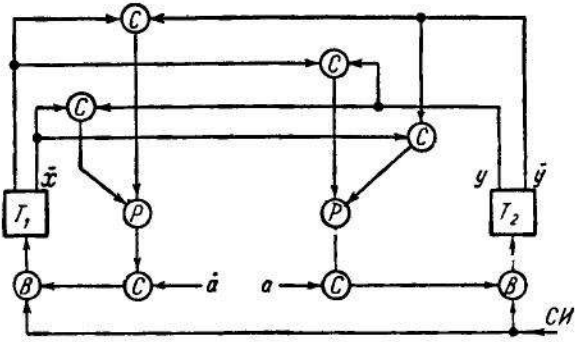

Рис. 20.

Решение. Для запоминания двух слагаемых используем $2n$ триггеров со счетными входами, i-й разряд первого числа обозначим через x_i, а i-й разряд второго числа — через y_i. Через p_i обозначим (двоичный) перенос из $(i-1)$-го разряда в i-й разряд $(i=1, \ldots, n)$. Условимся также i-й разряд суммы z_i получать в триггере, в котором хранится i-й разряд второго слагаемого (y_i). Сумматор с таким свойством естественно называть *накапливающим*, поскольку, подавая на триггеры $x_1, \ldots, x_n$ одно слагаемое за другим и запоминая вначале на триггерах $y_1, \ldots, y_n$ число 0, мы будем накапливать на триггерах $y_1, y_2, \ldots, y_n$ сумму всех этих слагаемых.

Для i-го разряда суммы z_i имеется следующее очевидное уравнение: $z_i = x_i + y_i + p_i$ $(i=1, \ldots, n)$. Функция возбуждения f_i триггера y_i (сигнал, подаваемый на его счетный вход) будет выражаться следующим образом (см. теорему 8. 2 из § 8 гл. III):

$$f_i = z_i + y_i = x_i + p_i = x_i\bar{p}_i \vee \bar{x}_i p_i. \qquad (1)$$

В каждом разряде (кроме самого старшего) необходимо образовать еще две функции — функцию p_{i+1} переноса из i-го разряда в $(i+1)$-й разряд и ее отрицание $\bar{p}_{i+1}$, поскольку соответствующие этим функциям сигналы используются в следующем $(i+1)$-м разряде. Нетрудно понять, что функция переноса имеет следующее выражение:

$$p_{i+1} = x_i y_i \vee x_i p_i \vee y_i p_i \quad (i = 1, 2, \ldots, n-1). \quad (2)$$

Функция отрицания переноса представится в виде:

$$\bar{p}_{i+1} = \bar{x}_i \bar{y}_i \vee \bar{x}_i \bar{p}_i \vee \bar{y}_i \bar{p}_i \quad (i = 1, 2, \ldots, n-1). \quad (3)$$

Вентильная схема, реализующая найденные представления функций f_i, p_{i+1} и $\bar{p}_{i+1}$, изображена на рис. 21.

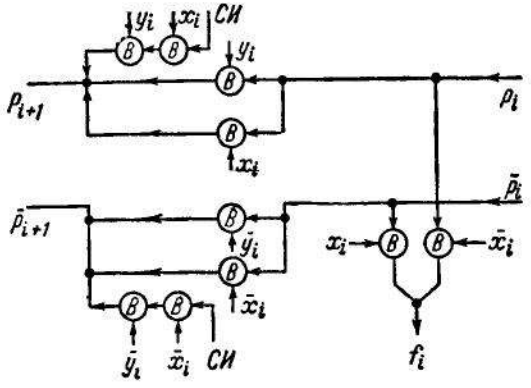

Рис. 21.

Через *СИ* на этом рисунке обозначены входные каналы схемы, подсоединенные к генератору синхронизирующих импульсов. С функциональной точки зрения это означает, что на входы *СИ* подается сигнал, тождественно равный единице.

Несмотря на то, что построенная схема правильна с чисто логической точки зрения, она, тем не менее, совершенно неудовлетворительна с точки зрения элементарных требований надежности. Дело не только в том, что при прохождении через импульсно-потенциальные вентили обычной конструкции (например, диодно-трансформатор-

ные) импульсы сильно ослабляются и требуют поэтому последующего усиления, дело также в том, что при выбранной схеме сумматора имеется опасность неправильного срабатывания триггеров за счет того, что по каналам f_i в течение одного такта работы схемы может прийти несколько сигналов.

Действительно, если $x_i=y_i=p_i=1$, то на выход p_{i+1} схемы, изображенной на рис. 21, приходят два различных сигнала: от входного канала p_i (через параллельно соединенные вентили x_i и y_i) и от входного канала *СИ* (через последовательно соединенные вентили x_i и y_i). При достаточно большом числе разрядов в сумматоре сигнал, приходящий со стороны входа p_i, может иметь значительную задержку во времени по сравнению с сигналом, поступающим непосредственно от генератора *СИ* через вентили x_i, y_i; если, например,

$$y_1=x_1=1,\ y_2=y_3=\ldots=y_{i-1}=1,\ x_2=x_3=\ldots=x_{i-1}=0,$$

то сигнал, приходящий на выход p_{i+1} со стороны входа p_i, приходит через i последовательно соединенных вентилей $y_1, y_2, \ldots, y_i$, в то время как сигнал от *СИ* в i-м разряде (при $x_i=y_i=1$) попадает на выход p_{i+1}, проходя лишь через два вентиля x_i и y_i. В результате на выход f_{i+1} ($i+1$)-го разряда может прийти не один импульс, а два, вызвав двойной переброс триггера y_{i+1}. Может случиться также, что эти два импульса сольются в один импульс неправильной формы, который вообще не перебросит триггер y_{i+1}.

Из сказанного ясно, что для достижения бóльшей надежности схема, изображенная на рис. 21, должна быть переделана так, чтобы не допускать поступления в одну и ту же точку схемы импульсов, прошедших через цепи, которые значительно отличаются друг от друга по длине. Этой цели можно достичь двумя различными путями: во-первых, для функций p_{i+1} и $\bar{p}_{i+1}$ вместо представлений (2) и (3) можно вводить следующие представления:

$$p_{i+1}=x_iy_i\vee\overline{x_iy_i}\,(x_ip_i\vee y_ip_i)=x_iy_i\vee(\bar{x}_i\vee\bar{y}_i)(x_i\vee y_i)p_i, \quad (4)$$

$$\bar{p}_{i+1}=\bar{x}_i\,\bar{y}_i\vee\overline{\bar{x}_i\,\bar{y}_i}\,(\bar{x}_i\,\bar{p}_i\vee\bar{y}_i\,\bar{p}_i)=$$
$$=\bar{x}_i\bar{y}_i\vee(x_i\vee y_i)(\bar{x}_i\vee\bar{y}_i)\bar{p}_i; \quad (5)$$

во-вторых, для этих функций можно вводить и представления такого рода:

$$p_{i+1} = x_i y_i \bar{p}_i \vee x_i p_i \vee y_i p_i = (x_i \vee y_i)\, p_i \vee x_i y_i \bar{p}_i, \qquad (6)$$

$$\bar{p}_{i+1} = \bar{x}_i \bar{y}_i p_i \vee \bar{x}_i \bar{p}_i \vee \bar{y}_i \bar{p}_i = (\bar{x}_i \vee \bar{y}_i)\, \bar{p}_i \vee \bar{x}_i \bar{y}_i p_i. \qquad (7)$$

Представления (4) и (5) запрещают прохождение сигналов на выходы p_{i+1}, $\bar{p}_{i+1}$ со стороны входов p_i и $\bar{p}_i$, если на них поступают сигналы со стороны входов *СИ* i-го разряда. Представления (6) и (7), наоборот, запрещают прохождение сигналов на выходы p_{i+1}, $\bar{p}_{i+1}$ со стороны входов *СИ* i-го разряда, если на них поступают сигналы со стороны входов p_i и $\bar{p}_i$.

Схема i-го разряда (при $1<i<n$) сумматора, соответствующая представлениям (6) и (7), изображена на рис. 22.

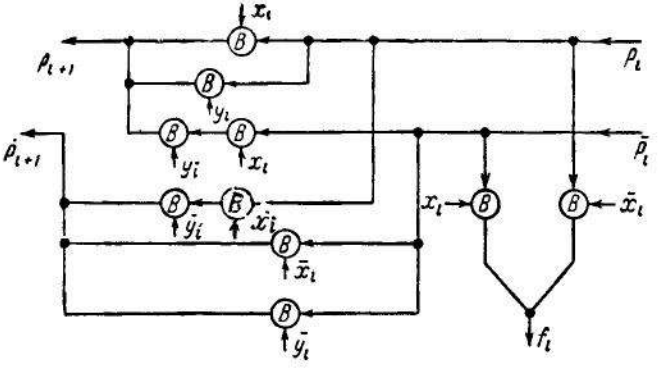

Рис. 22.

Эта схема столь же экономна с точки зрения числа вентилей, используемых в одном разряде, что и схема, изображенная на рис. 21 (10 вентилей на разряд), но в отличие от последней имеет то свойство, что в каждую точку схемы сигнал (импульс) может попасть через цепочку вентилей, имеющую строго определенную длину (зависящую от кодов $x_n x_{n-1} \ldots x_1$ и $y_n y_{n-1} \ldots y_1$). В каждом разряде импульс проходит по одному из путей $p_i \to p_{i+1}$, $\bar{p}_i \to \bar{p}_{i+1}$

(через вентильную цепь длины 1), либо по одному из путей $p_l \to \bar{p}_{l+1}$, $\bar{p}_l \to p_{l+1}$ (через вентильную цепь длины 2).

Поскольку $p_1 = 0$, то первый разряд сумматора может быть упрощен. Может быть существенно сокращен также последний (n-й) разряд. Это можно сделать за счет отбрасывания цепей, формирующих сигналы p_{n+1} и $\bar{p}_{n+1}$. Соответствующие упрощенные схемы изображены на рис. 23.

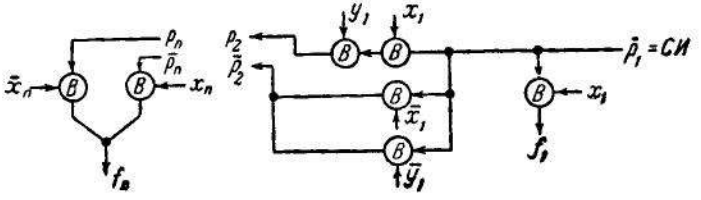

Рис. 23.

С учетом указанных упрощений весь n-разрядный сумматор требует для своего построения $10(n-2)+5+2=10n-13 (n \geqslant 2)$ вентилей. Разумеется, при его построении необходимы также усилительные элементы. Обычно в схемах такого рода импульсный усилитель ставится после каждого вентиля. Коэффициент разветвления описанной схемы не превосходит 3 для каждого из двух выходов триггера x_l и 2 для каждого из двух выходов триггера y_l.

Задержку импульсного сигнала в вентильных схемах принимают обычно равной сумме задержек в последовательно проходимых этим сигналом вентилях и усилительных элементах. Считая, что задержка сигнала одним вентилем, снабженным усилителем на выходе, равна τ, мы приходим к выводу, что максимальная задержка, которую может претерпеть импульс *СИ* на выходе f_n последнего разряда описанной схемы n-разрядного сумматора (на входе триггера y_n), равна $(2n-1)\tau$. Такая задержка будет иметь место при сложении двух чисел вида 010101...01. При этом в каждом разряде импульс будет переходить из цепи переноса в цепь отрицания переноса, либо наоборот.

Описанный сумматор осуществляет сложение в течение одного такта. В ряде случаев оказывается целесообр-

разным применять двухтактные сумматоры. Такие сумматоры на первом такте обычно осуществляют поразрядное сложение чисел, а во втором такте реализуют переносы. Возможны, разумеется, и другие варианты.

Пример 3. Построить импульсную схему на функциональных элементах с задержкой, реализующих функцию $x\bar{y}$, для автомата Мура A, заданного отмеченной таблицей переходов VI. 9.

Таблица VI. 9

	001
	012
0	101
1	212

Таблица VI. 10

	00	01	10
0	01	00	01
1	10	01	10

Решение. Закодируем состояния автомата следующим образом: 0=00, 1=01, 2=10. В таком случае физическая таблица переходов и совпадающая с ней, в силу теоремы 8. 1 гл. III, таблица возбуждений примет вид, представленный в табл. VI. 10.

Обозначая буквами x и y состояния двух элементов памяти автомата (элементов задержки), буквой a — входной сигнал, буквой z — выходной сигнал и предполагая, что f_1 и f_2 суть функции возбуждения элементов x и y, мы придем к картам Карнау для функций z, f_1, f_2, изображенным соответственно на табл. VI. 11, VI. 12 и VI. 13.

Таблица VI. 11

xy \ a	0	1
00	0	0
01	0	0
11	—	—
10	1	1

Таблица VI. 12

xy \ a	0	1
00	0	1
01	0	0
11	—	—
10	0	1

Таблица VI. 13

xy \ a	0	1
00	1	0
01	0	1
11	—	—
10	1	0

Отсюда нетрудно получить следующие представления для функций z, f_1 и f_2:

$$z = x, \quad f_1 = \bar{y}a, \quad f_2 = \overline{\bar{y}\,\bar{a}} \vee ya.$$

Предположим, что имеет место естественное разделение сигналов. В таком случае функцию f_2 можно получить следующим образом через функцию $x\bar{y}$ и константу 1:

$$f_2 = (1 \cdot \bar{y})\,\bar{a} \vee a\,\overline{(1 \cdot \bar{y})}.$$

С помощью полученного представления функция f_2 реализуется с задержкой на два такта. С той же задержкой необходимо реализовать также функцию f_1. Это нетрудно сделать, используя представление $f_1 = (a\bar{y})\,\bar{0}$.

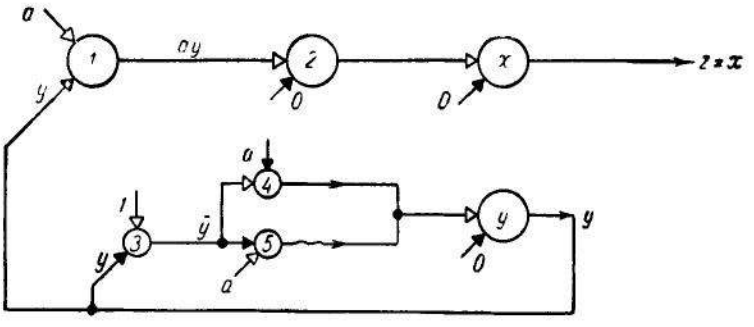

Рис. 24.

Считая, что на изолированном узле возникает нулевой сигнал, а единичный сигнал имеется в нашем распоряжении в качестве выходного сигнала тактирующего генератора, мы приходим к схеме, изображенной на рис. 24.

На этом рисунке стрелкам с зачерненными концами соответствуют запрещающие входы элементов $x\bar{y}$ (т. е. входы для сигнала y в случае функции $x\bar{y}$), а стрелкам с незачерненными концами — обычные (разрешающие) входы (то есть входы для сигнала x в случае функции $\bar{xy}$).

Элементы, обозначенные буквами x и y, являются запоминающими, а элементы, обозначенные цифрами *1*, *2*, *3*, *4*, *5*, выполняют роль логических элементов.

Поскольку сигналы в цепи обратной связи автомата совершают полный цикл за три такта, внешний такт работы автомата (то есть минимальное время, необходимое для изменения входных и выходных сигналов) равен трем его внутренним тактам. С точки зрения внутренней тактировки автомата все его внешние сигналы (входные и выходные) представляются словами, в которых каждая буква выписывается трижды.

Заметим, что коэффициент разветвления в схеме, изображенной на рис. 24, не превосходит 2, то есть не превышает минимального значения, при котором еще возможно построение схем достаточно большой сложности.

§ 3. Проблема синтеза надежных схем из ненадежных элементов

Все методы синтеза схем цифровых автоматов, которые были описаны до сих пор, исходят из задачи наиболее экономного расходования элементов (как запоминающих, так и логических). Ясно, что достижение максимальной экономии элементов предъявляет высокие требования к надежности работы каждого элемента. По самому определению наиболее экономной схемы она должна обладать тем свойством, что выход из строя любого ее элемента вызывает выход из строя всей схемы в целом. Действительно, наличие элементов, которые можно исключить, не нарушая работы схемы, означает, что эта схема не является минимальной с точки зрения числа элементов, израсходованных для ее построения.

Мы будем различать два вида неисправностей элементов цифровых автоматов. Неисправность первого вида, называемая *отказом*, состоит в том, что начиная с некоторого момента времени и во все последующие моменты времени элемент перестает функционировать или функционирует неверно. Чтобы снова заставить правильно функционировать отказавший элемент, его необходимо отремонтировать или заменить новым, исправным элементом. Второй вид неисправности — это мгновенная самоустраняющаяся неисправность, называемая обычно *сбоем*. Наступление сбоя означает, что элемент не сработал или неправильно сработал в течение одного такта работы автомата,

но отсюда никоим образом не следует, что возникшая таким образом мгновенная неисправность сохранится на протяжении последующих тактов.

Мы будем предполагать, что сбои элемента в различные моменты времени наступают независимо друг от друга. Это предположение достаточно хорошо оправдывается на практике. Из этого предположения получится, что достаточно хорошую характеристику надежности работы элементов до их отказа может дать вероятность сбоя каждого элемента. Мы будем исходить из допущения, что эти вероятности остаются постоянными в течение достаточно большого промежутка времени. При этом допущении вероятность сбоя элемента может быть определена как математическое ожидание отношения числа тактов, в течение которых имел место сбой рассматриваемого элемента, к общему числу тактов N, в течение которых производилась регистрация сбоев (при $N \to \infty$).

Для двоичных элементов принято различать *сбой в нуле* (состоящий в том, что элемент выдает единичный сигнал вместо нулевого) и *сбой в единице* (состоящий в том, что элемент выдает нулевой сигнал вместо единичного). Соответственно этому для каждого элемента определяются вероятности его сбоя в нуле и в единице.

Предположим, что схема из n элементов сбивается (неверно функционирует) всякий раз, когда происходит сбой хотя бы одного составляющего ее элемента (это будет обычно иметь место, как было отмечено выше, для схем с минимальным числом элементов). Если все элементы имеют одну и ту же вероятность сбоя p, то вероятность сбоя всей схемы в целом будет равна, очевидно, $1-(1-p)^n \overset{\text{Df}}{=} q_n$. Нетрудно понять, что даже для высоконадежных элементов (отличающихся тем, что их вероятность p очень мала) при достаточно большом числе n элементов в схеме автомата вероятность сбоя этого автомата (т. е. величина q_n) будет весьма велика; это означает, что автомат будет работать чрезвычайно ненадежно.

Современные сложные электронные цифровые автоматы состоят из многих тысяч логических и запоминающих элементов, требования же к надежности их работы чрезвычайно высоки (вероятность сбоя автомата должна иметь порядок не выше одной миллиардной или даже еще меньше).

Это накладывает еще более высокие требования на надежность элементов, вероятность сбоя которых не должна в большинстве случаев превосходить 10^{-12}—10^{-15}. В ряде случаев требования к надежности элементов оказываются практически невыполнимыми, и тогда возникает проблема построения высоконадежных схем из относительно ненадежных элементов.

Методы, на которых основано решение этой проблемы, сводятся к дублированию одних элементов другими, к многократному повторению, к автоматическому контролю и к исправлению работы отдельных участков схемы или всей схемы в целом. Несмотря на общность природы всех этих методов, конкретные особенности их применения в значительной мере определяются спецификой элементов, из которых строится схема. Мы опишем ниже более подробно случай вентильных схем, рассмотренный (на примере релейно-контактных схем) К. Шенноном и Э. Муром[1]).

