В. Ф. Романов

Лекции по теории автоматов

Часть 1

Теория абстрактных автоматов

Владимир 2009

Учебное пособие для студентов очной и заочной форм обучения специальностям в области вычислительной техники, информатики и управления.

ОГЛАВЛЕНИЕ

ЧАСТЬ 1. ТЕОРИЯ АБСТРАКТНЫХ АВТОМАТОВ

1.1. Общие сведения

Абстрактный автомат (АА) – дискретный преобразователь информации; представляет собой множество, состоящее из шести элементов:

$$S = \{X, Q, Y, \delta, \lambda, q_1\}$$

S – абстрактный автомат;

X – множество входных символов (входной алфавит автомата):

$$X = \{x_1, \ldots, x_m\};$$

Q – множество состояний автомата:

$$Q = \{q_1, \ldots, q_n\};$$

Y – множество выходных символов (выходной алфавит автомата):

$$Y = \{y_1, \ldots, y_p\};$$

δ – функция переходов автомата из одного состояния в другое:

$$q_j = \delta(q_i, x_k),$$

где: q_j – следующее (новое) состояние автомата; q_i – текущее состояние автомата;

x_k – текущий входной символ;

λ – функция выходов:

$$y_l = \lambda\,(q_i, x_k);$$

q_1 – начальное состояние автомата (применяется также обозначение q_0).

Автомат называется *конечным*, если множества X, Q, Y – конечны.

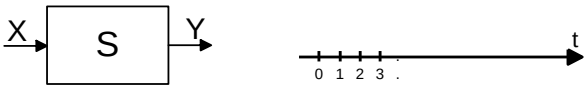


Рис.1. Представление абстрактного автомата

На рис. 1 t – дискретное время: $t = nT$, где T – интервал (такт), разделяющий дискретные моменты времени; если $T = 1$, то $t = n$, т. е. дискретное время сопоставляется упорядоченному ряду натуральных чисел.

Абстрактный автомат (АА) можно рассматривать как "черный ящик" с одним входом и одним выходом, с которым можно экспериментировать, не зная, что находится внутри.

Выходной символ ($y_l \in Y$) зависит не только от входного символа ($x \in X$), но и от того, в каком состоянии ($q_i \in Q$) находится автомат. Автомат функционирует в дискретном времени; это означает, что элементы описания автомата заданы только в упомянутые выше дискретные моменты.

Представим, что с некоторого начального, например, нулевого момента времени на вход автомата подаются входные символы, образующие *входное слово* некоторой длины (длина любого i-го слова измеряется числом символов). На выходе получаем *выходное слово* той же длины (рис. 2).

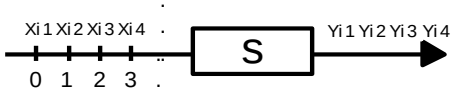


Рис.2. Преобразование входных слов в выходные

Сказанное означает, что автомат может рассматриваться как преобразователь входных слов в выходные с сохранением длины слов. Символы алфавитов, присутствующие на входе и выходе автомата, будем также называть входными и выходными сигналами.

На практике широкое распространение получили две основные модели, описывающие функционирование АА:

1. Модель Мили;
2. Модель Мура.

Модель Мили.

Законы функционирования автомата Мили представлены следующим образом:

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

t – текущий момент времени;
t+1 – следующий момент времени;
q(t+1) – состояние автомата в следующий момент времени;
q(t), x(t), y(t) – элементы описания автомата в текущий момент времени.

Модель Мура.

Законы функционирования автомата Мура представлены следующим образом:

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

В модели Мура выходной сигнал явно зависит только от состояния, а косвенно – и от входного сигнала.

Любой автомат можно спроектировать по той или иной модели.

1.2. Способы задания автоматов

Рассмотри два основных способов задания автоматов:

1. Табличный способ

Автомат Мили

Для автомата Мили табличный способ заключается в построении двух таблиц: таблицы переходов (ТП) и таблицы выходов (ТВ).

x\q	…	$\mathbf{q_i}$	…
$\mathbf{x_k}$		$\delta(q_i,x_k)$	

а

x\q	…	$\mathbf{q_i}$	…
$\mathbf{x_k}$		$\lambda(q_i,x_k)$	

б

Рис. 3. Табличный способ: **а** – таблица переходов, **б** – таблица выходов.

Пример:
а) Таблица переходов

x\q	$\mathbf{q_1}$	$\mathbf{q_2}$	$\mathbf{q_3}$
$\mathbf{x_1}$	q_3	q_1	q_1
$\mathbf{x_2}$	q_2	q_3	q_2

б) Таблица выходов

x\q	$\mathbf{q_1}$	$\mathbf{q_2}$	$\mathbf{q_3}$
$\mathbf{x_1}$	y_1	y_1	y_2
$\mathbf{x_2}$	y_1	y_2	y_1

Автомат Мура

Таблица переходов и таблица выходов объединяются, и добавляется строка выходных сигналов, соответствующих состояниям автомата. На рисунке 4 показана таблица переходов и выходов для автомата Мура.

	…	$\lambda(q_i,x_k)$	…
x\q	…	$\mathbf{q_i}$	…
$\mathbf{x_k}$	$\delta(q_i,x_k)$		

Рис. 4. Таблица переходов и выходов

Пример. *Таблица переходов и выходов:*

	y_1	y_1	y_3	y_2	y_3
x\q	$\mathbf{q_1}$	$\mathbf{q_2}$	$\mathbf{q_3}$	$\mathbf{q_4}$	$\mathbf{q_5}$
$\mathbf{x_1}$	q_2	q_5	q_5	q_3	q_3
$\mathbf{x_2}$	q_4	q_2	q_2	q_1	q_1

2. Графовый способ

Автомат представляется ориентированным связным графом (орграфом), вершины которого соответствуют состояниям автомата, а дуги – переходам из состояния в состояние. Обозначения входных и выходных сигналов распределяются в автоматах Мили и Мура по-разному.