Предположим, что выполняются следующие условия:

1. *Сбои одного и того же вентиля в различные промежутки времени независимы друг от друга.*

2. *Сбои различных вентилей схемы независимы друг от друга.*

3. *Вероятности сбоев в нуле и в единице одинаковы для всех вентилей, не зависят от выбора схемы и не меняются с течением времени (первую из этих вероятностей мы будем обозначать буквой a, вторую буквой b).*

Как мы уже отмечали выше, первое условие оказывается достаточно близким к действительному положению дела с реальными автоматами[2]). Что касается второго и третьего условия, то их применимость к реальным схемам цифровых автоматов может быть оправдана лишь с некоторым приближением. Тем не менее мы в дальнейших рассуждениях будем предполагать выполненными все три условия.

Поставим теперь задачу построения таких схем, которые выполняют работу одного вентиля, но обладают более

---

[1]) E. F. Moore, C. E. Shannon, Reliable circuits using less reliable relays. J. Franklin Inst., v. 262, № 3, 1956, p. 191—208, № 4, p. 281—297 (русский перевод: Кибернетический сборник № 1, ИЛ, М., 1960, стр. 109—148).

[2]) Старение элементов здесь в расчет не принимается.

высокой надежностью, чем отдельный вентиль. С этой целью рассмотрим произвольную вентильную схему с одним входным и с одним выходным полюсом, составленную из вентилей, управляемых одним и тем же сигналом x. Любую такую схему мы условимся называть *специальной вентильной схемой*.

Ясно, что специальная вентильная схема является аналогом обыкновенного вентиля и может замещать любой из вентилей в произвольной вентильной схеме. Операцию замещения всех вентилей какой-либо вентильной схемы S специальной вентильной схемой P мы договоримся называть подстановкой схемы P в схему S. В результате подстановки специальной вентильной схемы в любую другую специальную вентильную схему снова возникает некоторая специальная вентильная схема.

Если выходной полюс специальной вентильной схемы S_1 присоединить к входному полюсу другой специальной вентильной схемы S_2, управляемой тем же самым сигналом x, что и схема S_1, то возникающая таким образом вентильная схема S также будет специальной вентильной схемой, в этом случае говорят, что схема S получена в результате *последовательного соединения* схем S_1 и S_2. Если произвести отождествление входных полюсов схем S_1 и S_2, а также их выходных полюсов, то полученная вентильная схема снова будет специальной, в этом случае говорят, что она получена в результате *параллельного соединения* схем S_1 и S_2.

Для каждой специальной вентильной схемы можно определить функцию $h(p)$, определяющую вероятность пропускания сигнала этой схемой (от входного полюса к выходному полюсу), зависящую от вероятности p пропускания (импульсного) сигнала любым наугад выбранным, вентилем схемы. Согласно принятым выше условиям, вероятность p не зависит от выбора вентиля и может принимать лишь два значения, а именно значение a при нулевом управляющем сигнале и значение $1-b$ при единичном управляющем сигнале.

Определенную таким образом функцию $h(p)$ мы условимся называть *функцией вероятности пропускания сигнала* соответствующей специальной вентильной схемы или, кратко, *h-функцией* этой схемы.

Справедливо следующее предложение [1]).

3. 1. *Функция $h(p)$ вероятности пропускания сигнала в произвольной специальной вентильной схеме S, состоящей из m вентилей, может быть представлена в виде*

$$h(p) = \sum_{n=0}^{m} A_n p^n (1-p)^{m-n},$$

где A_n обозначает число способов, которыми в схеме S может быть выбрано n вентилей так, чтобы схема пропускала сигнал от входного полюса к выходному при условии, что все выбранные вентили пропускают, а все остальные вентили схемы не пропускают (импульсный) сигнал.

Справедливость предложения 3. 1 вытекает из того, что в соответствии с принятыми выше условиями вероятность пропускания импульсов любыми n вентилями равна p^n, а вероятность непропускания импульсов остальными $m-n$ вентилями равна $(1-p)^{m-n}$.

Рассмотрим теперь две специальные вентильные схемы S_1 и S_2 с функциями вероятности пропускания сигнала (h-функциями), равными $h_1(p)$ и $h_2(p)$ соответственно. Предполагая, что эти схемы управляются одним и тем же сигналом x, нетрудно доказать, что h-функция схемы, полученной в результате последовательного соединения схем S_1 и S_2, будет равна $h_1(p)\,h_2(p)$. Действительно, полученная схема будет пропускать сигнал в том и только в том случае, когда сигнал пропускается как схемой S_1, так и схемой S_2; поэтому вероятность пропускания сигнала схемой S будет равна произведению вероятностей пропускания сигналов схемами S_1 и S_2.

При параллельном соединении схем S_1 и S_2 следует, очевидно, перемножать вероятности непропускания сигналов $1-h_1(p)$ и $1-h_2(p)$. Наконец, при подстановке схемы S_2 в схему S_1 роль вероятности p для схемы S_1 после этой подстановки будет играть вероятность $h_2(p)$, так что вероятность пропускания сигнала схемой, построенной таким образом, будет равна $h_1(h_2(p))$.

Итак, мы доказали следующее предложение.

---

[1]) См работу Шеннона и Мура, указанную в примечании 1 на стр. 400.

3. 2. *Пусть S_1 и S_2 являются двумя произвольными специальными вентильными схемами, управляемыми одним и тем же сигналом x, а $h_1(p)$ и $h_2(p)$ представляют собой h-функции (функции вероятности пропускания сигнала) этих схем. Тогда h-функции схем, полученных из схем S_1 и S_2 в результате их последовательного соединения, параллельного соединения и подстановки схемы S_2 в схему S_1 равны соответственно $h_1(p)\,h_2(p)$, $1-(1-h_1(p))\,(1-h_2(p))$ и $h_1(h_2(p))$.*

Если вероятности сбоев в нуле и в единице для вентилей достаточно малы, то можно строить специальные вентильные схемы, имитирующие вентиль сколь угодно высокой надежности, с помощью следующей конструкции. Следует построить специальную вентильную схему S, для которой h-функция $h(p)$ была бы меньше p в некоторой окрестности нуля и больше p в некоторой окрестности единицы. Осуществляя многократную подстановку такой схемы в себя, мы можем получить, очевидно, схему, имитирующую вентиль сколь угодно высокой надежности, если вероятности a и $1-b$ исходных вентилей находились в указанных окрестностях нуля и единицы. Схему S мы будем называть *базовой схемой*, а процесс многократной подстановки этой схемы в себя, при котором возникают схемы с h-функциями $h(h(\ldots h(p)))$ — *итерацией* этой схемы.

В качестве базовых схем можно выбрать схемы $S_1=S_1(l, w)$ и $S_2=S_2(l, w)$, получаемые следующим образом. Схема S_1 представляет собой параллельное соединение w групп, каждая из которых состоит из l последовательно соединенных друг с другом вентилей. Схема S_2, двойственная схеме S_1, представляет собой последовательное соединение l групп, каждая из которых состоит из w параллельно соединенных друг с другом вентилей. Из теоремы 3. 2 непосредственно вытекает следующее предложение.

3. 3. *Функции вероятности пропускания сигналов схем $S_1(l, w)$ и $S_2(l, w)$ равны соответственно $1-(1-p^l)^w$ и $(1-(1-p)^w)^l$.*

В окрестности нуля h-функции схем S_1 и S_2 приближенно равны $w\,p^l$ и $w^l p^l$, то есть имеют порядок p^l. В окрестности единицы эти функции допускают приближенные представления вида $1-l^w(1-p)^w$ и $1-l(1-p)^w$. Изучение

этих представлений показывает, что как схема S_1, так и схема S_2 могут быть использованы в качестве базовых схем при условии достаточно высокой исходной надежности вентилей.

К. Шеннон и Э. Мур в работе «Надежные схемы из ненадежных реле» [1]) используют в качестве базовой схемы схему из двусторонних вентилей, изображенную на рис. 25. Мы будем называть эту схему *основной базовой*

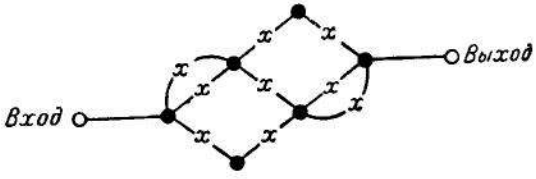

Рис. 25.

схемой; h-функция этой схемы может быть представлена (на основании теоремы 3. 1) в виде

$$h\ p) = 8p^3 - 6p^4 - 6p^5 + 12p^7 - 9p^8 + 2p^9 = \\ = 8p^3 - p^4\,[6\,(1-p^3) + 6p\,(1-p^2) + 7p^4 + \\ + 2p^4\,(1-p)]. \qquad (1)$$

Поскольку входящее в формулу (1) выражение в квадратных скобках положительно при $0 < p < 1$, то имеет место оценка:

$$h\,(p) \leqslant 8p^3, \qquad (2)$$

справедливая для всех возможных значений $p\,(0 \leqslant p \leqslant 1)$. Обозначая через $h\,(p)$ функцию схемы, полученной в результате k-кратной итерации основной базовой схемы S, мы сможем вывести из неравенства (2) для этой функции следующую очевидную оценку:

$$h^{(k)}\,(p) \leqslant 8\,(8\,(\ldots(8\,(8p^3)^3)^3\ldots)^3 = 8^{1+3+3^2+\ldots+3^{k-1}}\,p^{3^k}.$$

Поскольку $1 + 3 + 3^2 + \ldots + 3^{k-1} = \dfrac{3^k - 1}{2}$, то мы

---

[1]) См. русский перевод в «Кибернетическом сборнике», № 1, ИЛ, М., 1960, стр. 109—148.

получаем оценку

$$h^{(k)}(p) \leqslant \frac{1}{\sqrt{8}}(\sqrt{8}p)^{3^k} \quad (0<p<1). \tag{3}$$

Нетрудно проверить, что h-функция $h(p)$ основной базовой схемы удовлетворяет соотношению $1-h(p)=h(1-p)$. Из этого соотношения и неравенства (2) мы получаем неравенство $1-h(p)\leqslant 8(1-p)^3$, а из последнего неравенства точно таким же приемом, какой был применен при выводе соотношения (3), мы получаем неравенство

$$1-h^{(k)}(p) \leqslant \frac{1}{\sqrt{8}}(\sqrt{8}(1-p))^{3^k} \quad (0<p<1). \tag{4}$$

Из неравенств (3) и (4) без труда выводится оценка для числа вентилей, с помощью которых можно построить специальную вентильную схему, обладающую заданной степенью надежности. Действительно, предположим, что мы располагаем вентилями, вероятность сбоя которых в нуле равна α_0, а в единице — β_0. Допустим, что нам необходимо построить специальную вентильную схему (заменяющую один вентиль), имеющую вероятность сбоя в нуле не более чем α, а вероятность сбоя в единице не более чем β. Используем в качестве такой схемы k-кратную итерацию основной базовой схемы. Тогда, поскольку вероятности сбоя в нуле и в единице для этой схемы не должны превосходить α и β, мы можем найти необходимое число итераций k как наименьшее целое число, для которого одновременно удовлетворяются два неравенства:

$$h^{(k)}(\alpha_0) \leqslant \alpha, \quad 1-h^{(k)}(1-\beta_0) \leqslant \beta. \tag{5}$$

Используя соотношения (3) и (4), мы можем заменить эти неравенства следующими неравенствами:

$$\frac{1}{\sqrt{8}}(\sqrt{8}\,\alpha_0)^{3^k} \leqslant \alpha, \qquad \frac{1}{\sqrt{8}}(\sqrt{8}\,\beta_0)^{3^k} \leqslant \beta. \tag{6}$$

Неравенства (6) могут быть решены относительно k при $\alpha_0 < \frac{1}{\sqrt{8}}$ и $\beta_0 < \frac{1}{\sqrt{8}}$, то есть в том случае, если исходные вентили, использованные для построения схемы, обладают достаточной (хотя, может быть, и не слишком

высокой) надежностью (то есть вероятность их сбоя как в нуле, так и в единице не превосходит $\frac{1}{\sqrt{8}} \approx 0{,}353$).

Чтобы не производить двойного логарифмирования при определении k, воспользуемся неравенством (6) сразу для определения числа вентилей N в специальной вентильной схеме, обеспечивающей заданную надежность (вероятность сбоя в нуле не выше чем α, а вероятность сбоя в единице не выше чем β). Число вентилей в основной базовой схеме равно 9 и увеличивается в 9 раз при каждой очередной итерации. Таким образом, после k-кратной итерации мы будем иметь 9^k вентилей. Обозначая это число буквой N, мы приходим к выводу, что $3^k = \sqrt{N}$. Из неравенств (6) теперь можно вывести неравенства:

$$N \leqslant \left(\frac{\log(\alpha \sqrt{8})}{\log(\alpha_0 \sqrt{8})} \right)^2, \quad N \leqslant \left(\frac{\log(\beta \sqrt{8})}{\log(\beta_0 \sqrt{8})} \right)^2. \tag{7}$$

Эти неравенства, однако, не могут быть непосредственно использованы для оценки необходимого числа контактов, поскольку при принятых нами допущениях это число должно обязательно быть целой степенью девятки. Необходимо поэтому произвести округления правых частей неравенств (7) до ближайших больших целых степеней девяток. В качестве более грубых оценок можно применить, очевидно, простое умножение правых частей этих неравенств на 9. В результате мы приходим к оценкам:

$$N \leqslant 9 \left(\frac{\log(\alpha \sqrt{8})}{\log(\alpha_0 \sqrt{8})} \right)^2, \quad N \leqslant 9 \left(\frac{\log(\beta \sqrt{8})}{\log(\beta_0 \sqrt{8})} \right)^2. \tag{8}$$

Само собою разумеется, что неравенства (7) и (8) можно (и нужно) применять лишь в том случае, когда $0 < \alpha < \alpha_0 < \frac{1}{\sqrt{8}}$ и $0 < \beta < \beta_0 < \frac{1}{\sqrt{8}}$.

Из проведенных нами рассуждений может быть сделан следующий вывод.

3.4. *Если вероятности сбоев вентилей в нуле и в единице не превосходят* $\frac{1}{\sqrt{8}} \approx 0{,}353$, *то из этих вентилей может быть построена специальная вентильная схема, заменяющая собою один вентиль и имеющая сколь угодно высокую наперед заданную надежность.*

Заметим, что количественное ограничение, накладываемое теоремой 3.4 на надежность исходных вентилей, не является строго обязательным. К. Шеннон и Э. Мур в упоминавшейся выше работе показали, что можно построить вентильную схему сколь угодно высокой надежности, если вероятности сбоев в нуле и в единице (α_0 и β_0) для исходных вентилей удовлетворяют соотношению $0 \leqslant \alpha_0 < 1 - \beta_0 \leqslant 1$. Ясно, что такое условие является не только достаточным, но и необходимым для возможности построения правильно и надежно функционирующей вентильной схемы.

Построение надежных схем из ненадежных элементов не является, разумеется, исключительной привилегией вентильных схем. Такие построения (основанные на аналогичных идеях) можно осуществлять для схем, построенных из любых элементов. В литературе, кроме вентильных схем, исследован также случай схем, составленных из функциональных элементов задержки специального вида [1]).

Необходимо отметить, однако, что возможность увеличения надежности схем за счет описанных приемов не следует преувеличивать. Как правило, на практике, следуя по пути Шеннона — Мура — Неймана, можно добиться существенного,— но не беспредельного — увеличения надежности схем. Причина этого явления заключается в том, что описанные методы увеличения надежности не учитывают влияния многих факторов, оказывающих существенное влияние на надежность, например, не учитывают увеличения вероятности сбоя элементов при увеличении коэффициента разветвления схемы. Нетрудно видеть, что метод Шеннона — Мура приводит к значительному увеличению коэффициента разветвления: входной сигнал x, управляющий в первоначальной схеме n вентилями, после k-кратной подстановки в эту схему основной базовой схемы должен управлять уже $n \cdot 9^k$ вентилями. Ясно, что подобное увеличение коэффициента разветвления не может не сказаться на надежности схемы.

[1]) См. Дж. Нейман, Вероятностная логика и синтез надежных организмов из ненадежных компонент. Сб. «Автоматы», ИЛ, М., 1956, стр. 68—139.

Среди других неучтенных факторов необходимо отметить также ненадежность в соединениях элементов между собой, взаимозависимость сбоев в различных элементах и т. д.

Значительное увеличение надежности на основе описанных выше методов требует значительного увеличения числа элементов. Например, для увеличения надежности работы вентиля с $1-10^{-6}$ до $1-10^{-10}$ (то есть для уменьшения вероятности сбоев в нуле и в единице с 10^{-6} до 10^{-10}) необходимо, согласно неравенствам (7), заменить его схемой, содержащей $K\left(\frac{\log(10^{-10}\sqrt{8})}{\log(10^{-6}\sqrt{8})}\right)^2 = K\left(\frac{10-0{,}45}{6-0{,}45}\right)^2 = 2{,}95K = 9$ вентилей (здесь K — коэффициент, с помощью которого осуществляется округление до ближайшей степени девятки). Таким образом, в рассматриваемом случае будет иметь место девятикратное усложнение схемы. Если при той же надежности вентиля поставить задачу увеличить надежность схемы до величины $1-10^{-20}$, то для этого потребуется усложнение схемы уже в 81 раз. Можно, правда, несколько уменьшить размеры требуемого увеличения сложности схем, если в рассмотренных примерах воспользоваться базовой схемой, состоящей из меньшего числа вентилей, чем основная базовая схема. Тем не менее это увеличение останется еще весьма значительным

В реально существующих схемах цифровых автоматов употребляются обычно другие методы увеличения надежности работы схем. Наиболее употребительным из этих методов является простое дублирование схемы автомата или некоторых ее наиболее важных частей. Самое простое решение вопроса заключается в том, чтобы поставить рядом несколько совершенно идентичных между собой автоматов и считать правильными лишь те результаты, которые выдаются одновременно всеми или определенной частью этих автоматов.

Такое дублирование можно, разумеется, осуществить не пространственным, а временны́м путем, то есть, иными словами, производить многократную прогонку входного слова (условия задачи) через один и тот же автомат и принимать в качестве верных лишь достаточно часто повторяющиеся результаты. При условии независимости распределения сбоев во времени такой метод, очевидно, вполне эк-

вивалентен методу физического (пространственного) дублирования автоматов.

С целью увеличения надежности цифровых автоматов употребляются также специальные методы кодирования, позволяющие автоматически обнаруживать — и даже исправлять — ошибки, получающиеся в результате сбоев.

В большинстве современных цифровых автоматов информация естественно делится на слова (в двоичном алфавите), имеющие вполне определенную длину. Добавляя в коды этих слов новые разряды, можно использовать их для хранения информации, позволяющей обнаруживать и исправлять ошибки в коде, возникающие в результате сбоев. Подобные удлиненные коды являются избыточными, то есть содержат бо́льшее число разрядов, чем то, которое действительно необходимо для представления имеющейся информации.

Различают *избыточные коды с обнаружением ошибок и избыточные коды с исправлением ошибок*. Коды с исправлением ошибок называются также *самокорректирующимися кодами*.

Простейший избыточный код с обнаружением одной ошибки (ошибки в одном разряде) может быть получен следующим образом. К обычному двоичному коду p длины n прибавляется дополнительный двоичный разряд x, в который помещается нуль или единица так, чтобы сумма по модулю 2 всех разрядов полученного таким образом кода $q=px$ длины $n+1$ была равна нулю. Неравенство нулю этой суммы будет свидетельствовать, очевидно, о том, что имеется ошибка в нечетном числе разрядов кода q (не исключая и дополнительного разряда x). Поскольку одновременное появление нескольких ошибок (при предположении о независимости сбоев) гораздо менее вероятно, чем появление одной ошибки, то указанная ситуация (неравенство нулю поразрядной суммы) будет практически возникать в тех и только в тех случаях, когда код ошибочен.

Более интересными являются самокорректирующиеся коды, позволяющие не только обнаруживать, но и исправлять ошибки, возникающие в результате сбоев. Рассмотрим в качестве примера простейший код с

исправлением одной ошибки, называемый обычно *кодом Хэмминга* [1]).

Кодом Хэмминга (с исправлением одной ошибки) называется двоичный код $x_m x_{m-1} \ldots x_1$, в котором фиксирована некоторая нумерация разрядов, а суммы $x_{i_1}+x_{i_2}+\ldots+x_{i_r}$ (по модулю два) некоторых (определяемых ниже) групп разрядов равны нулю.

Мы примем нумерацию разрядов кода справа налево, а группы разрядов, о которых идет речь в определении кода Хэмминга, определим следующим образом: в i-ю группу отнесем все те разряды, номера которых в двоичном представлении имеют единицу в i-м разряде (считая справа налево). Таким образом, первую группу составят разряды с номерами 1, 3, 5, 7,..., вторую группу — разряды с номерами 2, 3, 6, 7,..., третью группу — разряды с номерами 4, 5, 6, 7,... и т. д.

Ясно, что наименьший номер, попадающий в i-ю группу, будет всегда равен 2^{i-1}. Следовательно, если число 2^{i-1} превышает число разрядов в коде, то i-я группа разрядов этого кода будет пустой (не будет содержать ни одного разряда). В последующем мы будем исключать из рассмотрения пустые группы и будем иметь дело исключительно с непустыми группами разрядов. Сумму по модулю два всех разрядов i-й группы рассматриваемого кода $x = x_m x_{m-1} \ldots x_1$ мы условимся обозначать через $\sigma_i(x)$.

Рассмотрим теперь произвольный n-разрядный двоичный код $x_n x_{n-1} \ldots x_1$. Добавим к нему p новых (двоичных) разрядов $x_{n+p} x_{n+p-1} \ldots x_{n+1}$, выбирая в качестве p наименьшее целое положительное число, удовлетворяющее неравенству

$$2^p > n + p + 1. \tag{9}$$

Ясно, что такое неравенство всегда имеет решение. Потребуем теперь, чтобы построенный нами $(n+p)$-разрядный код был кодом Хэмминга. Это означает, что значения $x_{n+1}, \ldots, x_{n+p}$ добавляемых разрядов должны быть выбраны так, чтобы удовлетворялись соотношениям:

$$\sigma_1(x) = 0, \quad \sigma_2(x) = 0, \quad \ldots, \quad \sigma_p(x) = 0. \tag{10}$$

---

[1]) A. W. Hamming, Error detecting and error correcting codes Bell System. Tech. J., v. 29, № 2, 1950, p. 147—160.

Из неравенства (9) и сделанного выше замечания относительно пустых групп разрядов вытекает, что p-я группа разрядов кода x будет непустой, а $(p+1)$-я и все последующие группы разрядов непременно пусты. Поэтому в системе (10) будет ровно p уравнений. Нетрудно показать, что из этих уравнений всегда (и притом однозначным образом) могут быть определены значения всех добавляемых разрядов $x_{n+1},\ldots,x_{n+p}$. Ввиду роли, которую играют эти разряды, мы условимся называть их вспомогательными, а разряды $x_n,\ldots,x_1$ — основными разрядами кода x.

Суммы (по модулю два) $\sigma_1=\sigma_1(x),\ldots,\sigma_p=\sigma_p(x)$ мы назовем *контрольными суммами* кода x, а двоичный код $\sigma_p\sigma_{p-1}\ldots\sigma_1$, составленный из этих кодов,— *контрольным кодом* кода x.

Для построенного нами кода Хэмминга с исправлением одной ошибки справедливо следующее предложение.

3.5. *Если в коде Хэмминга (с исправлением одной ошибки) возникла единственная ошибка в каком-либо разряде (безразлично, в основном или во вспомогательном), то контрольный код полученного ошибочного кода будет совпадать с двоичным представлением номера разряда, в котором возникла ошибка.*

В самом деле, ошибка в k-м разряде кода Хэмминга превратит в единицы все те и только те контрольные суммы σ_i, которые содержат в своем составе k-й разряд кода x, остальные же контрольные суммы останутся нулями. В силу принятого выше определения групп разрядов i-я группа разрядов, составляющая контрольную сумму σ_i, тогда и только тогда содержит k-й разряд, когда в двоичном представлении числа k на i-м месте стоит единица. Из сопоставления этих двух утверждений немедленно вытекает справедливость теоремы 3.5.

Благодаря тому, что коды, с которыми мы имеем дело, являются двоичными, знание разряда, в котором возникла ошибка, позволяет исправить эту ошибку. Для этой цели достаточно заменить цифру (0 или 1), стоящую в указанном разряде, на противоположную. При построении схем автоматов, оперирующих с кодами Хэмминга, в них включают специальные *корректирующие схемы*,

которые производят подсчет контрольных сумм и автоматическое исправление возникшей (одиночной) ошибки в соответствии с только что описанным правилом. В ряде случаев не делают специальных корректирующих схем, а для исправления ошибок в кодах вводят дополнительные такты работы автомата. Метод исправления при этом остается прежним (через вычисление контрольных сумм).

В связи с описанными возможностями автоматического исправления (одиночной) ошибки в кодах Хэмминга, эти коды принято называть самокорректирующимися.

Рассмотрим процесс построения кода Хэмминга и исправления возникающей (одиночной) ошибки на примере. Предположим, что первоначальный код x имел вид $x=101100$. Поскольку длина этого кода равна 6, из неравенства (9) мы находим, что число вспомогательных разрядов должно быть наименьшим натуральным числом, удовлетворяющим неравенству $2^r \geqslant 7+p$. Отсюда легко находится значение $p=4$. Таким образом, для построения требуемого кода необходимо иметь четыре вспомогательных разряда. Обозначим эти разряды через $y_4y_3y_2y_1$. Тогда код Хэмминга, соответствующий коду x, будет иметь вид $y=y_4y_3y_2y_1101100=y_4y_3y_2y_1x_6x_5x_4x_3x_2x_1$, где значения вспомогательных разрядов определяются из уравнений (сложение здесь понимается как сложение по модулю два):

$$\sigma_1(y)=x_1+x_3+x_5+y_1+y_3=y_1+y_3+1=0,$$
$$\sigma_2(y)=x_2+x_3+x_6+y_1+y_4=y_1+y_4=0,$$
$$\sigma_3(y)=x_4+x_5+x_6+y_1=y_1=0,$$
$$\sigma_4(y)=y_2+y_3+y_4=0.$$

Из этих уравнений находим решение:

$$y_1=y_4=0, \quad y_2=y_3=1.$$

Таким образом, искомый код Хэмминга будет иметь вид

$$y=0110101100.$$

Предположим теперь, что в результате ошибки в седьмом разряде этот код превратился в код $0111101100=y'$.

Найдем контрольные суммы:

$$\sigma_1(y') = 0+1+0+1+1 = 1,$$
$$\sigma_2(y') = 0+1+1+1+0 = 1,$$
$$\sigma_3(y') = 1+0+1+1 = 1,$$
$$\sigma_4(y') = 1+1+0 = 0.$$

Контрольный код $\sigma_4\sigma_3\sigma_2\sigma_1$ равен 0111. Поскольку 0111 есть не что иное, как двоичное представление числа 7, то в результате проверки обнаруживается, что ошибка возникла в седьмом разряде, что и имело место на самом деле.

Заметим, что благодаря произвольности порядка нумерации разрядов, совершенно безразлично, каким образом будут размещены вспомогательные разряды по отношению к основным. Можно, в частности, приписывать вспомогательные разряды не слева от основных, как было сделано выше, а справа.

Нетрудно произвести обобщение введенных кодов. С этой целью рассмотрим понятие *расстояния между двумя двоичными кодами равной длины*, определив его как суммарное число разрядов, в которых эти коды различаются между собой. Например, расстояние между кодами 0110 и 0011 равно, согласно этому определению, двум, а расстояние между кодами 010 и 011—единице. Систему двоичных кодов равной длины назовем *системой кодов ранга n*, если расстояние между любыми двумя различными кодами этой системы не меньше чем n, причем существует пара различных кодов, расстояние между которыми в точности равно n.

Если в автомате употребляется система кодов ранга 2, то она дает возможность, очевидно, обнаруживать одиночную ошибку; система кодов ранга 3 позволяет исправлять одиночную ошибку; система кодов ранга 4 дает возможность исправления одиночной и обнаружения двойной ошибки: при употреблении системы кодов ранга 5 можно исправлять двойную ошибку, и т. д.

ГЛАВА VII

АЛГОРИТМИЧЕСКАЯ СТРУКТУРА СОВРЕМЕННЫХ УНИВЕРСАЛЬНЫХ ЦИФРОВЫХ МАШИН

§ 1. Принцип программного управления. Блок-схема универсального программного автомата

Развитые в предыдущих главах методы синтеза схем автоматов с памятью оказываются пригодными в основном для синтеза автоматов с относительно небольшим числом состояний. В случае синтеза сложных автоматов применению этих методов предпосылается обычно еще один этап, называемый этапом *блочного синтеза*. Смысл этого этапа заключается в том, что на основе знания алгоритма, который должен быть реализован синтезируемым автоматом, схема автомата разбивается на ряд отдельных участков, называемых *блоками*, и производится определение условий работы этих блоков (то есть определение реализуемых ими алгоритмов, или логических функций).

Одновременно с этим обычно производится *циклирование* работы отдельных блоков; циклирование блоков состоит в том, что автоматное время для каждого блока разбивается на группы тактов. Условия работы блока определяются таким образом, чтобы они повторялись на каждой из этих групп. Более точно, реализуемое блоком отображение определяется лишь на тех входных словах, которые воспринимаются блоком в течение одного цикла. На протяжении каждого из следующих циклов блок будет реализовать то же самое отображение, что и в первом цикле, хотя и с другой, вообще говоря, входной информацией.

Поскольку разбиение на блоки и циклирование работы автомата осуществляется на основе свойств реализуемого

автоматом алгоритма, этап блочного синтеза (вместе с циклированием) автомата называют обычно также этапом *алгоритмического синтеза*. С этапом блочного синтеза мы фактически уже встречались выше, мы имели с ним дело, например, при синтезе схемы параллельного сумматора в предыдущей главе (§ 2, пример 2). В этом примере блоками являются схемы отдельных разрядов сумматора, а рабочий цикл состоит из одного такта, поскольку в каждом новом такте схема осуществляет новое сложение. Как уже отмечалось выше, при создании надлежащих условий для работы параллельных сумматоров чаще всего употребляется рабочий цикл, состоящий из двух тактов: такта поразрядного сложения и такта реализации переносов.

При синтезе особо сложных автоматов этап блочного синтеза может быть многоступенчатым. Это значит, что после первого разбиения на блоки и циклирования к каждому из построенных блоков (или к некоторым из них) снова применяется прием разбиения на блоки и циклирования и т. д. В соответствии с этим различаются блоки и циклы первой ступени, второй ступени и т. д. Счет ступеней оказывается удобным производить в направлении, обратном процессу их фактического получения, то есть, иначе говоря, блоками и циклами первой ступени считать не самые крупные, а самые мелкие блоки и циклы.

Этап блочного синтеза при синтезе различных цифровых автоматов варьируется в очень широких пределах. Общих методов блочного синтеза (применительно к цифровым автоматам произвольного назначения) в настоящее время не существует. В настоящей главе мы опишем процесс блочного синтеза применительно лишь к одному классу автоматов, наиболее широко распространенному в настоящее время, а именно к классу *универсальных программных автоматов*, известных обычно под именем *универсальных цифровых вычислительных машин с программным управлением*.

В основу этих автоматов положен так называемый *принцип программного управления*, использующий *операционно-адресную* организацию управления алгоритмическим процессом.

Суть этого принципа заключается в следующем. Информация, с которой оперирует автомат, разделяется на две части: на собственно входную информацию и на информацию об алгоритме, который должен быть реализован рассматриваемым автоматом. Информация об алгоритме называется обычно *программой*. Хотя всю информацию, воспринимаемую автоматом, можно считать одним словом, при командно-адресной организации управления эту информацию делят на более мелкие слова. В соответствии с принадлежностью информации к одному из двух описанных выше видов различают *информационные слова* (называемые в дальнейшем просто словами или числами) и *программные слова* (называемые обычно командами или приказами).

Чаще всего все слова (а равным образом и все команды) в автомате имеют одинаковую длину; обычно стремятся к тому, чтобы эта длина была одинаковой для информационных слов и для команд. Каждое слово (как информационное, так и программное) хранится в определенной части *памяти* (то есть в определенной группе запоминающих элементов) автомата, называемой *ячейкой памяти*. Ячейки памяти, а следовательно, и содержащиеся в них слова, нумеруются натуральными числами, называемыми *адресами* этих ячеек.

Каждая команда осуществляет элементарный акт преобразования информации, называемый *операцией*. Операция выполняется с одним или с несколькими словами (информационными или программными), задаваемыми своими адресами. В соответствии с этим *код команды* (программное слово) составляется из двух частей — *операционной* и *адресной*. В операционной части указывается код (условный номер) операции, выполняемой данной командой, в адресной части — адреса слов, над которыми должна выполняться данная операция. В соответствии с числом этих адресов различают *одноадресные*, *двухадресные*, *трехадресные* и *четырехадресные* команды. Команды с числом адресов, большим, чем четыре, на практике обычно не используются.

При построении автоматов, реализующих вычислительные алгоритмы, одним из наиболее употребительных типов

операций являются обычные арифметические операции: сложение, вычитание, умножение и деление. Каждая из таких операций выполняется над двумя информационными словами (числами), а ее результатом является одно информационное слово. Для указания ячеек, в которых должны храниться эти слова, необходимо иметь три адреса. Таким образом, для вычислительных алгоритмов представляется целесообразным иметь по крайней мере т р е х а д р е с н у ю систему команд.

В действительности, кроме выполнения самих операций, необходимо обеспечить еще определенный порядок следования команд. Различают два вида такого порядка: естественный и принудительный. При *естественном порядке следования команд* после выполнения каждой очередной команды выполняется команда, расположенная в следующей по порядку ячейке памяти, то есть в ячейке памяти, адрес которой на единицу больше адреса выполненной команды. В случае *принудительного порядка следования команд* адрес ячейки, в которой хранится следующая команда, указывается в адресной части предыдущей (выполняемой) команды.

В соответствии со сказанным, для выполнения вычислительных алгоритмов целесообразно иметь либо трехадресную систему команд (с естественным порядком следования), либо четырехадресную систему команд (с принудительным порядком следования). В последнем случае четвертый адрес в команде используется для указания адреса следующей команды программы.

Обычный рабочий цикл программного автомата, имеющего трех- или четырехадресную систему команд, состоит из следующих шагов:

1) выборка из памяти первого информационного слова A (по первому адресу команды);

2) выборка из памяти («чтение») второго информационного слова B (по второму адресу команды);

3) выполнение операции над выбранными словами A и B в соответствии с кодом операции выполняемой команды и получение результата операции,— некоторого слова C;

4) запись результата (то есть слова C) в память (по третьему адресу команды);

5) выборка из памяти следующей команды (в случае трехадресной системы эта выборка производится из следующей по порядку ячейки памяти, в случае четырехадресной системы — из той ячейки памяти, которая указана в четвертом адресе выполняемой команды).

В ряде случаев более удобно считать, что рабочий цикл автомата начинается с выборки из памяти той команды, которая должна выполняться на последующих шагах этого цикла. В случае команд меньшей адресности (т. е. одноадресных или двухадресных) этот цикл выполняется с помощью нескольких команд. Рабочий цикл автомата в этом случае занимает лишь некоторую часть описанного цикла. Так, в случае одноадресной системы команд с естественным порядком их следования в течение одного обычного рабочего цикла выполняется либо чтение, либо запись (в соответствии с кодом операции) информационного слова в память автомата (по адресу, указанному в команде) и выборка следующей по порядку команды. Описанный выше нормальный трехадресный цикл может быть выполнен, очевидно, в течение трех одноадресных циклов.

Важной особенностью современных универсальных программных автоматов является наличие у них таких операций, которые позволяют изменять команды программы (точнее, их адресную часть), а также менять порядок следования команд в зависимости от результатов, полученных после выполнения операций над информационными словами. Операции первого вида принято называть *операциями переадресации*, а операции второго вида — *операциями условного перехода*.

Будем для определенности, рассматривать какую-нибудь одну возможную реализацию этих операций. О п е р а ц и ю п е р е а д р е с а ц и и можно задать трехадресной командой, в первом адресе которой указывается адрес переадресуемой команды, во втором — константа переадресации, состоящая из трех отдельных констант (для всех трех адресов переадресуемой команды), а на месте третьего адреса записывается адрес ячейки, в которую направляется переадресованная команда. На практике удобно пользоваться различными константами переадресации, однако для целей настоящего параграфа доста-

точно рассматривать переадресации (то есть изменения адресов команд) лишь на ± 1.

Операцию условного перехода мы также будем задавать трехадресной командой. При выполнении этой команды, слово, выбранное на основании первого адреса команды, сравнивается со словом, выбранным на основании ее второго адреса. В случае несовпадения выбранных слов выбирается следующая по порядку команда, а в случае совпадения — команда, находящаяся в ячейке, адресом которой служит третий адрес команды условного перехода. Определенную таким образом операцию мы будем называть *операцией условного перехода по точному совпадению слов.*

Для построения универсальных программных автоматов большое значение имеют еще три операции, а именно операции пересылки, ввода и вывода.

Операция пересылки может быть реализована в виде двухадресной команды, при выполнении которой слово из ячейки с адресом, равным первому адресу команды, пересылается в ячейку с адресом, равным второму адресу команды. В случае бо́льшего числа адресных мест в команде остальные адреса не используются.

Операция ввода может быть реализована вообще без использования адресов. При выполнении этой операции информация (в виде последовательности слов), подаваемая на специальное вводное устройство, записывается в последовательно расположенные ячейки памяти (начиная с некоторой ячейки). Обычно оказывается целесообразным использовать адресную часть команды для указания ячейки, в которую помещается первое вводимое слово.

Операция вывода состоит в выводе из автомата через специальное выводное устройство последовательно, слово за словом, содержимого всех ячеек памяти автомата, адреса которых заключены между двумя заданными числами a и b. Эти числа указываются в двух адресах команды, реализующей операцию вывода.

Для наших целей более удобно считать, что операция вывода состоит в выводе содержимого всех ячеек памяти, начиная с ячейки заданного адреса, вплоть до ячейки, в которой записано определенное слово, чаще всего (при

двоичном структурном алфавите) слово, состоящее из одних нулей. Такую операцию вывода легко реализовать с помощью обычной команды вывода (с указанием границ a и b выводимого массива ячеек), а также команд переадресации и условного перехода.

Память в автомате предполагается устроенной таким образом, что при выборе (считывании) слова из какой-либо ячейки памяти информация в этой ячейке не разрушается, так что выборка слова из любой данной ячейки может осуществляться неограниченное число раз, коль скоро это слово было помещено в ячейку. При записи в ячейку нового слова содержавшаяся в ней ранее информация уничтожается.