Построим графы для автоматов Мили и Мура по вышеприведенным таблицам:

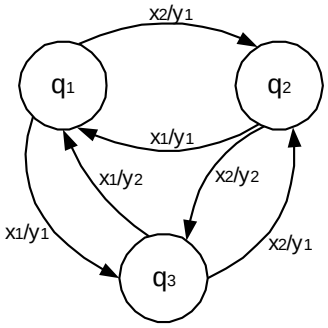


Рис. 5. Представление автомата Мили в виде графа

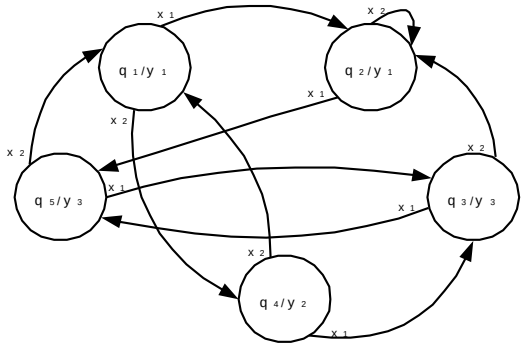


Рис. 6. Представление автомата Мура в виде графа

Замечание: В автомате Мили выходной сигнал вырабатывается при переходе, а в автомате Мура присутствует в течение всего периода дискретного времени, пока автомат находится в данном состоянии.

Детерминированный автомат – автомат, в котором имеется полная определенность переходов из всех состояний в зависимости от входных сигналов (под действием одного и того же сигнала автомат не может переходить из любого рассматриваемого состояния в различные состояния). Фрагмент графа, изображенный на рисунке 7, не может относиться к детерминированному автомату.

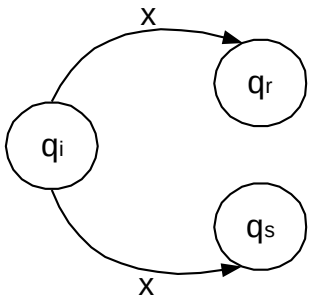


Рис.7. Фрагмент графа, иллюстрирующий неопределенность переходов

Замечание: Недетерминированные (например, вероятностные) автоматы существуют, но их рассмотрение не предусмотрено в данном курсе.

Состояние автомата q_i называется *устойчивым*, если для любого входного сигнала $x_к$, такого, что $\delta(q_s , x_k) = q_i$, справедливо соотношение: $\delta(q_i , x_k) = q_i$. (здесь q_s – состояние, предшествующее q_i). Это означает, что, автомат не изменяет своего состояния при повторении на следующем такте сигнала, приведшего его в состояние q_i . Фрагмент графа, иллюстрирующий устойчивость состояния, приведен на рисунке 8.

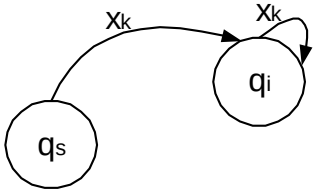


Рис. 8. Устойчивое состояние автомата

Автомат называется *асинхронным*, если каждое его состояние $q_i \in Q\ (i = 1, \ldots , n)$ устойчиво; в противном случае автомат называется *синхронным*. Синхронные автоматы важны для теории, а на практике чаще используются асинхронные; устойчивость состояний в асинхронных автоматах достигается введением специальных сигналов синхронизации.

1.3. Примеры абстрактных автоматов, выполняющих полезные действия

1**.** Автомат для задержки сигнала на один такт (для запоминания на один такт)

Опишем данный автомат таблицами и графом:

Таблица переходов и таблица выходов:

x\q	**q_0**	**q_1**
x_0	q_0	q_0
x_1	q_1	q_1

x\q	**q_0**	**q_1**
x_0	y_0	y_1
x_1	y_0	y_1

Поскольку автомат является двоичным, введем обозначения: $x_0 = y_0 = 0$; $x_1 = y_1 = 1$.

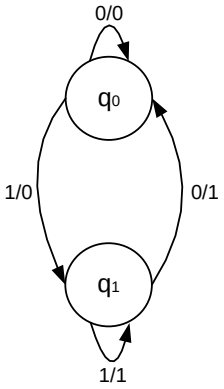


Рис. 9. Граф автомата для задержки сигнала на один такт

Простой анализ показывает, что выходной сигнал в текущем такте повторяет входной, который был на такт раньше.

2**.** Автомат для проверки двоичной последовательности на четность.

Опишем данный автомат таблицами и графом:

Таблица переходов и таблица выходов:

x\q	**q_0**	**q_1**
x_0	q_0	q_1
x_1	q_1	q_0

x\q	**q_0**	**q_1**
0	0	1
1	1	0

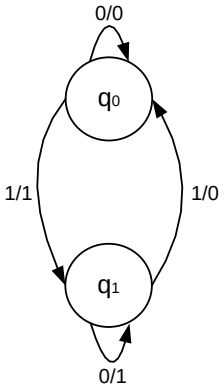


Рис. 10. Граф автомата для проверки двоичной последовательности на четность

Анализ показывает, что «0» на выходе автоматически вырабатывается при четном числе единиц на входе, а «1» на выходе вырабатывается при нечетном числе единиц на входе.