Программным автоматом мы будем называть цифровой автомат, основанный на принципе программного управления в том виде, в каком он был изложен выше, и имеющий в числе своих операций операции ввода и вывода. В произвольных программных автоматах набор операций, реализуемых автоматом, не обязан, вообще говоря, содержать описанные выше операции переадресации, условного перехода и пересылки. Наличие этих операций оказывается существенным лишь для *универсальности* автомата.

1.1. *Говорят, что программный автомат универсален, или, более точно, что он обладает универсальным набором операций, если любой алгоритм может быть представлен в виде конечного набора выполняемых этим автоматом команд (программы работы автомата) и фактически реализован им при условии игнорирования ограничений, накладываемых конечностью объема памяти (числа ячеек памяти) автомата.*

Реализация алгоритма автоматом, о которой идет речь в приведенном определении, предусматривает возможность изменения кодирования входной и выходной информации, в частности возможность записи входной и выходной информации в каком-либо фиксированном (не зависящем от выбора алгоритма) алфавите. Точный смысл термина «реализация алгоритма» может быть определен следующим образом.

Если дан произвольный алгоритм φ, *то его можно с помощью простейших побуквенных преобразований вход-*

ной и выходной информации (см. главу 1, § 2) *преобразовать в алгоритм* ψ *так, что, вводя в данный программный автомат* A *информацию об алгоритме* ψ *(состоящую из конечного числа команд и информационных слов и зависящую только от алгоритма* ψ*) и любое входное слово* p *алгоритма* ψ*, которое преобразуется алгоритмом* ψ *во вполне определенное выходное слово* $q=\psi(p)$*, мы всегда через конечное число циклов работы автомата получим слово* q *на выходе выводного устройства автомата* A*. При этом мы, не взирая на фактический объем памяти автомата* A*, предполагаем, что этот объем достаточен для того, чтобы поместить как программу работы автомата, так и всю необходимую для его работы информацию (входное слово* p*, выходное слово* q *и всю промежуточную информацию, которая возникает в процессе работы автомата).*

Подчеркивая зависимость программы т о л ь к о от алгоритма ψ, мы исключаем тем самым возможность зависимости ее от вида входного слова p. Ясно, что если бы мы не исключили указанную зависимость, то существовала бы возможность строить тривиальные программы, дающие верный ответ лишь для данного входного слова p (или некоторого множества входных слов), в то время как нам требуется такая программа, которая обеспечивала бы получение правильного ответа для любого входного слова, принадлежащего области определения алгоритма ψ. Упомянутые выше тривиальные программы можно было бы строить, вводя вместе со словом p соответствующее ему выходное слово $q=\psi(p)$. Программа в таком случае могла бы состоять из одной единственной команды вывода.

С учетом всех сделанных замечаний оказывается справедливым следующее важное предложение.

1.2. *Программный автомат является универсальным, если в набор его операций, кроме операций ввода и вывода, входят также операции пересылки, переадресации на* ± 1 *и условного перехода по точному совпадению слов.*

Полное доказательство предложения 1.2 с изложением всех входящих в него подробностей достаточно громоздко. Мы ограничимся поэтому лишь тем, что наметим общий ход этого доказательства, изложив наиболее существенные его стороны.

Прежде всего, имея в виду замечание, сделанное в конце § 3 гл. 1, мы можем предполагать при реализации алгоритмов в автомате, что все эти алгоритмы являются нормальными алгоритмами Маркова. Поскольку при определении реализации алгоритмов в автомате предусматривалась возможность произвольного перекодирования входной и выходной информации, мы можем предполагать, что все алгоритмы, о которых будет итти речь, имеют фиксированный входной и выходной алфавит $\varkappa=(x_1, x_2,\ldots,x_n)$.

Мы будем предполагать также, что размер ячеек памяти рассматриваемого автомата достаточен для того, чтобы в ней могло помещаться $2n+4$ различных кода, а именно коды, представляющие n букв $x_1,\ldots, x_n$, и специально вводимых дубликатов $\bar{x}_1,\ldots, \bar{x}_n$ этих букв, три кода, представляющих используемые при записи нормальных алгоритмов знаки (стрелку, стрелку с точкой, знак раздела между формулами подстановок) и специальный пустой код (состоящий из нулей), заполняющий ячейки, в которые не записывалась никакая информация. Заметим, впрочем, что если размер ячеек окажется недостаточным для помещения указанных кодов, каждый такой код может быть помещен в нескольких соседних ячейках. С некоторым усложнением доказательство теоремы 1.2 может быть проведено и в этом случае.

Пусть нам задан теперь произвольный (нормальный) алгоритм φ (в алфавите $\varkappa$) и произвольный программный автомат A, в набор операций которого входят операции пересылки, переадресации на ± 1 и условного перехода по точному совпадению слов. Разобьем память автомата A (предполагаемую неограниченной) на пять частей, в первую из которых введем программу работы автомата (определяемую ниже), во вторую — таблицу подстановок алгоритма φ (см. главу 1, § 3), а в третью — произвольное слово p во входном алфавите ($\varkappa$) алгоритма φ. Будем предполагать, что первая часть памяти включает в себя все ячейки с адресами от n_0+1 до n_1, вторая — все ячейки с адресами от n_1+1 до n_2, а третья — все ячейки с адресами от n_2+1 до n_3. В четвертую часть памяти, включающую в себя ячейки с адресами от 1 до n_0, мы введем некоторые постоянные коды, общие для всех алгоритмов φ

(коды всех букв $x_1, \ldots, x_n$ и из дубликатов, коды стрелки, стрелки с точкой, знака раздела и пустой код). Пятая часть памяти, состоящая из ячеек с адресами от n_3+1 до ∞, вначале заполняется пустыми (нулевыми) кодами. Мы будем предполагать также, что пустые коды помещены в ячейках с адресами n_0, n_1, n_2, n_3, лежащих на границах выделенных частей памяти.

Таблица подстановок алгоритма φ и входное слово p выписываются (начиная с ячеек n_1+1 и n_2+1) последовательно, буква за буквой и подстановка за подстановкой в следующие друг за другом ячейки памяти. В каждую ячейку помещается по одной букве, включая символы стрелки, стрелки с точкой и знаки раздела между подстановками.

Поясним теперь способ составления программы для реализации алгоритма φ. Для определенности допустим, что мы имеем дело с трехадресной системой команд, работающей в условиях естественного следования команд.

В программе, реализующей алгоритм φ, первой, после команд ввода информации, будет стоять команда K_1 условного перехода по точному совпадению слова, называемая для краткости командой УП. Эта команда должна осуществлять сравнение кодов в ячейках с адресами n_1+1 и n_2+1 (первой буквы первой подстановки алгоритма φ и первой буквы входного слова). Следующей должна стоять команда K_2 переадресации второго адреса предыдущей команды на $+1$ и еще некоторые команды переадресации, а вслед за ними команда УП K_j, сравнивающая два кода, выбираемые из какой-либо одной ячейки, и осуществляющая поэтому безусловный переход к команде, указанной по ее третьему адресу. В качестве такой команды должна фигурировать, очевидно, переадресованная команда K_1, с тем, чтобы автомат после неудачи первого сравнения стал сравнивать первую букву первой подстановки алгоритма φ со второй буквой входного слова p. В случае удачи первого сравнения, выполненного командой K_1, мы должны по третьему адресу этой команды отправляться к команде K_j, осуществляющей сравнение кода в ячейке n_2+1 с кодом буквы x_1, запасенном в одной из постоянных ячеек четвертой части памяти.

Вслед за этой командой должны стоять команды, осуществляющие аналогичные сравнения кода в ячейке n_2+1 с кодами букв $x_2,\ldots, x_n$. В случае удачи любого из этих сравнений (скажем, с буквой x_k) должна осуществляться команда пересылки кода $\overline{x_k}$ (дубликата буквы x_k, запасенного в одной из ячеек четвертой части памяти) в ячейку с адресом n_2+1. При этом как упомянутые команды сравнения, так и команды пересылок должны переадресовываться (во втором адресе на $+1$) командами переадресации, расположенными непосредственно вслед за командой K_2.

Продолжая подобным образом, мы можем построить программу, состоящую из конечного числа команд пересылки, переадресации и условного перехода, которая находит первое вхождение левой части первой формулы из таблицы подстановок алгоритма φ и заменяет все буквы этого вхождения их дубликатами.

Затем, используя аналогичные приемы, можно построить программу, заменяющую найденное вхождение левой части формулы подстановки ее правой частью. При этом необходимо осуществлять раздвигание частей слова, предшествующих этому вхождению и следующих за ним. Нетрудно понять, что все указанные преобразования можно осуществить, пользуясь только операциями пересылки, переадресации и условного перехода.

В случае отсутствия вхождения левой части первой формулы таблицы подстановок во входное слово все команды первой программы, за исключением команд, первые адреса которых относятся к четвертой зоне памяти (от 1 до n_0), должны быть переадресованы по первому адресу на величину, равную количеству букв в первой формуле (включая стрелку и знак раздела). Такая переадресация может быть выполнена с помощью конечного числа переадресаций на $+1$. В результате мы будем иметь возможность повторить с помощью уже построенной части программы поиск вхождения левой части второй формулы и замену этого вхождения правой частью формулы.

Если не ставить себе целью экономное расходование команд, можно составить программу, отводя отдельную команду УП для сравнения каждой буквы из левых частей подстановок (переадресуемой) с буквой входного слова и тем самым избежать переадресаций во всех зонах

памяти, кроме третьей. Переадресации же в третьей зоне принципиально нельзя избежать, поскольку входное слово может иметь сколь угодно большую длину. Если не использовать переадресации, то на все возможные сравнения потребовалось бы в таком случае бесконечное число команд.

Проведенные рассуждения дают читателю достаточное понятие о том, каким образом может быть построена полная программа, выполняющая работу заданного нормального алгоритма φ. Заметим, что такая программа будет состоять исключительно из команд пересылки, переадресации и условного перехода. Лишь в начале и в конце программы будут стоять команды, обеспечивающие соответственно ввод и вывод информации из автомата. Мы можем, следовательно, считать теорему 1.2 доказанной.

Хотя теорема 1.2 и показывает, что свойство универсальности достигается посредством использования весьма ограниченного набора операций, тем не менее при построении реальных универсальных программных автоматов стремятся обычно к расширению этого набора. Такое расширение осуществляется прежде всего за счет арифметических операций. В набор операций вводятся обычно также различные логические операции, например поразрядное логическое умножение и поразрядная дизъюнкция (двоичных) кодов:

$$(x_1, x_2, \ldots, x_n) \cdot (y_1, y_2, \ldots y_n) = (x_1y_1, x_2y_2, \ldots, x_ny_n),$$
$$(x_1, x_2, \ldots, x_n) \vee (y_1, y_2, \ldots, y_n) = (x_1 \vee y_1, x_2 \vee y_2, \ldots, x_n \vee y_n).$$

Весьма полезными оказываются различные операции сдвига кодов: *правый сдвиг*, преобразующий код $(x_1, x_2, \ldots, x_n)$ в код $(0, x_1, x_2, \ldots, x_{n-1})$, *левый сдвиг*, преобразующий код $(x_1, x_2, \ldots, x_n)$ в код $(x_2, x_3, \ldots, x_n, 0)$, *левый и правый циклические сдвиги*, преобразующие код $(x_1, x_2, \ldots, x_n)$ соответственно в коды $(x_2, x_3, \ldots, x_n, x_1)$ и $(x_n, x_1, \ldots, x_{n-1})$ и др.

Для того чтобы не держать в рабочем режиме автомат после того, как он выдаст всю предусмотренную в программе информацию, имеется специальная команда, выключающая автомат. Эта команда получила в литературе по электронным вычислительным машинам название *команды останова*. Наконец, в больших универсальных

программных автоматах обычно существует память двух видов: *оперативная память*, имеющая относительно небольшой объем и отличающаяся относительно высоким быстродействием, и *внешняя память*, имеющая значительно бóльший объем, но значительно мéньшую скорость действия. В соответствии с этим в набор операций автомата включаются операции, осуществляющие обмен информацией между оперативной и внешней памятью (в одну и в другую сторону). Команды такого обмена имеют почти такую же структуру, как и команды ввода и вывода. Различие состоит лишь в необходимости указания граничных адресов передаваемой информации не только в оперативной, но и во внешней памяти.

Блок-схема современных универсальных программных автоматов состоит из пяти основных блоков: 1) *запоминающего устройства* (ЗУ), *арифметического устройства* (АУ), 3) *устройства управления* (УУ), 4) *вводнóго устройства* (ввод) и 5) *выводного устройства* (вывод). Как уже отмечалось выше, в больших универсальных автоматах запоминающее устройство разбивается на два блока: 1) *оперативное запоминающее устройство* (ОЗУ) и 2) *внешнее запоминающее устройство* (ВЗУ).

Каждый из указанных блоков имеет свое особое назначение и выполняет определенные функции в работе универсального программного автомата. Назначение устройства управления состоит в координации действий всех остальных устройств автомата. В синхронных автоматах в состав устройства управления входит обычно тактирующий генератор, задающий (с какой-либо постоянной частотой) моменты изменения состояний отдельных устройств автомата. Частота сигналов, выдаваемых этим генератором, называется *рабочей частотой* рассматриваемого универсального автомата. В современных быстродействующих универсальных электронных цифровых машинах эта частота измеряется сотнями тысяч и даже миллионами герц (колебаний в секунду).

Оперативное запоминающее устройство служит для запоминания (записи) и хранения информации (информационных и программных слов) и характеризуется возможностью быстрого извлечения

(чтения) записанной информации и пересылки ее в другие блоки в соответствии с сигналами, поступающими из УУ. В настоящее время наиболее употребительными являются оперативные запоминающие устройства, построенные на миниатюрных магнитных (ферритовых) кольцах.

Внешнее запоминающее устройство служит резервом для оперативного запоминающего устройства; оно получает из оперативного запоминающего устройства информацию, предназначенную для относительно длительного хранения, и передает в оперативное запоминающее устройство ту информацию, которая требуется в соответствии с сигналами, поступающими из УУ. Обычно обмен информации между ОЗУ и ВЗУ осуществляется не посредством пересылки одиночных слов, а посредством передачи целых групп слов, называемых *массивами*. Внешние запоминающие устройства реальных автоматов изготавливаются обычно в виде магнитных лент и магнитных барабанов.

Арифметическое устройство служит для выполнения арифметических, а также логических, операций над информационными словами, поступающими в него из ОЗУ по сигналам, вырабатываемым УУ. Результаты этих операций с помощью других сигналов УУ передаются на хранение в ОЗУ.

Устройство ввода осуществляет (по соответствующему сигналу УУ) ввод информации (информационных и программных слов), запасенной вне автомата на специальных карточках или лентах, снабженных отверстиями (так называемые перфорированные карты, или, коротко, *перфокарты* и перфорированные ленты, или, коротко, *перфоленты*). Эта информация поступает в ОЗУ либо непосредственно, либо после предварительного прохождения через АУ. Необходимость пропускания информации через АУ вызывается тем обстоятельством, что числовая информация вне машины представляется обычно в привычной для человека десятичной системе счисления, а внутри машины — чаще всего в двоичной системе. Пропуская числовую информацию через АУ, мы получаем возможность попутно преобразовать ее из одной системы счисления в другую.

Устройство вывода служит для автоматического печатания (по соответствующему сигналу УУ) информации, накопленной в тех или иных ячейках ОЗУ. Печатание может носить буквенный, цифровой или комбинированный (буквенно-цифровой) характер. При выводе числовой информации ее также можно пропускать через АУ для автоматического перевода в десятичную систему счисления. Современные цифровые автоматы кроме вывода в виде печатного текста имеют еще и другие выводные устройства, позволяющие выводить информацию на перфокартах, перфолентах и т. п.

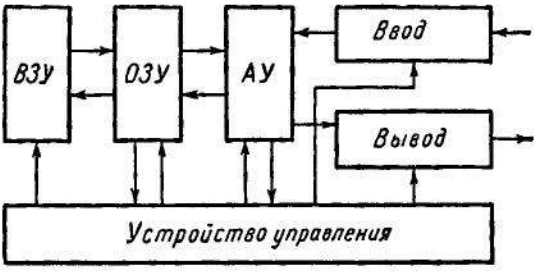

Рис. 26.

Обозначая направление передачи информации от одного блока к другому с помощью стрелок, мы можем представить блок-схему универсального программного автомата так, как это изображено на рис. 26. Стрелки, идущие на этом рисунке от УУ к другим блокам, означают направление передачи соответствующих управляющих сигналов. Стрелка, идущая из ОЗУ в УУ, соответствует передаче команд из ОЗУ в УУ, где они подвергаются расшифровке и исполнению. Назначение обратной связи от АУ к УУ будет объяснено ниже (в § 3). Заметим, что хотя каждый канал передачи информации от одного блока к другому обозначен на рис. 26 одной линией, в действительности, однако, за ним может скрываться целый ряд параллельных каналов.

Возможны два различных способа передачи информационных и программных слов между блоками автомата. При одном способе передачи слов для передачи

каждого разряда кода слова используется отдельный физический канал (называемый обычно *шиной*), так что передача всех разрядов кода производится одновременно. При другом способе передачи слов все разряды кода передаются по одному и тому же физическому каналу (шине) последовательно, один за другим.

Программные автоматы, в которых осуществлен первый способ передачи кодов, называются *параллельными*, а автоматы, в которых имеет место второй способ передачи кодов, называются *последовательными*. Возможна, разумеется, и смешанная (параллельно-последовательная) система передачи кодов в автомате.

Скорость работы универсальных программных автоматов измеряется обычно числом рабочих циклов, выполняемых автоматом в течение одной секунды. Часто различные рабочие циклы (например, циклы, соответствующие командам сложения и умножения) имеют различную длительность, поэтому при определении *быстродействия автомата* прибегают к подсчету среднего числа циклов (команд), выполняемых им в единицу времени. При этом в большинстве случаев не учитывают команд ввода, вывода и обмена информацией между ОЗУ и ВЗУ, предполагая, что машина (автомат) работает все время с оперативным запоминающим устройством.

Быстродействие универсального автомата, подсчитанное при условии работы автомата с оперативным запоминающим устройством и выраженное средним числом операций (команд), выполняемых автоматом в течение одной секунды, мы будем называть номинальным быстродействием этого автомата.

Номинальное быстродействие не определяет, как правило, реального времени, которое необходимо затратить, решая с помощью универсального автомата ту или иную задачу. Для подсчета реального времени решения задачи нужно учитывать время, затрачиваемое на ввод и на вывод информации, потери времени за счет обращения к внешнему запоминающему устройству (работающему значительно медленнее, чем ОЗУ), потери времени на многократное повторение решения для получения ответа с заданной степенью надежности (в случае недостаточной

надежности автомата) и, наконец, средние потери времени (приходящиеся на данную задачу) за счет таких факторов, как профилактический ремонт и устранение возникающих в процессе работы неисправностей.

Необходимо, далее, учитывать различную сложность операций, выполняемых автоматом. Устанавливая те или иные *весовые коэффициенты* для различных операций, мы получаем возможность выражать все операции через какую-нибудь одну операцию (например, сложение), принимаемую в качестве *стандартной операции*.

Быстродействие универсального автомата, выраженное числом выполняемых им в единицу времени (секунду) стандартных операций и учитывающее все указанные выше потери времени, мы будем называть эффективным быстродействием этого автомата.