Оба рассмотренных автомата имеют "слабую" память, но слабую в разном смысле. У первого автомата "короткая" память во времени (помнит только один сигнал). У второго автомата память "длинная" (длина входной последовательности может быть любой), но он различает (*распознает*) лишь два класса последовательностей.

3**.** Автомат для задержки сигнала на два такта.

Автомат имеет четыре состояния, закодированных двумя двоичными разрядами: $q_0 = 00$; $q_1 = 01$; $q_2 = 10$; $q_3 = 11$.

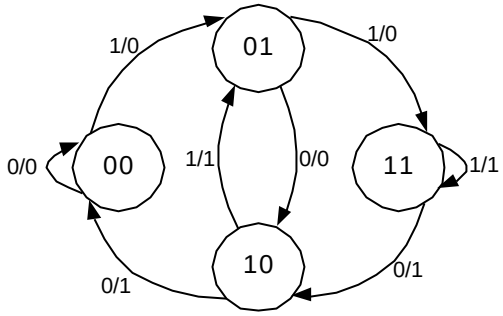


Рис. 11. Граф автомата для задержки сигнала на два такта

Достоинства примененного кодирования:

- первая цифра кода состояния показывает, какой сигнал выдает автомат (он легко преобразуется в автомат Мура); вторая цифра кода показывает, под действием какого сигнала автомат приходит в данное состояние.
- легко проверить, что выходной сигнал повторяет входной через два такта.

4**.** Двоичный последовательный сумматор, реализованный для одного разряда входного кода.

Автомат имеет два состояния: q_0 – нет переноса (сложение цифр операндов не требует учета единицы переноса из предыдущего разряда кода);

q_1 – есть перенос (единица переноса должна быть учтена).

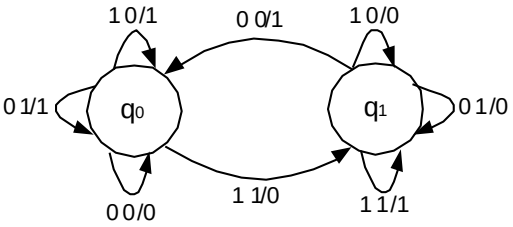


Рис. 12. Граф одноразрядного двоичного последовательного сумматора

В числителе "дроби", записанной при каждой из дуг графа, указаны цифры слагаемых; в знаменателе – результат суммирования в текущем разряде. Сумматор позволяет суммировать двоичные последовательности произвольной длины.

1.4. Поведение изолированного синхронного автомата

Изолированный автомат – автомат, на входе которого присутствует сигнал, имеющий постоянное значение, что эквивалентно "отключению" входа. Если изолированный автомат является синхронным, переходы из одного состояния в другое возможны, но при этом исключены разветвления путей, отображающих последовательности переходов (автомат является детерминированным). Следовательно, автомат с конечным числом состояний (*конечный автомат*) неизбежно должен попасть в состояние, в котором уже находился ранее, и на диаграммах переходов обязательно будет присутствовать поглощающее состояние или цикл.

Примеры диаграмм, иллюстрирующих поведение рассматриваемого автомата при разных начальных состояниях, приведены рисунке 13.

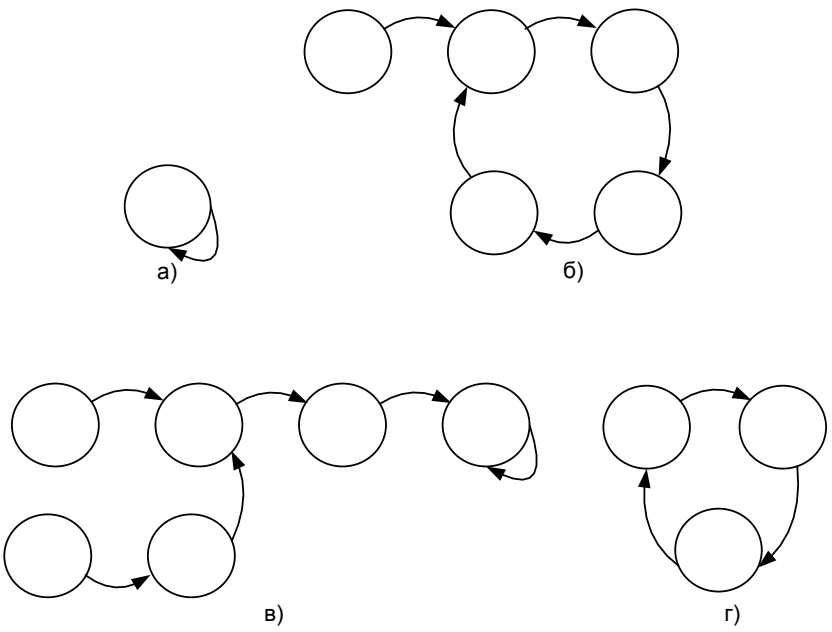


Рис.13. Поведение изолированного синхронного автомата

Отметим, что длина цикла, измеренная числом дуг на диаграмме, не превышает числа состояний (n), то же самое можно сказать и о "времени" вхождения в цикл, измеренном в условных дискретных единицах.

Проблема умножения.

Теорема. *Никакой конечный автомат не может перемножать пары чисел с произвольно большим числом разрядов.*

Причина заключается в том, что с возрастанием числа разрядов сомножителей при умножении необходимо накапливать неограниченно большие (по объему занимаемой памяти) промежуточные результаты.

Для математического доказательства используем метод "от противного": предположим, что существует автомат S, перемножающий пары двоичных чисел с произвольно большим числом разрядов (система счисления может быть любой без ограничения общности). Используем для опровержения последнего утверждения частный случай: оба сомножителя равны 2^n . В этом случае каждый из сомножителей содержит единицу, за которой следуют n нулей; результат умножения ($2^n \times 2^n = 2^{2n}$) содержит единицу и $2n$ нулей. Применим экономный способ использования памяти: пары разрядов сомножителей подаются последовательно, начиная с младших разрядов (аналогично тому, как это делалось в рассмотренном выше сумматоре).

Чтобы автомат правильно работал, он должен после прекращения подачи сомножителей добавить к уже выработанным $n + 1$ нулям еще $n - 1$ нулей, а затем в заключение единицу. Но после прекращения подачи сомножителей автомат будет работать как изолированный.

Если изолированный автомат S имеет k состояний и способен выработать на выходе подряд n нулей, где $n > k$, то это означает, что он находится в поглощающем состоянии или вошел в цикл. Следовательно, он уже не сможет выработать единицу. Так как всегда возможно сделать значение n таким, что $n - 1 > k$, правильный результат при выполнении указанного неравенства не будет получен и, следовательно, предположение о существовании автомата S приводит к противоречию. Теорема доказана.

1.5. Регулярные выражения и конечные автоматы

Рассмотрим структуру рекурсивных определений, которые применяются как средство строгого математического описания классов объектов. Рекурсивное определение некоторого класса K включают следующие части:

База – определяет один или несколько простейших объектов класса K.

Рекурсия – состоит из одного или нескольких пунктов. В каждом пункте говорится, что если определенные виды объектов принадлежат классу K, то и другие объекты, построенные из первых по некоторому правилу, также принадлежат классу K. Число пунктов рекурсии соответствует числу правил.

Ограничение – устанавливает, что никакие объекты, кроме тех, которые построены посредством применения базы и рекурсии, не принадлежат классу K.

Пример: рекурсивное определение правильного скобочного выражения (ПСВ).

БАЗА: ( ) – ПСВ;

РЕКУРСИЯ:

а) если A – ПСВ и B – ПСВ, то AB – ПСВ;

б) если A – ПСВ, то (A) – ПСВ;

ОГРАНИЧЕНИЕ: никаких других ПСВ нет.

Отметим, что в определении использованы метасимволы A и B, обозначающие любые ПСВ; выражение AB обозначает конкатенацию (последовательную запись) A и B.

Существует тесная связь между возможностями конечных автоматов и теми входными последовательностями, которые автомат распознает. Будем говорить, что автомат S *распознает* последовательность входных сигналов, если эта последовательность приводит к тому, что автомат переходит в заранее обусловленное состояние или выдает обусловленный выходной сигнал. Можно построить автомат, распознающий, например, слова некоторого языка, то есть отличающий "правильное" входное слово от недопустимого в данном языке входного слова.

Определим рекурсивно класс R регулярных выражений на заданном алфавите Σ.

БАЗА: Любой однобуквенный символ $x \in \Sigma$ – регулярное выражение ($x \in R$).

РЕКУРСИЯ:

а) если E – регулярное выражение и F – регулярное выражение, то $E \vee F$ (выбор) – регулярное выражение $((E \vee F) \in R)$;

б) если E – регулярное выражение и F – регулярное выражение, то EF – регулярное выражение $(EF \in R)$;

в) если E – регулярное выражение, то и E^* – регулярное выражение $(E^* \in R)$.

ОГРАНИЧЕНИЕ: никаких других регулярных выражений нет.

В приведенном определении E и F – метасимволы, обозначающие любые регулярные выражения; E^* – любое число повторений E, включая и отсутствие E. Приоритет введенных операций возрастает в порядке их перечисления.

Теорема Клини:

1. *Конечный автомат может распознавать лишь регулярные множества последовательностей.*

2. *Любое регулярное множество последовательностей может быть распознано*[1] *некоторым конечным автоматом.*

Доказательство теоремы приведено в [1].

---

[1] Иначе говоря, представимо.

Принципиальная важность теоремы Клини заключается в том, что она устанавливает ограничения на возможности конечных автоматов; в теории алгоритмов доказано, что любое вычисление может быть сведено к распознаванию.

1.6. Алгоритмы и машины Тьюринга

В данном разделе рассматриваются более мощные (по сравнению с конечными автоматами) модели устройств, выполняющих вычисления, – машины Тьюринга (МТ). Анализ показал, что с помощью машин Тьюринга можно реализовать любой алгоритм. Одновременно МТ служит для уточнения самого понятия алгоритма и его формализации, поскольку, как установлено в теории вычислений, широко применяемые словесные определения алгоритма не являются точными и исчерпывающими.

Рассмотрим, например, следующее определение [2]: *Алгоритм – это определенное на некотором языке конечное предписание (способ, рецепт), задающее дискретную последовательность исполнимых элементарных операций для решения проблемы. Процесс выполнения предписания состоит из отдельных шагов, на каждом из которых выполняется одна очередная операция.*

Как отмечено в [2], это определение, понятное в интуитивном смысле, не является формальным. Употребленные в нем термины "предписание", "элементарная операция", а также объекты, к которым применяется алгоритм, требуют уточнения, если мы хотим говорить об алгоритмах строго. Алгоритмы в интуитивном смысле не являются математическими объектами, к ним не применимы формальные исследования и доказательства. Так, сравнение двух алгоритмов по эффективности, проверка их эквивалентности и т. д., возможны только на основе их формального представления.

Машина Тьюринга представляет собой *бесконечный* автомат, благодаря бесконечной (потенциально) ленте, разбитой на ячейки. В ячейках записываются символы некоторого алфавита МТ. Имеется также конечный автомат с головкой записи и считывания (ГЗЧ). ГЗЧ *обозревает* одну ячейку ленты в текущий момент дискретного времени.