Нетрудно понять, что определенное таким образом эффективное быстродействие зависит от задачи, которая решается автоматом. Для определения *среднего эффективного быстродействия* автомата можно воспользоваться следующим приемом. Прежде всего выделяются: конечное множество M типовых задач $R_1, \ldots, R_n$, которые должен решать рассматриваемый автомат A, эффективные быстродействия $v_1, v_2, \ldots, v_n$ автомата A по каждой из этих задач и вероятности $p_1, \ldots, p_n$ введения в автомат каждой из данных задач $R_1, \ldots, R_n$ (равные процентам времени, занимаемого каждой из них в общем времени работы автомата A). Далее предполагается, что $p_1+p_2+\ldots+p_n=1$, тогда среднее эффективное быстродействие v автомата A на множестве задач M определяется по формуле

$$\frac{1}{v}=\frac{p_1}{v_1}+\frac{p_2}{v_2}+\ldots+\frac{p_n}{v_n}. \qquad (1)$$

Среднее эффективное быстродействие представляет собою универсальный критерий эффективности программного автомата, ибо он определяет фактическую его производительность (при решении задач данного класса) и характеризует усредненным образом все блоки автомата. Весьма часто случается, что автомат с меньшим номинальным быстродействием, но обладающий бо́льшим объемом оперативной памяти, бо́льшей скоростью ввода и вывода или бо́льшей надежностью,

имеет бо́льшее среднее эффективное быстродействие, чем более скоростной — в смысле номинального быстродействия — автомат.

В ряде случаев оказывается целесообразным относить среднее эффективное быстродействие универсального программного автомата на единицу оборудования, затраченного на его изготовление. Получаемый таким образом критерий *цены эффективного быстродействия* позволяет сравнивать между собой эффективность самых различных автоматов [1]).

§ 2. Принципы построения арифметических устройств

Вопрос о различных способах построения арифметических устройств универсальных программных автоматов является одним из наиболее разработанных вопросов теории таких автоматов. Ему посвящено большое число работ различных авторов, среди которых мы укажем лишь монографии Р. К. Ричардса [2]) и М. А. Карцева [3]). Настоящий параграф не преследует цели дать сколько-нибудь полное изложение полученных в этой области результатов или законченную классификацию различных типов арифметических устройств. Цель настоящего параграфа более скромная. Она заключается в том, чтобы на примере арифметического устройства какого-либо одного типа изложить некоторые общие принципы, позволяющие уяснить себе методы организации переработки информации в современных универсальных программных автоматах, предназначенных для реализации вычислительных алгоритмов.

Арифметические устройства современных универсальных программных автоматов составляются из *сумматора* того или иного типа и одного или нескольких *регистров*. Регистр представляет собою совокупность запоминающих

---

[1]) Более подробно об этом критерии см. В. М. Глушков, Два универсальных критерия эффективности вычислительных машин. ДАН УССР, № 4, 1960, стр. 477—481.

[2]) Р. К. Ричардс, Арифметические операции на цифровых вычислительных машинах, ИЛ, М., 1957.

[3]) М. А. Карцев, Арифметические устройства электронных цифровых машин, Физматгиз, М., 1958.

элементов, предназначенных для запоминания одного слова, которые упорядочиваются в соответствии с принятым порядком расположения букв (разрядов) в слове. В регистрах АУ запоминаются информационные слова, рассматривающиеся обычно как числа, в других регистрах (прежде всего в регистрах УУ) могут запоминаться также и программные слова — команды. На практике наиболее употребительны регистры, состоящие из триггеров; такие регистры называются *триггерными регистрами*.

Арифметические устройства в зависимости от характера их работы делятся на два больших класса: на класс последовательных арифметических устройств и класс параллельных арифметических устройств. В так называемых *последовательных арифметических устройствах* арифметические операции выполняются последовательно, разряд за разрядом; в так называемых *параллельных арифметических устройствах* арифметические операции со всеми разрядами производятся одновременно. В последовательных АУ используются одноразрядные сумматоры, в параллельных — многоразрядные сумматоры (подобные сумматору, построенному нами в примере 2 из § 2 предыдущей главы).

Регистры последовательных АУ также обычно несколько отличаются от регистров параллельных АУ. Отличие это состоит в том, что в параллельных АУ входные и выходные каналы каждого запоминающего элемента, входящего в состав регистра, подключаются к узлам схем, расположенным за пределами регистра, чем обеспечивается возможность одновременной записи и чтения информации во всех разрядах регистра. Регистры, обладающие таким свойством, мы будем называть *параллельными регистрами*.

Регистры последовательного АУ воспринимают информацию через один (например, младший) разряд (называемый входным разрядом) и выдают ее также через один (например, старший) разряд (называемый входным разрядом). Только эти два разряда, которые мы назовем *внешними*, находятся в связи со схемами, расположенными вне регистра. Что же касается внутренних разрядов, то они вступают в связь друг с другом, так сказать, в виде цепочки; по этой цепочке сигналы могут передаваться (с за-

держкой на один такт в каждом разряде) в направлении от входного разряда к выходному. Регистры, построенные по такому принципу, мы назовем *последовательными*.

Необходимо подчеркнуть, что способность передавать информацию от разряда к разряду не является исключительной принадлежностью лишь последовательных регистров. Возможность осуществления сдвигов информации вдоль регистра предусматривается обычно и в некоторых параллельных регистрах, которые называются в этом случае *сдвиговыми регистрами*. Для случая числовых кодов мы условимся, что справа располагаются младшие разряды, а слева — старшие. В соответствии с этим сдвиг в сторону младших разрядов носит название *правого сдвига*, а в сторону старших разрядов — *левого сдвига*.

При проектировании арифметического устройства первой основной задачей, после определения типа устройства, является определение числа регистров и выбора так называемых *микроопераций*. Под *микрооперацией* мы будем понимать процесс переработки информации в автомате или в какой-либо его части, происходящий за время одного такта работы автомата (промежуток между двумя последовательными моментами дискретного автоматного времени). Поскольку в теории вычислительных автоматов в понятие такта вкладывается обычно другой смысл, мы, чтобы избежать путаницы, будем для обозначения указанного (элементарного) промежутка времени на протяжении настоящего и последующего параграфа употреблять термин «микротакт».

Обычно принято все арифметические операции в параллельных АУ составлять из *микроопераций суммирования* (раскладываемой в большинстве случаев на две микрооперации — операции *поразрядного сложения* и операции *реализации переносов*), а также микроопераций *сдвига и передачи кодов с одного регистра на другой*. Существует много различных способов разложения арифметических операций на отдельные микрооперации. Ниже мы рассмотрим в качестве примера лишь один из таких способов.

Общая тенденция при выборе набора микроопераций арифметического устройства заключается в максимальном

уменьшении числа различных микроопераций, поскольку введение каждой новой микрооперации связано с дополнительными затратами оборудования. В то же время набор операций (команд), составляемых из выбранных микроопераций, стремятся сделать, по возможности, более широким, поскольку наличие бо́льшего числа операций, как правило, упрощает постановку задач для конструируемого универсального автомата (то есть упрощает программирование).

В борьбе между этими двумя противоположными тенденциями выработалась такая методика выполнения арифметических операций, которая использует, в возможно большей степени, одни и те же микрооперации для осуществления различных арифметических операций. Для успеха этой методики существенное значение имеют специальные системы кодирования, облегчающие выполнение операций над отрицательными числами.

Мы рассмотрим одну из самых распространенных систем кодирования, известную под названием *системы обратных кодов*.

Хотя эту систему кодирования можно применять для произвольных систем счисления, мы ограничимся лишь двоичной системой.

Обычный код — называемый *прямым кодом* — n-разрядного двоичного числа $x=x_n x_{n-1}\ldots x_1$ строится с помощью прибавления к нему $(n+1)$-го двоичного разряда x_{n+1}, называемого *знаковым разрядом*. В случае, если число x является положительным, в знаковом разряде ставится 0, если же это число является отрицательным, то в знаковом разряде ставится 1.

Обратный код для положительных чисел совпадает с прямым кодом; что же касается отрицательного числа $1x_n x_{n-1}\ldots x_1$, то его обратный код получается из прямого кода заменой цифры каждого из разрядов $x_n\ x_{n-1}\ldots x_1$ противоположной ей цифрой $\bar{x}_i$. Число н у л ь имеет два обратных кода, первый из которых состоит из одних нулей, а второй — из одних единиц. Условимся в качестве обратного кода нуля фиксировать всегда код, состоящий из одних единиц. Иными словами, при использовании обратного кода н у л ь будет считаться отрицательным числом.

Для обратных кодов вводится специальная операция, называемая *циклическим сложением*, для обозначения которой мы введем специальный символ $\oplus$. Смысл этой операции заключается в следующем. Все $n+1$ разрядов обратного кода (включая знаковый разряд) рассматриваются как совершенно равноправные; правило сложения этих кодов в случае циклического сложения остается обычным (поразрядное сложение и образование переносов из младших разрядов в старшие); отличие циклического сложения от обычного состоит лишь в том, что в случае образования сигнала переноса в $(n+1)$-м (знаковом) разряде сигнал переноса не теряется, а направляется снова в младший разряд кода суммы для осуществления повторного сложения, образование сигнала переноса последнего рода носит название *циклического переноса*.

Заметим, что циклический перенос может вызвать, в свою очередь, образование новых переносов из младших разрядов в старшие вплоть до знакового разряда. Поэтому возникает вопрос о корректности определения операции циклического переноса. Иначе говоря, необходимо проверить, не будет ли определенный нами процесс переноса осуществляться неограниченно долго, так что сигнал переноса снова и снова будет пробегать весь путь от младшего разряда до знакового и обратно.

Нетрудно, однако, убедиться в том, что такая опасность полностью исключена. Пусть нам даны два произвольных $(n+1)$-разрядных двоичных кода $x=x_{n+1}\ x_n \ldots x_1$ и $y=y_{n+1}y_n \ldots y_1$. Если при циклическом сложении этих кодов сигнал переноса не возникает ни в одном из разрядов, то процесс циклического сложения вполне определен. Предположим теперь, что перенос возникает впервые в i-м разряде (считая справа). Это означает, очевидно, что $x_i=y_i=1$. Поскольку перенос p_i из $(i-1)$-го разряда в i-й отсутствует (в силу выбора i), ясно, что в результате сложения кодов еще до реализации циклического переноса в i-м разряде суммы будет получен нуль. Теперь очевидно, что переносы из младших разрядов, возникшие под влиянием циклического переноса, не могут распространиться дальше i-го разряда. Тем самым корректность определения операции циклического сложения полностью оправдана.

При осуществлении сложения чисел в автомате мы будем предполагать коды этих чисел *нормализованными* [1]); это значит, что если i-й разряд кода одного слагаемого соответствует какому-либо двоичному разряду, то i-й разряд кода остальных слагаемых соответствует тому же самому разряду. Для последующих рассуждений можно предполагать, что i-му разряду кода соответствует i-й двоичный разряд, то есть, иными словами, что все рассматриваемые числа являются целыми двоичными числами. Как уже было отмечено выше, обычно в машине фиксируется определенная длина информационного слова (называемая *разрядностью машины*). В соответствии с этим для представления чисел, с которыми оперирует машина, устанавливается некоторое определенное число разрядов n. Если результат той или иной операции оказывается занимающим большее чем n число разрядов, то некоторые его разряды (обычно старшие) теряются. В таком случае принято говорить, что имеет место переполнение разрядной сетки, в результате чего, разумеется, возникает ошибка в вычислениях. При построении универсальных автоматов — и программ для них — принимаются различные меры, чтобы избежать переполнения разрядной сетки. В случае, если выход из имеющегося числа разрядов все же будет иметь место, предусматривается автоматическая остановка машины с соответствующей сигнализацией.

При принятом обычно (при ручном счете) способе сложения чисел различных знаков приходится пользоваться различными правилами в зависимости от характера сочетания знаков слагаемых. В вычислительных автоматах подобное обстоятельство является нежелательным и должно рассматриваться как недостаток принятой методики сложения. Этот недостаток полностью устраняется при пользовании обратным кодом, что вытекает из следующего важного предложения.

2.1. *В результате циклического сложения обратных кодов (нормализованных) чисел любых знаков получается обратный код (алгебраической) суммы этих чисел, при условии, что не имеет места переполнение разрядной сетки.*

[1]) В применении к вычислительным машинам понятие нормализованного кода обычно употребляется в несколько более узком смысле.

Признаком переполнения разрядной сетки при циклическом сложении является получение отрицательной суммы при положительных слагаемых и положительной суммы при отрицательных слагаемых.

Доказательству сформулированного предложения мы предпошлем два замечания. Первое замечание состоит в том, что обратный код $y=1\bar{x}_n\ldots\bar{x}_1$ отрицательного двоичного (целого) числа $-x=-x_n\ldots x_1$ можно трактовать как представление в обычном двоичном коде числа $2^{n+2}-x-1$. Действительно, сложение кодов y и $+x$ по обычным правилам сложения приводит, как легко проверить, к $(n+1)$-разрядному коду, состоящему из одних лишь единиц. Нетрудно понять, что такой код представляет собою число $2^{n+2}-1$. Второе замечание относится к характеру циклического сложения. Ясно, что при отсутствии циклического переноса это сложение ничем не отличается от обычного сложения $(n+1)$-разрядных кодов. При наличии циклического переноса результат циклического сложения будет на $2^{n+2}-1$ меньше результата обычного сложения соответствующих кодов, поскольку при этом теряется одна единица $(n+2)$-го разряда и приобретается (за счет циклического переноса) одна единица первого (младшего) разряда. Наличие циклического переноса легко устанавливается по величине обычной суммы рассматриваемых кодов: циклический перенос имеет, очевидно, место тогда и только тогда, когда эта сумма больше или равна 2^{n+2}.

Для доказательства теоремы 2.1 рассмотрим различные случаи, которые могут встретиться при сложении. Первый случай имеет место тогда, когда оба слагаемые x и y положительны. Ясно, что в этом случае циклический перенос отсутствует, а перенос в знаковый разряд будет иметь место лишь в случае переполнения разрядной сетки. Справедливость утверждения, составляющего содержание теоремы 2.1, для этого случая очевидна. Второй случай имеет место тогда, когда оба слагаемых отрицательны (то есть имеют вид $-x$ и $-y$). Обозначая через x' и y' обратные коды чисел $-x$ и $-y$, мы получим, очевидно, что

$$x'\oplus y'=2^{n+2}-x-1+2^{n+2}-y-1-(2^{n+2}-1)=$$
$$=2^{n+2}-(x+y)-1=(x+y)',$$

где через $(x+y)'$ обозначен обратный код суммы $(-x)+(-y)$. Мы применили здесь второе из сделанных выше замечаний, поскольку наличие единиц в старших разрядах кодов x' и y' обеспечивает наличие циклического переноса.

Если отбросить в кодах x' и y' знаковые разряды, то получатся, очевидно, коды чисел $x''=2^{n+2}-2^{n+1}-x-1$ и $y''=2^{n+2}-2^{n+1}-y-1$. Перенос в знаковый разряд будет иметь место в том и только в том случае, когда сумма этих чисел и единицы циклического переноса больше или равна 2^{n+1}:

$$x''+y''+1=2^{n+2}-(x+y)-1\geqslant 2^{n+1}$$

или, что равносильно этому же, когда $x+y\leqslant 2^{n+1}-1$. Иными словами, перенос в знаковый разряд отсутствует в том и только в том случае, когда имеет место переполнение разрядной сетки. В этом случае при циклическом сложении в знаковом разряде возникнет н у л ь, то есть, иначе говоря, сложение отрицательных чисел приведет к положительному результату. В случае же отсутствия переполнения разрядной сетки циклическое сложение приведет, как следует из проведенных рассуждений, к правильному результату. Следовательно, теорема 2.1 оказывается справедливой и при сложении двух отрицательных чисел.

Рассмотрим теперь случай, когда одно слагаемое (например, $-x$) отрицательно, а второе (y) положительно. Обратный код y' положительного числа y совпадает с двоичным представлением этого числа, а обратный код x' отрицательного числа $-x$ представляет двоичное число $2^{n+2}-2^{n+1}-x-1$. Ясно, что переполнения разрядной сетки в этом случае быть не может. Циклический перенос будет иметь место тогда и только тогда, когда еще до реализации циклического переноса имел место перенос в знаковый разряд. Очевидно, это случится лишь в том случае, если

$$2^{n+2}-2^{n+1}-x-1+y\geqslant 2^{n+1}.$$

Последнее неравенство равносильно неравенству $y-x\geqslant 1$. Это значит, что циклический перенос будет иметь место лишь в случае положительной суммы (напомним, что, в силу принятого выше соглашения, нуль считается отрицательным числом).

Таким образом, рассматриваемый случай разбивается на два подслучая в зависимости от того, будет ли сумма $(-x)+y$ положительна или отрицательна. В первом подслучае мы будем иметь:

$$x' \oplus y' = 2^{n+2} - x - 1 + y - (2^{n+2} - 1) = (-x) + y > 0.$$

Во втором подслучае мы получим:

$$x' \oplus y' = 2^{n+2} - x - 1 + y = 2^{n+2} - (x - y) - 1 \leqslant 0.$$

Поскольку в этом подслучае $-x+y \leqslant 0$, то результат циклического сложения будет равняться обратному коду суммы $-x+y$, что и требовалось доказать.

Таким образом, в результате рассмотрения всех возможных случаев справедливость теоремы 2.1 полностью доказана.

Теорема 2.1 позволяет (при условии использования обратных кодов) выполнять сложение положительных и отрицательных чисел единообразным приемом, осуществляя, например, передачу этих чисел обратным кодом на *накапливающий параллельный сумматор с циклическим переносом* из старшего разряда в младший. Схема такого сумматора может быть построена точно так же, как и схема обычного двоичного параллельного сумматора, рассмотренного нами в примере 2 из § 2 гл. VI (см. рис 22). Различие состоит лишь в том, что все разряды сумматора с циклическим переносом должны быть совершенно идентичны друг другу, а выходной сигнал p_{n+1} (или $\overline{p}_{n+1}$) последнего разряда сумматора должен быть использован для установки в нуль всех триггеров $x_i (i=1, \ldots, n)$ и через задержку (на время переброса триггеров x_i) импульс с выхода p_{n+1} должен подаваться на вход p_1 первого разряда сумматора.

Таким образом может быть построено параллельное арифметическое устройство с одним регистром x, воспринимающим последовательно, один за одним, обратные коды слагаемых и передающее их на второй регистр y, накапливающий обратный код суммы всех этих слагаемых. Этот второй регистр, вместе с соответствующей комбинационной схемой, мы и будем называть *накапливающим сумматором*. В описанном примере прибавление каждого

нового слагаемого осуществляется в один микротакт, в связи с чем сумматор называется *однотактным*.

Операция сложения двух слагаемых при использовании только что описанного АУ может быть составлена из следующих микроопераций:

1) передача (обратного) кода первого слагаемого из ЗУ в регистр x;

2) передача кода с регистра x на сумматор y;

3) передача (обратного) кода второго слагаемого из ЗУ в регистр x;

4) передача кода с регистра x на сумматор y (собственно суммирование);

5) передача полученной суммы из сумматора в ЗУ.

Указанная последовательность микроопераций представляет собою одну из возможных *микропрограмм* для операции сложения (алгебраического). Заметим, что первой микрокоманде (микрооперации) выписанной выше микропрограммы обычно предпосылается микрооперация *очистки регистров* АУ (включая сумматор), то есть посылка импульса, устанавливающая триггеры всех регистров АУ в нулевое состояние.