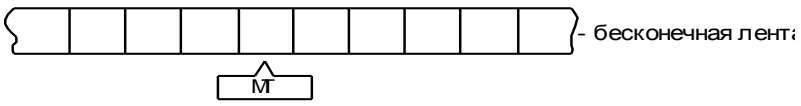


Рис.14. Машина Тьюринга

Функции ГЗЧ: считывание символа из обозреваемой ячейки; запись символа в обозреваемую ячейку; передвижение влево или вправо на одну ячейку.

В каждый момент времени МТ описывается следующей пятеркой:

$$(q_i, s_j, \delta(q_i, s_j), \lambda(q_i, s_j), d(q_i, s_j)),$$

где

q_i – состояние МТ в текущий момент времени;

s_j – обозреваемый символ в текущий момент времени;

δ – функция переходов, которая определяет следующее состояние;

λ – функция выходов, определяющая запись нового символа в обозреваемую ячейку;

d – функция, определяющая передвижение головки влево (L) или вправо (R) на один шаг.

Более краткое обозначение элементов пятерки: (q_i , s_j , q_{ij} , s_{ij} , d_{ij}).

Тезис Тьюринга: *Любой процесс, который было бы естественно назвать эффективной процедурой*[2]*, может быть реализован МТ.*

[2] *Алгоритмом* в современной терминологии

Следует подчеркнуть, что *тезис* – это, в общем случае, правдоподобное утверждение, которое не обязательно математически строго доказано. Однако многие научные тезисы действенны, так как прошли проверку временем и практикой.

Примеры машин Тьюринга для конкретных вычислений.

1. Счетчик четности единиц.

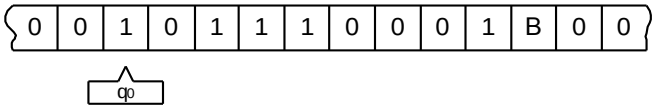


Рис.15. МТ для счетчика четности

На рис. 15 показана МТ в начальном состоянии q_0 , при этом обозревается первый символ двоичной последовательности, которая заканчивается ограничителем *B*. Далее на каждом такте ГЗЧ продвигается вправо, заменяя все символы символом "0", причем смена состояний (q_0 на q_1 и обратно) происходит всякий раз, когда ГЗЧ обнаружит единицу. Таким образом, состояние q_0 связано с четным числом обнаруженных единиц, а состояние q_1 – с нечетным. Останов МТ (переход в заключительное состояние H) происходит по достижении символа *B*, при этом на его место записывается "0", если число единиц в исходной последовательности было четным, и "1", если это число было нечетным. После завершения процесса вычислений ГЗЧ будет указывать на эту ячейку с заключительной информацией, а во всех остальных ячейках ленты будут записаны нули.

Представим работу МТ двумя способами: таблицей и графом.

Строки таблицы содержат все "пятерки", описывающие функционирование МТ.

Таблица

q_i	s_j	q_{ij}	s_{ij}	d_{ij}
q_0	0	q_0	0	R
q_0	1	q_1	0	R
q_0	B	H	0	–
q_1	0	q_1	0	R
q_1	1	q_0	0	R
q_1	B	H	1	–

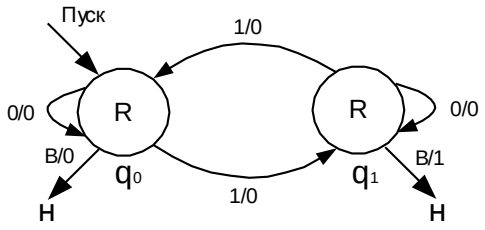


Рис. 16. Граф счетчика четности единиц

В вершинах графа явно указан символ, обозначающий движение ГЗЧ вправо в каждом из двух состояний. Заключительное состояние обозначено на рисунке буквой *H* – "halt" (останов).

2. Машина Тьюринга для проверки правильности скобочных выражений.

Рекурсивное определение правильного скобочного выражения (ПСВ) было дано в разделе 1.3. Заметим, что конечный автомат не может решить поставленную задачу для скобочного выражения произвольной длины. МТ решает задачу в общем случае, так как наличие неограниченной ленты эквивалентно неограниченному объему памяти.

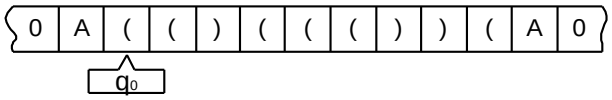


Рис. 17. МТ для проверки скобочных выражений

Скобочное выражение заключено между левым и правым ограничителями, обозначенными символом *A*. В начальном состоянии автомата ГЗЧ обозревает первый символ скобочного выражения (рис. 17). Реализация вычислений представлена графом (рис. 18).

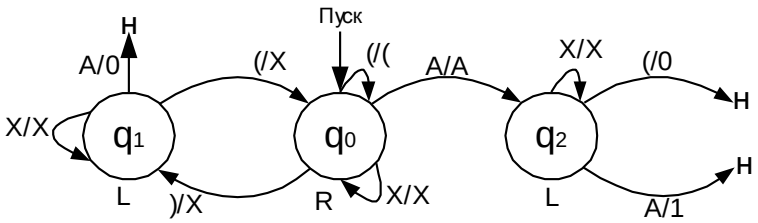


Рис. 18. Граф МТ для проверки скобочных выражений

Работа машины: сначала ГЗЧ движется вправо до первой правой скобки, заменяет ее символом *X*, переходит в состояние q_1 и движется влево до ближайшей левой скобки, заменяет ее символом *X*, переходит в состояние q_0, и челночное движение повторяется.

Если машина, находясь в состоянии q_1, достигает левого символа *A*, то печатает "0" и останавливается – скобочное выражение неправильно.

Если машина, находясь в состоянии q_0, достигает правого символа *A*, не обнаружив больше правой скобки, то переходит в заключительное состояние q_2, связанное с движением влево, и в последний раз просматривает последовательность: не осталась ли непарная левая скобка. Если по пути встретится левая скобка, то машина печатает "0" и останавливается. При достижении в состоянии q_2 левого символа *A*, машина печатает "1" и останавливается – скобочное выражение правильно.

3. Машина Тьюринга для умножения двух чисел в унарном коде.

Действие машины представлено рисунками 19 – 21.

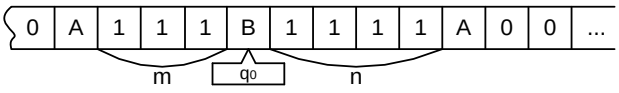


Рис. 19. МТ для умножения – исходное состояние

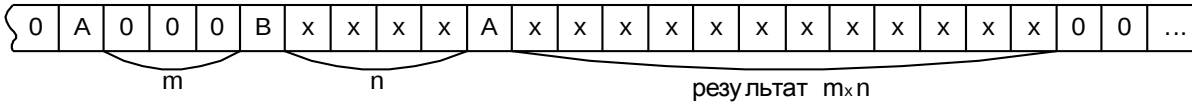


Рис. 20. МТ для умножения – результат операции

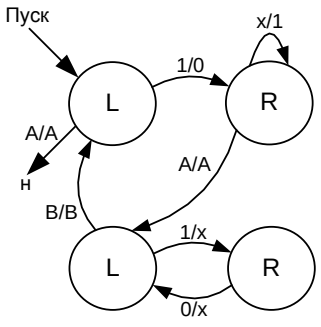


Рис. 21. Граф МТ – множительного устройства

На рис. 21 граф представлен в сокращенном виде – не показаны петли с одинаковыми числителем и знаменателем.

МТ представляет собой простой базис для описания *эффективных процедур*, поскольку все шаги, регламентирующие поведение автомата, определяются четкими правилами. В современной терминологии, МТ – простой базис для описания алгоритмов и уточнения самого понятия алгоритма.

В приведенных выше примерах для каждого вычисления использовался свой специальный конечный автомат – так называемая *конкретная* машина Тьюринга. Конечный автомат в конкретных МТ играет роль алгоритма вычислений.

Можно считать, что любая МТ вычисляет некоторую функцию (заключительное содержимое ленты) от заданного аргумента (исходное содержимое ленты). В связи с этим возникло понятие: *функция, вычислимая по Тьюрингу*.

Тьюринг показал, как построить *универсальную* машину Тьюринга (УМТ), которая интерпретирует поведение любой конкретной МТ и, следовательно, может вычислить любую функцию, которую вычисляет конкретная МТ.

УМТ имеет структуру, показанную на рис. 22.

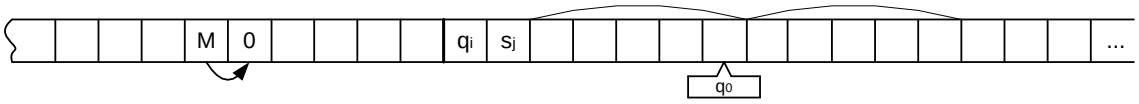


Рис. 22. Структура УМТ

Левая часть ленты (до символа q_i) имитирует ленту конкретной МТ, правая – содержит описание автомата конкретной МТ в виде пятерок.

Символ *M* показывает расположение ГЗЧ конкретной МТ; будем считать, что обозреваемый символ находится справа от символа *M*.

Пара ячеек q_i, s_j – зона режима, указывающая текущее состояние и текущий обозреваемый символ (на рисунке это символ "0").

Работа УМТ: УМТ начинает работу с запоминания символов q_i, s_j зоны режима, затем ГЗЧ движется вправо до тех пор, пока не найдет пятерку, в которой первые два символа совпадают с символами зоны режима. Найдя такую пятерку, УМТ запоминает q_{ij}, s_{ij}, d_{ij} , и ГЗЧ движется влево. Далее выполняются следующие действия:

a) $q_i := q_{ij}$, т. е. в зону режима записывается символ, обозначающий новое состояние автомата конкретной МТ;
b) справа от M записывается s_{ij} ;
c) выполняется сдвиг M согласно значению d_{ij} (L или R), что соответствует передвижению ГЗЧ имитируемой машины;
d) запоминается новый обозреваемый символ (справа от M) и переносится в зону режима в качестве s_j .

Таким образом, зона режима оказывается обновленной, и указанные действия повторяются до останова машины.

Вариант формулировки тезиса Тьюринга**:** *Вычислимы те, и только те, объекты, которые могут быть вычислены УМТ.*

Согласно Тьюрингу, алгоритмом можно считать только ту процедуру, закодированную для МТ, которая приводит к останову машины.

В связи с этим возникает вопрос: существует ли универсальный распознаватель алгоритмов, т. е. МТ, которая для любой другой МТ определит, остановится последняя или нет. Этот вопрос назван в теории машин Тьюринга *проблемой останова*. Доказано, что проблема останова неразрешима и, следовательно, универсального распознавателя алгоритмов не существует [1]. Существуют и другие *алгоритмически неразрешимые* проблемы, к доказательству неразрешимости которых привлекается механизм вычислений, введенный Тьюрингом.