В случае необходимости выполнения в АУ операции вычитания необходимо иметь еще микрооперацию *инвертирования кода*, позволяющую превращать обратный код положительного числа a в обратный код отрицательного числа $-a$. Для этой цели достаточно, очевидно, инвертировать (заменить 0 на 1, а 1 на 0) каждый разряд первого кода. Микрооперацию инвертирования кода можно осуществлять на одном из регистров АУ, то есть инвертировать код, заполненный на этом регистре, а запоминание результата инвертирования осуществлять на том же самом регистре.

Чаще всего, однако, процесс инвертирования предпочитают совмещать с процессом передачи кода. Для этой цели вводят два типа микроопераций передачи кодов: *обычную передачу кода* и *передачу инвертированного кода*.

Микрооперация инвертирования кода весьма близка к микрооперации преобразования прямого кода в обратный и наоборот (в этом случае инвертирование не затрагивает знакового разряда). Поэтому, если в автомате ис-

пользуется (как это часто случается) как прямой, так и обратный код, то возникает необходимость в д в у х различных микрооперациях для передачи чисел: микрооперации передачи чисел, *сопровождающейся изменением типа кода* (с прямого на обратный и наоборот) и микрооперации передачи чисел *без изменения типа кода*. Первую из этих микроопераций называют обычно *передачей числа обратным кодом*, а вторую — *передачей числа прямым кодом*. Заметим, что при употреблении прямых кодов операция инвертирования кода сводится к инвертированию одного лишь зна́кового разряда.

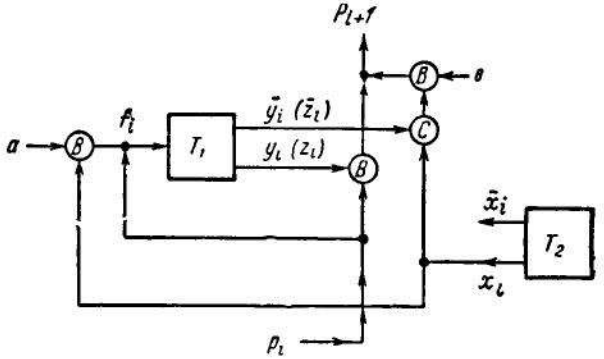

Рис. 27.

Как уже отмечалось выше, на практике применяются чаще всего не однотактные, а двухтактные накапливающие (параллельные) сумматоры, раскладывающие акт сложения на две микрооперации — *поразрядного сложения* и *реализации переноса*. При использовании таких сумматоров лишь для сложения и вычитания, кроме регистра самого сумматора, в АУ необходимо иметь еще один регистр для хранения второго слагаемого (первое слагаемое хранится при этом в самом сумматоре).

Произвольный (i-й) разряд одного из возможных вариантов двухтактного накапливающего сумматора импульсно-потенциального типа изображен на рис. 27. На этом рисунке буквами y_i и x_i обозначены i-е разряды первого и второго слагаемого, буквы T, C и B обозначают

соответственно потенциальный триггер, потенциальное совпадение и импульсно-потенциальный вентиль, а буквы a и b обозначают входные каналы, на которые подаются импульсы от устройства управления микрооперациями в момент, когда необходимо выполнить поразрядное сложение и соответственно реализацию переноса. Через f_i обозначен счетный вход триггера i-го разряда (y_i) сумматора. Входы триггера i-го разряда регистра (x_i) и усилители в цепи p_i на рисунке не обозначены.

В первом такте работы схемы, изображенной на рис. 27, импульс приходит лишь на вход a. Благодаря этому в триггере T_i образуется сумма (по модулю два) $z_i=x_i+y_i$. Во втором такте импульс приходит на вход b и, при наличии переноса из предыдущего разряда, также на вход p_i. На выходах триггера T_i к этому моменту устанавливаются сигналы z_i и $\bar{z}_i$, так что на выходе p_{i+1} возникает импульсный сигнал, выражаемый булевой функцией $p_{i+1}=z_ip_i\vee x_i\bar{z}_i$, а в триггере T_i запоминается i-й разряд суммы (с учетом переноса): $s_i=z_i+p_i=x_i+y_i+p_i$. Поскольку $z_i=x_i+y_i=(x_i\vee y_i)\,(\bar{x}_i\vee\bar{y}_i)$, а $\bar{z}_i=\bar{x}_iy_i\vee x_i\bar{y}_i$, то $p_{i+1}=x_iy_i\vee p_i(x_i\vee y_i)\,(\bar{x}_i\vee\bar{y}_i)$, что совпадает с выражением для функции переноса, которое дает формула (4) из § 2 предыдущей главы.

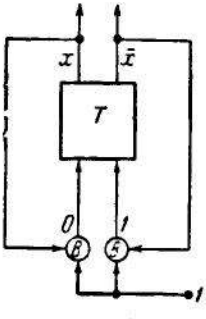

Рис. 28.

Для выполнения микрооперации инвертирования кода на регистре, построенном из триггеров со счетными входами, достаточно, очевидно, подать импульс на входы всех триггеров. То же самое нужно проделать и при выполнении микрооперации перевода хранящегося на регистре прямого кода в обратный и наоборот (только в этом случае не следует подавать импульс на триггер, запоминающий знаковый разряд). Заметим, что при употреблении триггеров с раздельными входами нетрудно построить аналог счетного входа с помощью простой схемы, использующей всего два вентиля (на каждый триггер). Соответствующая схема изображена на рис. 28. При подаче импульса на вход схемы изображенный на ней триг-

гер будет изменять свое состояние на противоположное, то есть работать в режиме счетного входа.

Рассмотренные до сих пор микрооперации достаточны для реализации простейших арифметических операций — сложения и вычитания. При переходе к более сложным операциям (умножения и деления) необходимо ввести еще микрооперации *сдвига кода на регистрах*.

Схема *сдвигового регистра* наиболее просто осуществляется на триггерах с раздельными входами, требуя при этом затраты всего двух вентилей на каждый разряд регистра (для сдвига в одном направлении). Схема двух соседних разрядов сдвигового регистра, осуществляющего сдвиг кода вправо, изображена на рис. 29.

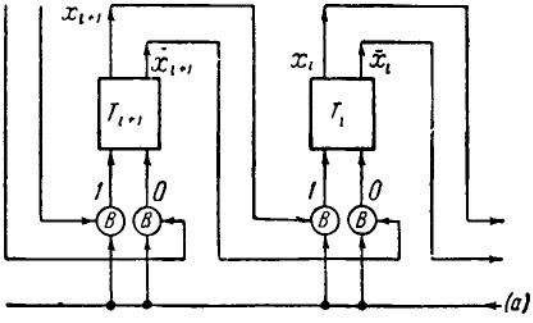

Рис. 29

Подача импульса на вход (*a*) схемы, изображенной на этом рисунке, переводит триггер T_i в то состояние, в котором перед этим находился триггер T_{i+1}.

Рассмотрим теперь одну из возможных схем реализации операции умножения. Предположим, для определенности, что речь идет об умножении правильных двоичных n-значных дробей, причем результат умножения также должен быть округлен до n двоичных разрядов. При таком предположении исключается возможность выхода из имеющегося числа разрядов, поскольку при перемножении правильных дробей в результате снова получается правильная дробь. Случай умножения целых двоичных чисел отличается от рассматриваемого лишь некоторыми второстепенными деталями.

Выполнение операции умножения можно осуществить на арифметическом устройстве, которое кроме сумматора имеет еще два регистра, предназначенных для хранения множимого и множителя. Регистр самого сумматора служит для хранения сумм частичных произведений. На нем получается также и окончательный результат. На регистре множителя и на сумматоре можно реализовать микрооперацию правого сдвига. Микропрограмма умножения строится с помощью чередования микроопераций передачи множимого на сумматор (сложения его с ранее полученной суммой частичных произведений), сдвига (без знакового разряда) вправо кода на сумматоре и сдвига (без знакового разряда) вправо кода на регистре множителя. Первая из этих микроопераций должна, очевидно, осуществляться лишь в том случае, когда в младшем разряде множителя стоит единица. Для этой цели сигнал с выхода триггера младшего разряда регистра множителя должен передаваться в устройство, осуществляющее управление последовательностью микроопераций. После n-кратного повторения указанной последовательности микроопераций на сумматоре будет получен, как нетрудно видеть, код (округленного) произведения, знак которого совпадает со знаком множимого. Для получения истинного знака произведения необходимо в заключение выполнить еще одну микрооперацию, осуществив сложение (по модулю 2) знаковых разрядов содержимого сумматора и регистра множителя. Такая микрооперация выполняется на обычном одноразрядном сумматоре без использования сигнала переноса на его входе. Такой сумматор без переноса на входе называется обычно *полусумматором*. При наличии счетного входа в триггере знакового разряда сумматора такой полусумматор требует всего один вентиль, управляемый выходным сигналом знакового разряда регистра множителя. На этот вентиль должен быть подан импульс от устройства управления микрооперациями в момент, когда необходимо выполнить микрооперацию сложения знаков множителя и произведения.

Блок-схема АУ, на котором можно выполнять операции сложения, вычитания и умножения, показана на рис. 30. Это АУ должно иметь следующий набор микроопераций:

1) передача числа из ЗУ на регистр P_1;
2) передача числа из ЗУ на регистр P_2;
3) передача числа из сумматора в ЗУ;
4) передача числа из регистра P_1 в сумматор (суммирование);
5) сдвиг кода (без знакового разряда) на сумматоре вправо;
6) сдвиг кода (без знакового разряда) на регистре P_2 вправо;
7) сложение знаковых разрядов на сумматоре и регистре P_2;
8) инвертирование кода на регистре P_1;
9) очистка АУ (установка всех регистров в нуль);
10) очистка регистра P_1.

В случае двухтактного сумматора четвертая микрооперация расщепляется на две микрооперации — поразрядного сложения и реализации переносов. В случае необходимости выполнения других операций (например, деления) указанный набор микроопераций должен быть, вообще говоря, пополнен. Каждой из перечисленных микроопераций на блок-схеме АУ соответствует стрелка с номером микрооперации. Выходящая из регистра P_2 стрелка, помеченная буквой q, обозначает упомянутый выше сигнал обратной связи из младшего разряда этого регистра в устройство управления микрооперациями.

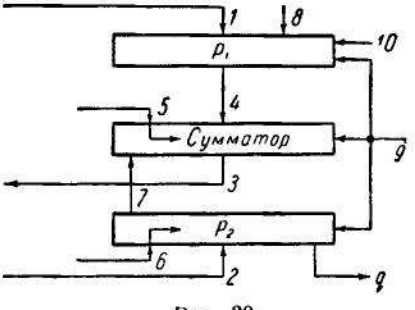

Рис 30.

Все сигналы, управляющие микрооперациями, в рассматриваемом случае предполагаются импульсными, а сигнал обратной связи q — потенциальным. На каждую микрооперацию затрачивается один физический импульсный канал от устройства управления микрооперациями.

Управление всеми микрооперациями, кроме микроопераций 1), 2) и 3), было уже описано выше. Что же касается этих последних микроопераций, то они могут быть

реализованы следующим образом. Число, передаваемое из ЗУ, хранится на специальном триггерном выходном регистре ЗУ, который мы обозначим через P_3. Выходной сигнал каждого разряда этого регистра управляет двумя вентилями, на которые подаются импульсы, управляющие микрооперациями 1) и 2). Выход первого вентиля подсоединен к единичному входу триггера соответствующего разряда регистра P_1, выход второго — к единичному входу триггера соответствующего разряда регистра P_2. Аналогичным образом строится управление передачей числа из сумматора в регистр P_3.

При описанной организации управления микрооперациями перед передачей числа на какой-либо регистр этот регистр должен быть предварительно очищен (установлен в нуль). Заметим также, что с целью уменьшения числа каналов связи АУ с ЗУ, по которому передаются коды чисел (эти каналы называются кодовыми шинами), целесообразно либо иметь несколько иную, более сложную систему управления микрооперациями, либо осуществлять передачу чисел из ЗУ лишь на один регистр АУ, добавляя микрооперации передачи кодов с одного регистра на другой внутри АУ.

Приведем в качестве примера последовательность микроопераций (микропрограмму) для управления операциями в блок-схеме АУ, изображенной на рис. 30, без учета микроопераций, необходимых для управления другими блоками машины. При этом будем предполагать, что все числа в машине представляются обратными кодами, но в регистр P_2 число передается прямым кодом.

Микропрограмма сложения двух чисел:

$$9, 1, 4, 10, 1, 4, 3.$$

Микропрограмма вычитания двух чисел:

$$9, 1, 4, 10, 1, 8, 4, 3.$$

Микропрограмма умножения двух n-разрядных чисел:

$$9, 1, 2, \underbrace{4\ (\text{если } q=1), 5, 6, \ldots, 4\ (\text{если } q=1), 5, 6}_{n \text{ раз}}, 7, 3.$$

Заметим, что в микропрограмме умножения можно совместить во времени (в одной микрокоманде) выполнение двух микроопераций, а именно, микроопераций 5) и 6). Ясно также, что последний (n-й) раз микрооперацию 6) сдвига на регистре P_2 можно не выполнять.

§ 3. Организация управления универсальным программным автоматом

В предыдущем параграфе был рассмотрен вопрос об организации управления арифметическими операциями в АУ. Целью настоящего параграфа является рассмотрение вопроса об организации управления переработкой информации во всем универсальном программном автомате в целом. Как и раньше, мы рассмотрим более подробно лишь один из возможных вариантов такой организации, дающий представление об основных проблемах, которые возникают в этой области, а также об общих принципах их решения.

С целью упрощения задачи мы полностью исключим из рассмотрения вопросы управления внешним запоминающим устройством, вводом и выводом, а также вопросы рациональной организации переадресации команд (групповые операции, индекс-регистры и т. п.), хотя, используя развиваемые далее общие принципы, не представляет особого труда дать решение и этих вопросов.

Поскольку в организации управления всем автоматом большая роль принадлежит оперативному запоминающему устройству (ОЗУ), необходимо прежде всего остановиться на некоторых деталях его строения. Всякое оперативное запоминающее устройство состоит из собственных запоминающих элементов, образующих *ячейки памяти*, из *дешифратора адреса* и двух регистров, называемых — один *регистром числа* (или кодовым регистром), а другой — *регистром адреса*.

Каждая ячейка ОЗУ представляет собою регистр, в котором хранится одно информационное или командное слово. Однако лишь в самых первых программных автоматах в качестве ячеек ОЗУ использовались обычные триггерные регистры. В настоящее время эти ячейки организуются гораздо более экономными способами. При

этом регистр, представляющий собою ячейку ОЗУ, коренным образом отличается от триггерного регистра не только по своей конструкции, но и по характеру своего функционирования.

Основное различие заключается в том, что информация в запоминающих ячейках ОЗУ хранится обычно в таком виде, что она не может быть непосредственно использована для управления комбинационными схемами (особенно потенциальными), как это имеет место в случае информации, хранящейся в триггерном регистре. Для того чтобы выступить в такой роли, информация, записанная в той или иной ячейке ОЗУ, должна быть предварительно *считана*, то есть передана на триггерный или иной регистр, имеющий выходы, которые могут непосредственно использоваться для управления комбинационными схемами. Регистры, обладающие такими свойствами, естественно назвать *управляющими*, *активными* или *открытыми* регистрами, в противоположность *закрытым* (неуправляющим) регистрам, которыми являются ячейки ОЗУ.

После записи информации в закрытый регистр (ячейку ОЗУ) она временно как бы исчезает из поля зрения активной зоны автомата и снова попадает в него лишь после ее считывания и передачи на управляющий регистр. Регистр числа в ОЗУ как раз и представляет собою такой активный (управляющий) регистр, на который попадает каждое слово (информационное или программное) непосредственно после его считывания из любой ячейки ОЗУ или непосредственно перед его записью в произвольную ячейку ОЗУ.

Для определения того, из какой именно ячейки ОЗУ нужно считать или в какую именно ячейку нужно записать слово, хранящееся на регистре числа, служит регистр адреса и дешифратор адреса ОЗУ. Регистр адреса представляет собою активный (управляющий) регистр, предназначенный для запоминания адреса ячейки ОЗУ, из которой нужно считать или в которую нужно записать информацию. Выходные сигналы запоминающих элементов регистра адреса поступают на входные узлы дешифратора адреса, а выходные сигналы дешифратора используются для того, чтобы открыть путь

управляющим сигналам считывания или записи в выбранную ячейку ОЗУ.

Сигналы считывания и сигналы записи осуществляют передачу кода слова из выбранной ячейки памяти в регистр числа и наоборот. Мы условимся считать, что в самом ОЗУ выполняются лишь эти две микрооперации — передача кода из выбранной ячейки в регистр числа (микрооперация считывания) и обратная передача кода из регистра числа в ячейку (микрооперация записи), а также микрооперация очистки регистров ОЗУ, состоящая в установке на нуль регистра числа и регистра адреса. Микрооперации же обмена кодами между регистрами ОЗУ и другими блоками автомата мы будем относить к тем блокам, с которыми этот обмен осуществляется.

Разъяснение технических подробностей устройства современных ОЗУ не входит в задачу настоящей книги. При желании читатель может представлять себе ячейки ОЗУ в виде обычных триггерных регистров. В этом случае нетрудно представить себе полную схему всего ОЗУ. Действительно, предположим, что регистры числа и регистры адреса построены на обычных потенциальных триггерах, а дешифратор адреса — на потенциальных совпадениях и разделениях, что выходные сигналы дешифратора носят потенциальный характер, а число их равняется числу ячеек памяти (оперативного запоминающего устройства).

Каждый из этих сигналов используется как управляющий сигнал для вентилей записи, установленных на входах всех триггеров данной ячейки (регистра) памяти, соответствующей этому сигналу. Управляющий сигнал записи проходит через вентили, управляемые выходными сигналами триггера регистра числа (РЧ), а отсюда уже n импульсных сигналов, несущих n-разрядный код числа из РЧ, поступают на соответствующие входы вентилей записи. Каждый из этих сигналов поступает одновременно на вентили записи соответствующего разряда всех ячеек памяти, но проходит лишь через вентиль той ячейки, которая выбрана дешифратором. Разумеется, в этом случае перед записью необходимо произвести очистку выбранной ячейки.

Аналогичным образом можно осуществить также считывание кода из выбранной ячейки: импульс считывания подается на вентили, управляемые выходными сигналами дешифратора, и через тот (единственный) вентиль, который будет открыт в данный момент, поступает на входы вентилей, управляемых выходными сигналами триггеров выбранной дешифратором ячейки памяти. Выходы этих вентилей подсоединены к входам триггеров (соответствующих разрядов) регистра числа, куда и передается код из выбранной ячейки. Ясно, что перед считыванием регистр числа должен быть очищен (установлен на нуль).

Описанная схема ОЗУ, как уже отмечалось выше, чрезвычайно неэкономична и в настоящее время при построении ОЗУ большой емкости не применяется. Поэтому мы не будем иллюстрировать произведенное описание рисунками соответствующих схем. Заметим, однако, что проведенное выше описание дает известное представление о специфике проблем, возникающих при конструировании и организации функционирования современных схем ОЗУ. Наибольшую трудность при этом представляет не построение самих ячеек ОЗУ, а организация их коммутирования (выборки), поскольку дешифраторы с большим числом выходных каналов оказываются чрезвычайно громоздкими.

Переходя к описанию методов синтеза устройств управления универсальных программных автоматов, рассмотрим прежде всего одно важное устройство, входящее в качестве составной части почти во все существующие схемы УУ. Речь идет о так называемом импульсном счетчике.