Простые системы (базисы) для решения проблем вычислимости создали независимо друг от друга и другие выдающиеся исследователи в области теории алгоритмов: А. Черч (математический аппарат рекурсивных функций), А. А. Марков (*нормальные алгорифмы*, сводящиеся к преобразованию слов в некотором алфавите), Э. Пост (механизм преобразования двоичных последовательностей, подобный МТ). Все названные системы равноценны с точки зрения их принципиальных возможностей и эквивалентны как формализмы для определения алгоритма [3].

1.7. Элементы теории формальных грамматик и языков

Основные определения

Алфавит – множество символов, например, $\Sigma = \{a, b, c,\ldots, z\}$.

Цепочка (слово) – последовательность символов из алфавита. Для обозначения цепочек используются строчные буквы греческого алфавита: α, β, γ …, а сами цепочки заключаются в ординарные кавычки. Например, $\Sigma = \{a, b, c, =, –, +\}$ – алфавит, α = ‘a + b = c’– цепочка над алфавитом; п*устая цепочка* ε = ' ' – цепочка, не содержащая символов.

Рекурсивное определение цепочки над алфавитом Σ**:**

БАЗА : ε – цепочка;

РЕКУРСИЯ: если $x \in \Sigma$ и А – цепочка, то Ах – цепочка (здесь А – метасимвол, обозначающий любую цепочку);

ОГРАНИЧЕНИЕ: других цепочек нет.

Длина цепочки – число символов в цепочке, заключается в прямые скобки, например, $|\alpha| = 5$, $|\varepsilon| = 0$.

Подцепочка: цепочка β является подцепочкой цепочки α, если существуют такие цепочки γ и δ , быть может, пустые, что α = γ β δ .

Цепочка ‘победа’ содержит подцепочки: 'обед', 'беда', 'еда', 'да' (здесь цепочки заданы над алфавитом русского языка).

Конкатенация – сцепление цепочек; обозначается αβ или α·β. Конкатенация *ассоциативна*, но не *коммутативна.*

Обозначим через Σ* – множество всех цепочек над алфавитом Σ.

Система {Σ*, ·**}** образуют известную алгебру – *полугруппу*, в которой Σ* - носитель алгебры, · – ассоциативная бинарная операция.

Обращение цепочки. Цепочка β называется *обратной* цепочкой цепочки α, если она представляет собой последовательность символов цепочки α, выписанных в обратном порядке.

Формальный язык

Формальным языком L на алфавите Σ называется любое подмножество множества Σ^* ($L \subseteq \Sigma^*$).

Примеры**:**

1) некоторые языки программирования, например, АЛГОЛ (с рядом оговорок);

2) $\Sigma = \{0, 1\}$, $L = \{\alpha \mid, |\alpha| \leq 100\}$ – множество цепочек в двоичном алфавите, длина которых не превышает 100 символов;

3) $L = \{\varepsilon\}$ – пустой язык;

4) $L = \Sigma^*$;

5) $\Sigma = \{a, b, c\}$, $\alpha = \{a^n b^n c^n \mid n \geq 1\}$, здесь условный показатель *n* при символе – число повторений символа.

Операции над языками. Поскольку языки – множества, к ним применимы известные теоретико-множественные операции, в частности, объединение языков ($L_1 \cup L_2$) и пересечение языков ($L_1 \cap L_2$).

Специфическая операция – *конкатенация языков:*

Пусть L_1 – язык на Σ_1 и L_2 – язык на Σ_2, тогда конкатенация языков имеет вид:

$$L_1 \cdot L_2 = \{\alpha\,\beta \mid \alpha \in L_1, \beta \in L_2\}$$

Способы задания языков**:**

1. Способ перечисления элементов; в искусственных языках не всегда удобен из-за возможного большого числа элементов.

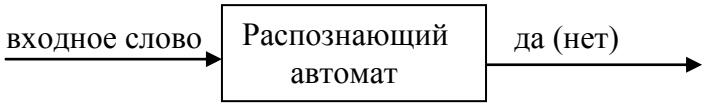
2. Способ задания с помощью характеристического свойства:

$L = \{\alpha \mid P(\alpha)\}$, где $P(\alpha)$ – предикат (характеристическое свойство множества).

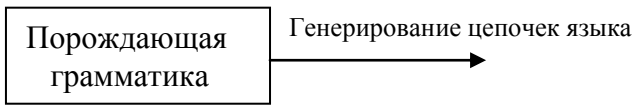
Способ задания языка не должен ограничивать длину элемента, входящего в язык. В теории формальных грамматик известны методы определения языков, свободные от указанного ограничения. Выделим два наиболее важных метода.

1. Задание языков помощью допускающих (распознающих) автоматов.
2. Задание языков с помощью порождающих грамматик.

Распознающий автомат выделяет из цепочек (слов), подаваемых на его вход, слова, принадлежащие определяемому языку.



Порождающая грамматика генерирует "правильные" цепочки, принадлежащие языку L.



Далее предметом нашего рассмотрения будут порождающие грамматики.

Компонентами порождающих грамматик являются:

1. Два алфавита:
   - вспомогательный (нетерминальный);
   - основной (терминальный).

   Эти два алфавита не содержат общих символов, их пересечение пусто.
2. Множество *правил вывода* (*продукций*), состоящее из упорядоченных пар цепочек $\langle\alpha, \beta\rangle$. Каждая цепочка в этих парах составлена из объединения элементов терминального и нетерминального алфавитов. Сама продукция обозначается α –> β и подразумевает замену при подходящих условиях цепочки α цепочкой β.

3. Начальный символ (аксиома) грамматики – один из нетерминалов, который наделен особым статусом – с него начинается порождения языка.