Таблица VII 1

	1	2	3	..	$n-1$	n
0	1	2	3	...	$n-1$	n
u	2	3	4	...	n	1

Импульсным счетчиком, или счетчиком импульсов по модулю n, называется цифровой автомат Мура с n состояниями, имеющий один физический импульсный входной

канал и задаваемый таблицей переходов VII.1. Наличие импульса во входном канале обозначено в таблице входным сигналом u, а отсутствие импульса — сигналом 0. Каждому выходному сигналу счетчика должен соответствовать свой собственный выходной сигнал.

Схемы счетчиков существенно зависят от принятого способа кодирования их состояний. В счетчике с *естественным кодированием состояний* последовательные (переходящие друг в друга под действием одного импульса) состояния кодируются обычными двоичными кодами последовательных целых неотрицательных чисел, начиная с нуля:

$$00\ldots00,\ 00\ldots01,00\ldots10,00\ldots11.$$

При этом число разрядов кода выбирается наименьшим, то есть равным наименьшему целому числу m, которое больше или равно двоичному логарифму числа n состояний автомата:

$$m = \min k \quad \text{где} \quad k \geqslant \log_2 n.$$

В так называемом *кольцевом счетчике* с n состояниями последовательные состояния кодируются n-разрядными двоичными кодами, каждый из которых состоит из $n-1$ нулей и одной единицы:

$$100\ldots0,\ 010\ldots0,001\ldots0,\ldots,000\ldots01.$$

Схемы кольцевых счетчиков по сравнению со счетчиками с естественным кодированием состояний содержат больше запоминающих, но зато, как правило, меньше логических элементов.

Договоримся в дальнейшем, в целях краткости, если не оговорено противное, понимать под термином «счетчик» импульсный счетчик с естественным кодированием состояний. Покажем, как может быть построена схема счетчика по модулю n с использованием импульсно-потенциальной логики. Основу счетчика составляет m-разрядный регистр на потенциальных триггерах. Для упрощения рассуждений предположим, что каждый триггер имеет как счетный вход (обозначаемый буквой f), так и раздельные входы (обозначаемые цифрами 0 и 1) для установки

триггера, соответственно в 0 и 1. Каждый из этих трех входов снабжается индексом, соответствующим номеру (от 1 до m) рассматриваемого триггера, так что входы i-го триггера обозначаются через 0_i, 1_i и f_i соответственно. Состояния триггеров мы будем обозначать через $x_1, \ldots, x_m$. Введем еще обозначение K_n для $(n-1)$-й конституенты единицы от переменных $x_1, \ldots, x_n$, обозначение $\bar{K}_n$ — для ее отрицания и обозначение p для входного канала, по которому подаются считываемые счетчиком импульсы.

Потенциальный сигнал, равный K_n, можно образовать с помощью m-входового совпадения соответствующих выходных сигналов (x_i и $\bar{x}_i$), составляющих счетчик триггеров, а потенциальный выходной сигнал, равный $\bar{K}_n$, — с помощью m-входового разделения. Заменяя, для простоты, указанные совпадение и разделение их выходными сигналами, мы получаем схему счетчика по модулю n,

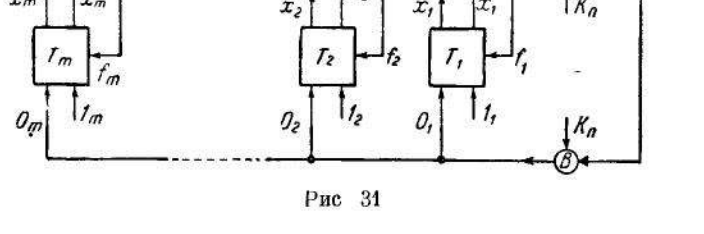

Рис 31

изображенную на рис. 31. В этой схеме верхний ряд вентилей образует цепь, вполне аналогичную той цепи реализации переноса, возникшего в самом младшем разряде, в схеме сумматора, изображенной на рис. 27 из § 2 настоящей главы. С помощью этой цепи поступление каждого нового импульса на вход p вызывает увеличение на единицу двоичного кода, хранящегося на триггерах $T_1 — T_m$. Как только этот код достигает значения $n-1$, открывается вентиль K_n, и очередным поступившим импульсом p все триггеры перебрасываются в нулевое состояние. Вентиль $\bar{K}_n$ в этот момент будет закрыт, так что распространение импульса по верхней цепи, которое могло бы помешать установке триггеров на нуль, исключается.

В ряде случаев счетчики строятся таким образом, что в процессе работы они никогда не достигают максимального значения $(n-1)$ запоминаемого ими кода. Тогда цепь автоматического сброса в нуль и вентили K_n, $\overline{K}_n$ становятся излишними. Такие упрощенные счетчики мы будем называть *незамкнутыми*,— в противоположность *замкнутому*, или *циклическому*, счетчику, о котором шла речь выше. Заметим, что как циклические, так и незамкнутые счетчики допускают возможность установки их в нулевое состояние вручную за счет посылки специального импульса с пульта управления.

Следуя принятому нами методу изложения, мы рассмотрим теперь общие проблемы синтеза УУ в связи с каким-либо частным примером.

Предположим, что требуется построить трехадресный универсальный программный автомат с естественным порядком следования команд, у которого i-я команда программы запоминается в $(a+i)$-й ячейке памяти ($i=1,2,..$, a—любое неотрицательное число). Будем считать, что ОЗУ и АУ этого автомата построены в соответствии с принципами, изложенными в настоящем параграфе, в частности, что они имеют импульсное управление микрооперациями.

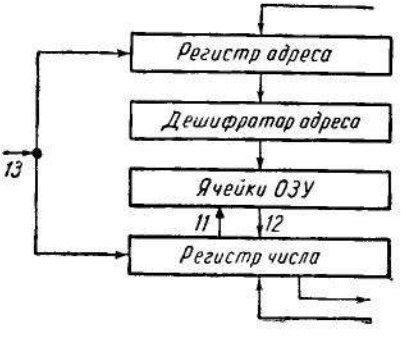

Рис. 32

В качестве АУ для нашего автомата примем АУ, изображенное на рис. 30 (со всеми указанными микрооперациями), дополнив его кроме потенциального сигнала обратной связи q потенциальным сигналом обратной связи p, принимающим значения, запоминаемые в знаковом разряде сумматора.

Блок-схема ОЗУ, которую мы применим в нашем автомате, изображена на рис. 32. Стрелками на этом рисунке показаны направления передачи информации. Стрелкам с номерами 11, 12, 13 соответствуют микрооперации

записи, считывания и очистки регистров ОЗУ. Только эти микрооперации считаются микрооперациями собственно ОЗУ.

Задачей устройства управления является, во-первых, управление последовательностью микроопераций в АУ и в ОЗУ, а во-вторых, управление последовательностью собственных микроопераций. Для установления набора микроопераций УУ зафиксируем прежде всего количество и характер работы используемых в нем регистров.

Важнейшим из регистров УУ (и всего автомата в целом) является так называемый *регистр команд* (РК). При трехадресной системе команд этот регистр составляется фактически из четырех регистров: *регистра операций* (РО), на котором запоминается код операции выполняемой команды, а также регистров первого, второго и третьего адреса ($РА_1$, $РА_2$, $РА_3$), на которых запоминается соответственно первый, второй и третий адреса выполняемой команды.

Важной составной частью УУ является также *счетчик команд* (СК), служащий для запоминания адреса ячейки ОЗУ, из которой должна извлекаться очередная команда программы. Наконец, имеется еще *регистр микроопераций* (РМО), заменяемый иногда *счетчиком микротактов* (СМТ). О назначении этих устройств мы скажем ниже. Счетчик микротактов осуществляется всегда как циклический (замкнутый) счетчик; что же касается счетчика команд, то он может быть и незамкнутым.

Как следует из общего описания принципа программного управления, сделанного в § 1 настоящей главы, управляющее устройство (УУ), чтобы обеспечить управление работой всех устройств (включая и самого себя), должно осуществлять автоматическую выборку команд из ОЗУ, а также их расшифровку и выполнение. С этой целью в набор микроопераций самого УУ должны быть включены следующие микрооперации:

14) передача кода (программного слова) из регистра числа ОЗУ на регистр команд;

15) передача первого адреса из РА в регистр адреса ОЗУ;

16) передача второго адреса из РА в регистр адреса ОЗУ;

17) передача третьего адреса из РА в регистр адреса ОЗУ;

18) посылка импульса в счетчик команд (увеличение номера команды на 1);

19) очистка регистра команд и регистра микроопераций;

20) передача третьего адреса из РА в счетчик команд;

21) передача содержимого счетчика команд на регистр адреса ОЗУ.

Микрооперация 20 дает возможность реализовать операцию условного перехода.

Для организации управления последовательностью микроопераций строится конечный автомат M, памятью которого служит регистр микроопераций. Выходные сигналы этого автомата представляют собою не что иное, как импульсы управления микрооперациями, посылаемые по одному или одновременно по нескольким из введенных выше двадцати одного канала управления микрооперациями (с изменением набора микроопераций меняется, вообще говоря, и число этих каналов).

Входными сигналами автомата M служат потенциальные сигналы обратной связи, подаваемые из АУ (в нашем примере сигналы p и q), выходные сигналы триггеров регистра операций (операционные входы). Кроме того, имеется один импульсный тактирующий вход, задающий *тактировку* (разбиение времени на микротакты) для автомата M и всего универсального программного автомата в целом. В нашем примере мы рассмотрим случай синхронного автомата, у которого импульсные тактирующие сигналы вырабатываются специальным *синхронизирующим генератором*, работающим с постоянной частотой.

Нетрудно понять, что характер входных и выходных сигналов автомата M связан со спецификой рассматриваемого примера. Вовсе необязательно, разумеется, чтобы выходные сигналы были импульсными, а входные — потенциальными. Необязательным является и применение синхронизирующего генератора. Можно построить автомат M как асинхронный автомат, который, выдав выходной сигнал для производства очередной микрооперации, ждет ответного сигнала от соответствующего устройства, свидетельствующего об окончании выполнения микрооперации.

Эти ответные сигналы и будут играть роль тактирующих сигналов. Длительность микротактов в этом случае будет, вообще говоря, уже неодинаковой.

Задача организации управления микрооперациями любого универсального программного автомата Q есть задача синтеза соответствующего автомата M, который мы будем называть *микропрограммным блоком*, или *микропрограммным подавтоматом* автомата Q. Синтез микропрограммного блока в общем случае осуществляется с помощью методов синтеза автоматов, развитых в предыдущих главах.

Применительно же к рассматриваемому примеру мы используем сейчас один специальный метод синтеза, предложенный Уилксом и Стринджером [1]). Этот метод приводит к весьма прозрачной— хотя далеко не к самой экономной — схеме микропрограммного блока. Основу схемы Уилкса — Стринджера составляют две диодные матрицы, называемые *А-матрицей* и *В-матрицей*. Каждая из этих матриц состоит из некоторого числа вертикальных и некоторого числа горизонтальных шин (проводников), по которым осуществляется передача импульсов. Импульсы с выходов специального *микропрограммного дешифратора* поступают на горизонтальные шины A-матрицы, причем в каждый данный момент дешифратор открывает путь импульсам только на одну из горизонтальных шин. С этой шины, которую мы будем называть выбранной, через специальные элементы, пропускающие импульсы только в одном направлении (так называемые диоды), импульсы поступают на одну или несколько вертикальных шин A-матрицы. Каждая из таких вертикальных шин управляет какой-нибудь одной микрооперацией: от нее по *шине управления* соответствующей микрооперацией импульс поступает в тот блок автомата, в котором эта микрооперация должна быть в данный момент выполнена.

Каждая горизонтальная шина A-матрицы соединена с одной или с двумя горизонтальными шинами B-матрицы,

---

[1]) M. V. Wilkes, J. B. Stringer, Microprogramming and the design of the control circuits in an electronic digital computer Proc. Cambridge Philos. Soc., v. 49, № 4 : 2, 1953, p. 230—238.

причем в первом случае соединение осуществляется непосредственно, а во втором случае — через вентили, один из которых управляется каким-нибудь потенциальным сигналом обратной связи (в нашем примере сигналом p или q), а другой — отрицанием (инверсией) этого сигнала. В любом случае импульс с выбранной горизонтальной шины A-матрицы попадает в точности на одну горизонтальную шину B-матрицы, которую в этом случае мы также будем называть в ы б р а н н о й. Вертикальные шины B-матрицы группируются в пары; каждая пара шин подсоединяется к входам (нулевому и единичному) какого-либо (своего для каждой пары) триггера р е г и с т р а м и к р о о п е р а ц и й.

Таким образом, число пар вертикальных шин в B-матрице равно числу разрядов (двоичных) в регистре микроопераций. Каждая горизонтальная шина через диод соединяется с одной из шин (нулевой или единичной) каждой пары вертикальных шин. Через эти диоды импульс с выбранной горизонтальной шины поступает на вертикальные шины и устанавливает на регистре микроопераций новый код, определяемый способом включения диодов между выбранной горизонтальной шиной и соответствующими вертикальными шинами (то есть тем, подсоединены ли они к нулевым или к единичным шинам).

Дешифратор микроопераций управляется выходными сигналами триггеров регистра операций и регистра микроопераций. Комбинация кодов на этих регистрах полностью определяет, какая из горизонтальных шин A-матрицы будет выбрана в настоящий момент. Сам дешифратор может быть выполнен на потенциальных элементах совпадения по любой из схем, рассматривавшихся выше (см. § 2, гл. V). Каждый из выходов этого дешифратора управляет вентилем, выход которого подсоединен к соответствующей горизонтальной шине A-матрицы.

На импульсные входы всех этих вентилей непрерывно подаются *синхронизирующие импульсы* (СИ) от тактирующего генератора, однако в каждый данный момент очередной синхронизирующий импульс может пройти только через один вентиль, выбранный дешифратором в соответствии с кодами, установленными на регистрах операций и микроопераций. Дешифратор микроопераций можно,

разумеется, строить и из одних лишь вентильных элементов, не употребляя потенциальных элементов.

На рис. 33 схематически изображен описанный нами микропрограммный блок. Для простоты мы изобразили матрицы A и B со сравнительно небольшим числом вертикальных и горизонтальных шин. На практике число шин

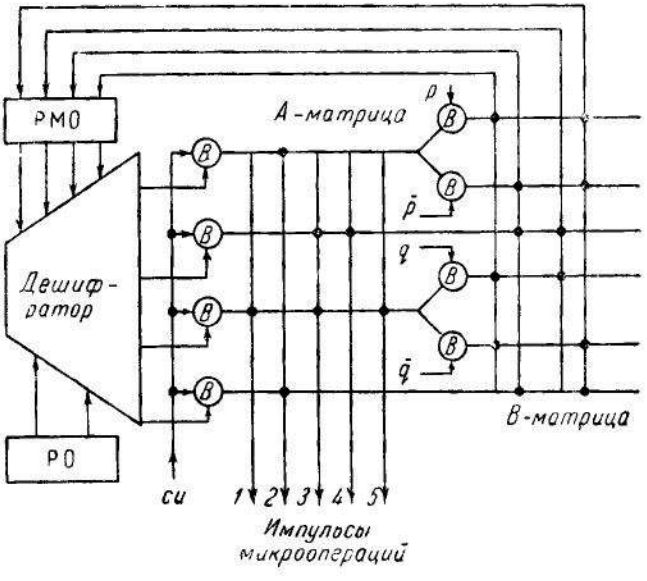

Рис. 33.

бывает обычно значительно бо́льшим. Так, в рассматриваемом нами примере число вертикальных шин A-матрицы должно быть равно 21 (то есть числу различных микроопераций). Точками на матрицах A и B в схеме, изображенной на рис. 33, показаны места включения диодов, соединяющих горизонтальные шины с вертикальными.

Из рассмотрения рис. 33 видно, что в матрицах A и B диоды нельзя заменить простыми соединениями горизонтальных и вертикальных шин, так как при этом могут возникнуть ложные импульсы микроопераций. В самом

деле, при поступлении импульса на вторую (сверху) горизонтальную шину импульсы, согласно задуманному плану соединений, должны передавать лишь на 3-ю и 4-ю вертикальные шины. Однако при непосредственном соединении шин импульс с 3-й вертикальной шины перешел бы на 3-ю горизонтальную шину, а с нее распространился бы на 1-ю и на 5-ю вертикальные шины.

Назначение диодов как раз и состоит в том, чтобы, пропуская беспрепятственно импульсы с горизонтальных шин на вертикальные, лишить их возможности обратного перехода на горизонтальные шины. Ясно, что при соединении шин диодами импульсы будут переходить на те и только на те вертикальные шины, которые соединены с выбранной горизонтальной шиной. Меняя эти соединения, можно менять микропрограмму автомата, то есть изменять последовательности микроопераций, соответствующих любому данному коду на регистре операций. Таким образом, можно, в частности, не меняя схему всего автомата в целом, путем одного лишь пересоединения диодов в матрицах A и B изменить набор операций данного программного автомата.

Каждая горизонтальная шина A-матрицы соответствует одной микрокоманде микропрограммы (множеству микроопераций, выполняемых за один микротакт), а соединенная с ней горизонтальная шина B-матрицы определяет микрокоманду, которая будет выполняться в течение следующего микротакта. Наличие вентилей, управляемых сигналами обратной связи (p и q), исходящими из АУ, позволяет осуществлять в микропрограмме ветвления, то есть производить выбор в качестве следующей команды одной из двух команд в зависимости от значения соответствующего сигнала обратной связи.

При построении микропрограммы можно различным образом строить начало (и соответственно также конец) рабочего цикла машины. Можно начинать каждый цикл с момента, когда код очередной команды уже хранится на регистре команд (РК) и заканчивать его выборкой из ОЗУ и передачей на РК очередной команды. Но мы примем сейчас другой вариант, отличающийся тем, что рабочий цикл машины начинается с выборки из ОЗУ и

передачи на РК очередной команды программы, а заканчивается изменением содержимого счетчика команд (то есть установкой на нем адреса следующей команды) и очисткой АУ регистра команд, регистра микроопераций и обоих регистров ОЗУ.

В начале работы адрес ячейки, в которой хранится первая команда программы, устанавливается на счетчике команд вручную, все же остальные циклы выполняются машиной автоматически до тех пор, пока в какомлибо из рабочих циклов ей не встретится команда «останов». Приняв код этой команды на РК, машина автоматически останавливается. Возможна также остановка машины по специальному сигналу АУ, выдаваемому в случае выхода из используемого числа разрядов.

Приведем теперь полные микропрограммы для некоторых операций, в которых учтены микрооперации, выполняемые в течение полного рабочего цикла выполнения данной операции во всех блоках машины. Для компактности записи микропрограмм используем обозначение микроопераций номерами, фиксированными в настоящем и в предыдущем параграфе.

Микрооперации, выполняемые за один микротакт (то есть совместимые во времени), мы будем заключать в скобки.

Напомним, что выполнение микропрограммы каждой операции (команды) начинается при нулевом заполнении АУ, регистра команд, регистра микроопераций и обоих регистров ОЗУ и при наличии адреса соответствующей команды в счетчике команд.

Микропрограмма сложения двух чисел (в трехадресной системе):

(21) (12) (14) (13) (15) (12) (1) (4,13) (16) (12,10) (1) (13,4) (17, 3) (11,5) (18, 13, 19) (выполняется за 19 микротактов).

Микропрограмма условного перехода по неравенству двух чисел:

(21) (12) (14) (13) (15) (12) (1) (4,13) (16) (12,10) (1) (13,8) (4), если $p=0$, то (18, 9, 19), а если $p=1$, то (20, 9, 19).

Команда условного перехода по неравенству выполняется, в силу приведенной микропрограммы, за 14 микротактов. Суть этой команды состоит в том, что из числа a,

хранящегося в ячейке по первому адресу команды, вычитается число b, хранящееся по второму адресу команды. Если эта разность положительная ($a>b$), то выполняется следующая по порядку команда, если же разность этих чисел не положительна ($a\leqslant b$), то следующая команда извлекается из ячейки с адресом, указанным в третьем адресе команды. Заметим, что условный переход по совпадению (равенству) двух чисел, о котором говорилось в § 1 настоящей главы, может быть выполнен в результате последовательного выполнения двух команд условного перехода по неравенству.

Отметим еще некоторые особенности организации микропрограммного управления в универсальных программных автоматах.

Описанные выше микропрограммы исходят из того, что на выполнение любой из имеющихся 21 микрооперации отводится одно и то же время (время одного микротакта). В действительности, однако, различные микрооперации требуют различного времени для своего выполнения. Действуя, как было описано выше, мы должны время микротакта определять по самой длительной микрооперации. Тем самым мы искусственно снижаем рабочую частоту и понижаем, следовательно, быстродействие автомата.

Выход из этого положения заключается в том, чтобы осуществлять длительные микрооперации в течение нескольких микротактов. В микропрограмме для этой цели вводятся пустые микрокоманды. При выполнении такой микрокоманды импульс поступает на шину A-матрицы, не соединенную с вертикальными шинами, в результате чего не выполняется ни одна из микроопераций, за исключением изменения содержимого регистра микроопераций через B-матрицу.

Выделение необходимого времени для производства длительной микрооперации производится с помощью дописывания в микропрограмме после соответствующей ей команды необходимого числа пустых микрокоманд. Можно, разумеется, вместо пустых микрокоманд использовать для той же цели (полностью или частично) микрокоманды других микроопераций микропрограммы, совместимых во времени с рассматриваемой длительной микрооперацией.

Из указанного выше набора микроопераций к числу длительных микроопераций относятся обычно считывание и запись в ОЗУ, а также микрооперация суммирования (точнее, реализация переносов).

В силу указанных соображений оказывается целесообразным уточнить ранее введенное понятие микрооперации, *называя микрооперацией действие, производимое в универсальном программном автомате, одним импульсом, посылаемым по любому выходному каналу микропрограммного блока*. С точки зрения указанного выходного канала микрооперация, как и при прежнем определении, будет выполняться фактически за один микротакт (ибо до конца выполнения микрооперации изменений состояния этого канала не происходит), однако с точки зрения всей машины в целом она требует для своего выполнения нескольких микротактов.

При описанном способе построения микропрограммного блока время рабочего цикла автомата будет, вообще говоря, различным для различных операций. В ряде случаев, особенно когда имеют дело с операциями примерно одинаковой длительности, идут на то, чтобы уравнять время рабочего цикла для всех операций (ориентируясь, разумеется, на самую длительную операцию).

В этом случае в микропрограммном блоке можно обойтись без B-матрицы, заменив регистр микроопераций *счетчиком микротактов*. Этот счетчик осуществляет счет по модулю n (где n — число микротактов в рабочем цикле) импульсов, подаваемых на его вход непосредственно с выхода тактирующего генератора. Матрица A остается точно такой же, как и раньше, а управления дешифратором микроопераций по-прежнему осуществляется выходными сигналами триггеров регистра операций и счетчика микротактов, к которым присоединяются теперь еще сигналы обратной связи от АУ (p и q в нашем примере). Для согласования длительности микропрограмм различной длины употребляются, как и выше, пустые микрокоманды.

В заключение отметим, что методы синтеза сложных автоматов, развитые в настоящей главе, применимы лишь к одному из классов автоматов, а именно к классу автоматов с программным управлением, имеющим, к тому же,

вполне определенную алгоритмическую (блочную) структуру. При построении различного рода специализированных цифровых автоматов эти методы часто оказываются непригодными, так как приводят к неоправданно сложным схемам. Они никоим образом не отменяют общих методов, развитых в предыдущих главах, равно как не отменяют и необходимости поиска новых методов блочного синтеза, основанных на алгоритмических структурах, отличных от алгоритмических структур универсальных программных автоматов.

ЛИТЕРАТУРА

Ангер С. (Unger S. H.), Hazards and delays in asynchronous sequential switching circuits. IRE Trans., v. CT-6, 1959, p. 12—25.

Ауфенкамп Д. (Aufenkamp D. D.), Analysis of sequential machines, II. IRE Trans., v. EC-7, № 4, 1958, p. 299—306 (русск. перев. в периодическом сборнике переводов иностранных статей «Математика», 1959, № 3 : 6, стр. 145—158).

Ауфенкамп Д и Хон Ф. (Aufenkamp D. D. and Hohn F. E.), Analysis of sequential machines, IRE Trans., v. EC-6, № 4, 1957, p. 276—285 (русск. перев. в периодическом сборнике переводов иностранных статей «Математика», 1959, № 3 : 3, стр. 129—146).

Базилевский Ю. Я., Вопросы теории временны́х логических функций. Вопросы теории математических машин, сб. 1, Физматгиз, М., 1958, стр. 9—37.

Бёркхарт В. (Burkhart W. H.), Theorem minimization, Proc. Ass. Comp. Mach., May, № 2 and 3, 1952, p. 259—263.

Бёркс А. (Bürks A. W.), The logic of fixed and growing automata. Intern. Sympos. on the Theory of Switching (Proc. 1957), 1959, v. 1, p. 147—188.

Бёркс А. и Ванг Х. (Bürks A. W and Wang H.), The logic of automata. J. Ass. Comp. Mach., 4, 1957, № 2, p. 193—218, N 3, p. 279—297.

Бёркс А. и Копи Дж. (Bürks A. W. and Copi J. M.), The logical design on an idealized general purpose computer. J. Frankl. Inst., v. 261, 1956, p. 299—314, 421—436.

Бёркс А. и Райт Дж. (Bürks A. W. and Wright J. B.), Theory of logical nets. Proc. IRE, 1953, v. 41, N 10, p. 1357—1365.

Блейк А. (Blake A.), Canonical expression in Boolean Algebra, Dissertation, Chicago, 1937.

Вейч Е. (Veitch E. W.), A chart method for simplifying truth functions. Proc. Ass. Comp. Mach., 1952, May, № 2, and 3, p. 127—133.

Гаврилов М. А., Теория релейно контактных схем. Изд-во АН СССР, М., 1950.

Гинзбург С. (Ginsburg S.), On the reduction of superfluous states in a sequential machine. J. Ass. Comp. Mach., 1959, v. 6, N 2, p. 259—282.

Его же, A technique for the reduction of a given machine to a minimal-state machine, IRE Trans., v. EC-8, N 3, 1959, p. 346—355.

A synthesis technique for minimal state sequential machines, IRE Trans., v. EC-8, № 1, p. 13—24 (русск. перев. в периодическом сборнике переводов иностранных статей «математика», 1960, № 4 : 4, стр. 145—168).
A synthesis of minimal state machines, IRE Trans., v. EC-8, № 4, 1959, p. 441—448.

Гланц (Glantz H. T.), A note of microprogramming, J. Ass. Comp. Mach., v. 3, № 2, 1956, p. 77—84.

Глушков В. М., Об одном алгоритме синтеза конечных автоматов. Укр. матем журнал, т. 12, № 2, 1960, стр. 147—156.
Два универсальных критерия эффективности вычислительных машин. ДАН УССР, № 4, 1960, стр. 477—481.
Об одном методе анализа абстрактных автоматов. ДАН УССР, т. 12, № 9, 1960, стр. 1151—1154.
Некоторые проблемы синтеза цифровых автоматов. Вычислит. математика и матем. физика, т. I, № 3, 1961, 371—411.

Жегалкин И. И., О технике вычисления предложений в символической логике. Матем. сб., т. 34, 1927, стр. 9—28.

Закревский А. Д., Операторный метод синтеза алгоритмических систем. Изв. высш. учебн. заведений, Радиофизика, т. 2, № 2, 1959, стр. 306—315.
К синтезу логических многополюсников. Изв. высших учебн. заведений, Радиофизика, т. 2, № 5, 1959, стр. 814—817.

Игонне, Греа (Higonnet R. Grea R.), Étude logique des circuits électriques et des systèmes binaires, Paris, 1955.

Кардо (Cardot C.), Quelques résultats sur l'application de l'algebra de Bool à la synthèse cirquits à relais. Annales des Telecommunications, v. 7, № 2, 1952.

Карнау (Karnaugh M.), The map method for synthesis of combinational logic circuits, Trans. AIEE, v. 72, № 1, 1953, p. 593—599.

Карцев М. А., Арифметические устройства электронных цифровых машин. Физматгиз, М., 1958.

Кафенгст (Kaphengst H.), Eine abstrakte programmgesteuerte Rechenmaschine, Zeitschr. für mathematische Logik and Grundlagen der Mathematik, B. 5, № 3—4, 1959, S. 366—379.

Квайн (Quine W. V.), The problem of simplifying of truth functions, Amer. Math. Monthly, v. 59, № 8, 1952, p. 521—531.
A way to simplify truth functions. Amer. Math. Monthly, v. 62, 1955, p. 627—631.
On cores and prime implicants of truth functions. Amer. Math. Monthly, v. 66, № 9, 1959, p. 755—760.

Кейстер, Ричи, Уошборн (Keister W., Ritchie A. E., Washburn S. N.) The design of switching circuits, New York, 1951.

Китов А. И. и Криницкий Н. А., Электронные цифровые машины и программирование. Физматгиз, М., 1959.

Клини С. К., Представление событий в нервных сетях и конечных автоматах. Сб. «Автоматы» под редакцией К. Э. Шеннона и Дж. Маккарти, ИЛ, М., 1956, стр. 15—67.

Колдуэлл (Caldwell S. H.), Switching circuits and logical design, New York, 1958.

Коллектив авторов вычислительной лаборатории Гарвардского ун-та, Синтез электронных вычислительных и управляющих схем. ИЛ, М., 1954 (перев с изд.: Cambridge, 1951).

Кобринский Н Е., Трахтенброт Б. А., О построении общей теории логических сетей. Сб «Логические исследования», Изд. АН СССР, М., 1959, стр. 352—378.

Копи, Элгот, Райт (Copi J. M., Elgot C., Wright J. B.), Realization of events by logical nets, J. Ass. Comp. Mach., v. 5, № 2. p 181—196.

Лебедев С. А., Мельников В А., Общее описание БЭСМ и методика выполнения операций. Физматгиз, М., 1959.

Лунц А. Г., Приложение матричной булевской алгебры к анализу и синтезу релейно-контактных схем. ДАН СССР, т. 70, № 3, 1950, стр. 421—423.

Синтез и анализ релейно-контактных схем с помощью характеристических функций. ДАН СССР, т. 75, № 2, 1950, стр. 201—204.

Алгебраические методы анализа и синтеза контактных схем. Изв. АН СССР (сер матем), т. 16, 1952, стр. 405—426.

Лупанов О Б., О возможностях синтеза схем из разнообразных элементов ДАН СССР, т. 103, 1955, стр. 561—563.

Об одном методе синтеза схем. Изв. высш. учебн. заведений, Радиофизика, 1958, № 1, стр. 122—140.

Мак-Класки (Mc Cluskey E.), Minimizations of boolean functions. Bell system Techn. J., v. 35, № 6, 1956, p. 1417—1444.

Iterative combinational switching networks-general design considerations. IRE Trans., v. EC-7, 1958, p. 285—291.

Мак-Нотон, Ямада (McNaughton R. F., Jamada H.), Regular expressions and state graphs for automata, IRE Trans., v. EC-9, № 1, 1960, p. 39—48.

Марков А. А., Теория алгорифмов. Труды матем. ин-та им. В А. Стеклова, т. 42, Изд. АН СССР, М., 1954.

Медведен Ю Т., О классе событий, допускающих представление в конечном автомате. Сб. «Автоматы», под редакцией К. Э. Шеннона и Дж. Маккарти, ИЛ, М., 1956, стр. 385—401.

Мерсер (Mercer D. J.), Micro-programming, J. Ass. Comp. Mach., v. 4 1957, p 157—171.

Мили (Mealy G. H.), A method for synthesizing sequential cirquits. Bell system Techn. J., v. 34, 1955, p. 1045—1079.

Мойсил Г. К., Об упрощении цепей с транзисторами, электронными лампами и криотронами. Rev. de math. pures et appl. Akad., RPR, v. 4, № 4, 1959, p. 497—554.

Мур, Умозрительные эксперименты с последовательностными машинами. Сб. «Автоматы», под редакцией К. Э. Шеннона и Дж. Маккарти ИЛ, 1956, стр. 179—212.

Мур и Шеннон (Moore E. F., Shannon C. E.), Reliable circuits using less reliable relays, J. Franklin Inst.,

v. 262, № 3, 1956, p. 191—208, № 4, p. 281—297 (русск. перев. в кн.: «Кибернетический сборник», № 1, ИЛ, М., 1960, стр. 109—148).

Мюллер (Muller D. E.), A theory of asynchronous circuits. Сб. Internat. Symp. on the Theory of switching (Proc. 1957), 1959, v. 1, p. 204—243.

Незервуд (Netherwood D. S.), Minimal sequential mashines, IRE Trans., v. EC-8, № 3, 1959, p. 339-345.

Нельсон (Nelson R. J.), Simplest normal truth functions. J. Symb. Logic, v. 20, № 2, 1955, p. 105—108.
Weak simplest normal truth functions, J symb. Logic, v. 20, № 3, 1955, p. 232—234.

Нейман Дж. (von Neumann John), The general and logical theory of automata. Celebral mechanisms in behavior. New York, 1951, p. 1—41 (русск. перев. под названием «Общая и логическая теория автоматов» в кн.: А Тьюринг, Может ли машина мыслить?, Физматгиз, М., 1960, стр. 59—101).
Вероятностная логика и синтез надежных организмов из ненадежных компонент. Сб. «Автоматы» под редакцией К. Э. Шеннона и Дж Маккарти, ИЛ, М., 1956, стр. 68—139.

Петрик (Petrick S. R.), A direct decomposition of the irredundant forms of a boolean function from the set of prime-implicants, Techn. reports Air Force Cambridge Research Center, 1956, p 56—110.

Поваров Г. Н., О синтезе контактных многополюсников. ДАН СССР, т. 94, № 6, 1954, p. 1075—1078.
Математическая теория синтеза контактных (1,k)-полюсников. ДАН СССР, т. 100, № 5, 1955, стр. 909—912.
О методике анализа симметрических контактных схем. Автоматика и телемеханика, т. 16, 1955, стр 364—366.
К математической теории синтеза контактных (1,k)-полюсников. ДАН СССР, т. 111, № 1, 1956, стр. 102—104.

Пол, Ангер (Paull M. C., Unger S. H.), Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans., v. EC-8, № 3, p. 356—367.

Поспелов Д. А., Синтез схем, работа которых описывается временными булевыми функциями. Автоматика и телемеханика, т. 21, № 10, 1960, стр. 1410—1413.

Пост (Post E.), Introduction to a general theory of elementary propositions. Amer. J. Math., v. 43, 1921, p. 163—185.

Ричардс Р. К. (Richards R. K.), Арифметические операции на цифровых вычислительных машинах. ИЛ, М., 1957.
Элементы и схемы цифровых вычислительных машин, ИЛ, М., 1961.

Свобода Ф., Синтез релейных схем при помощи машин. Автоматика и телемеханика, т. 18, № 3, 1957, стр. 240—255.

Трахтенброт Б. А., Синтез логических сетей, операторы которых описаны средствами исчисления одноместных предикатов. ДАН СССР, т. 118, № 4, 1958.

Уилкс (Wilkes M. V.), Microprogramming, Proc. East. Joint. Comput Conf., v. NT-114, New York, 1959, p. 18—20.

Уилкс, Стриндже р (Wilkes M. V., Stringer J. B.), Microprogramming and the design of the control circuits in an electronic digital computer. Proc. Cambridge Philos. Soc., v. 49, № 4 : 2, 1953, p. 230—238.

Фистер (Phister M.), Logical design of digital computers, New York, 1957.

Хафмен Д. (Huffman D. A.), The synthesis of sequential switching circuits, Journal of the Franklin Inst., v. 257, № 3 and 4, 1954, p. 161—190, 275—303.

The design and use of hazard free switching networks. Journ. Ass. Comp. Machinery, v. 4, № 1, 1957, p. 47—62.

Хон (Hohn F. E.), A matrix method for the design of relay circuits. IRE Trans., v. CT-2, № 2, 1955, p. 154—161.

Some mathematical aspects of switching. Amer. Math. Monthly, v. 62, № 2, 1955, p. 75—90.

2*N*-terminal contact networks Internat. sympos. on the theory of switching (Proc. 1957), v. 2, 1959, p. 51—58.

Хон и Шислер (Hohn F. E., Schissler L. R.), Boolean martices and the design of combinational relay switching circuits, Bell system. Techn. J., v. 34, 1955, p. 177—202.

Хэмминг (Hamming A. W.), Error detecting and error correcting codes. Bell system Techn. J., v. 29, № 2, 1950, p. 147—160.

Хэмфри (Humphrey W. S.), Switching circuits with computer applications, New York, 1958.

Цетлин М. Л., Матричный метод анализа и синтеза электронно-импульсных и релейно-контактных (непримитивных) схем, ДАН СССР, т. 117, № 6, 1957, стр. 979—892.

О композиции и разбиениях непримитивных схем. ДАН СССР, т. 118, № 3, 1958, стр. 488—491.

О непримитивных схемах. Проблемы кибернетики, вып. 1, Физматгиз, М., 1958, стр. 23—45.

Цетлин М. Л., Эйдус Г. С., Алгебраический метод синтеза схем на триггерных ячейках. Изв. высш. учебн. заведений, Радиофизика, т. 1, № 5—6, стр. 166—176.

Шеннон К. (Shannon C. E.). A symbolic analysis of relay and switching circuits, Trans. AIEE, v. 57, 1938, p. 713—723

The synthesis of two-terminal switching circuits. Bell. System. Techn. J., v. 28, 1949, p. 59—98.

Шестаков В. И., Алгебра двухполюсных схем, построенных исключительно из двухполюсников. Алгебра *А*-схем. ЖТФ, т. 11, № 6, 1941, стр. 532.

Алгебраический метод синтеза автономных систем двухпозиционных реле. Автоматика и телемеханика, т. 15, № 4, 1954, стр. 310—324.

Алгебраический метод синтеза многотактных релейных систем. ДАН СССР, т. 99, № 6, 1954, стр. 987—990.

Шуберт (Schubert E. I.), Matrix analysis of logical networks. Commun. and Electronics, № 35, 1958, p. 10—13.

Matrix synthesis of high-speed logic. Commun. and Electronics, № 41, 1959, p. 4—8.
Matrix algebra of sequential logic., Commun. and Electronics, № 46, 1960, p. 1074—1079.
Simultaneous logical equations Matrix logic, III. Commun. and Electronics, № 46, 1960, p. 1080—1083.
Эйнгорин М. Я., О системах уравнений алгебры логики и синтеза дискретных управляющих схем с обратными связями. Изв. высш. учебн заведений, Радиофизика, т. I, № 2, 1958, стр. 169—184.
Яблонский С. В., О полных системах функций алгебры логики, УМН, т. 7 : 5 (51), 1952, стр. 197.
Функциональные построения в k-значной логике. Труды матем. ин-та им. В. А. Стеклова, т. 51, 1958, стр. 5—142.

ИМЕННОЙ УКАЗАТЕЛЬ

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