Примем соглашение, касающееся обозначений в формальных грамматиках:

- малыми буквами начала латинского алфавита (a, b, c, …) обозначаются терминалы;
- большими буквами начала латинского алфавита (A, B, C, …) обозначаются нетерминалы;
- большими буквами конца латинского алфавита (U, V, W, X, Y, Z) обозначаются как терминалы, так и нетерминалы;
- греческими буквами (α, β, γ, …) обозначаются цепочки, которые состоят как из терминалов, так и нетерминалов;
- малыми буквами конца латинского алфавита (u, v, w, …) обозначаются цепочки, состоящие только из терминалов: именно они являются целевыми.

Порождающие грамматики общего типа (ПГОТ)

Обозначение грамматики: $G = \langle N, \Sigma, P, S \rangle$,
где:
N – нетерминальный алфавит;
Σ – терминальный алфавит;
P – множество продукций; т. е. правил вывода вида α – –> β, где $\alpha, \beta \in (N \cup \Sigma)^*$;
S – аксиома грамматики.

Пример 1. $G_1 = \langle \{S\}, \{a, b\}, P, S \rangle$,
где:
P: 1. aSa – –> aSb;
2. S – –> a;
3. S – –> aSa.

Непосредственная выводимость: цепочка β непосредственно выводима из цепочки α в грамматике G, если найдутся такие цепочки γ_1, γ_2, ω_1, ω_2, что $\alpha = \gamma_1\omega_1\gamma_2$, $\beta = \gamma_1\omega_2\gamma_2$, и среди правил вывода имеется правило ω_1 – –> ω_2 .

В цепочке α подцепочки γ_1, γ_2 – *контекст*. Непосредственно выводимая цепочка получается применением одной продукции грамматики; обозначение непосредственной выводимости: α = => β. Примеры непосредственной выводимости в приведенной грамматике: aSa = => aSb; aSa = => aaa.

Знак «= =>» можно интерпретировать как обозначение бинарного отношения непосредственной выводимости.

Используя понятие непосредственной выводимости, определим *выводимость*. Будем записывать: α = =>* β (β *выводима* из α), если существует последовательность цепочек $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n$, таких, что $\alpha_0 = \alpha$, $\alpha_n = \beta$ и α_i = => α_{i+1} ($i = 0, \ldots, i$-1), или β = α. Указанную последовательность цепочек назовем *выводом длины n* .

Таким образом, выводимость – рефлексивное и транзитивное бинарное отношение.

В рассмотренной грамматике G_1 возможны выводы: S = =>* a (вывод длины 1); S = =>* aaa (вывод длины 2: S = =>* aSa = =>* aaa).

Язык, порождаемый грамматикой (обозначение – L(G)) – множество терминальных цепочек, выводимых из аксиомы, т. е. $L(G) = \{z \in \Sigma^* \mid S =>^* z\}$.

Пример 2. $G_2 = \langle N, \Sigma, P, S \rangle$;
N = {A, B, C, S}; S – аксиома;
Σ = {a, b, c};
P: 1. S – –> aSBC;
2. S – –> abC

3. CB – –> BC
4. bB – –> bb
5. bC – –> bc
6. cC – –> cc

Пример вывода: S = => aSBC = => aabCBC = => aabBCC = => aabbCC = => = => aabbcC = => aabbcc (здесь правила применены в порядке нумерации).

Грамматика G_2 порождает язык $L(G) = \{a^n b^n c^n \mid n \geq 1\}$.

Порождающие грамматики общего типа реализуются машинами Тьюринга.

В теории и практике разработки систем программирования, операционных систем, средств синтаксического анализа, компиляции и перевода применяются и другие грамматики, например, *НС-грамматики* (грамматики непосредственно составляющих), *КС-грамматики* (контекстно-свободные грамматики), *А-грамматики* (автоматные грамматики).

Все известные грамматики являются частным случаем ПГОТ и получаются посредством ограничений, налагаемых на продукции.

Автоматные грамматики

Порождающая грамматика G = ⟨N, Σ, P, S⟩ называется *автоматной*, если ее правило вывода имеет следующий вид:

$\begin{cases} A - - > aB \\ A - - > a \end{cases}$ - правосторонние правила, или $\begin{cases} A - - > Ba \\ A - - > a \end{cases}$ - левосторонние правила,

где *A, B* – элементы нетерминального алфавита; *a* – элемент терминального алфавита.

Языки, порождаемые А-грамматиками, называются *регулярными языками* (также *языками Клини*).

Пример 3: G_3 = ⟨{S, A}, {a, b}, P, S>⟩, где приняты правосторонние правила:

P: 1. S – –> aS;
2. S – –> aA;
3. A – –> bA;
4. A – –> b.

Пример вывода: S = = > aS = = > aaA = = > aabA = = > aabb. В общем случае грамматика G_3 порождает язык $L(G_3) = \{a^n b^m \mid n, m > 0\}$.

А-грамматики *маломощны* (этот термин характеризует большое в среднем число шагов при выводе). Тем не менее, существует несложное доказательство того факта, что А-грамматики порождают все возможные конечные языки.

Иллюстрация связи между автоматными грамматиками и автоматами

Рассмотрим праворостороннюю А-грамматику G_4 = ⟨N, Σ, P, S⟩ :
$N = \{S_1, S_2, S_3\}$, где S_1 – аксиома; Σ = {x, y, a, b}

P: 1. S_1 – –> y$b S_1$
2. S_1 – –> x$a S_2$
3. S_2 – –> x$a S_2$
4. S_2 – –> y$a S_3$
5. S_3 – –> x$b S_1$

Правила вывода включают входные терминальные символы x и y и выходные терминальные символы a и b.

Граф автомата Мили, реализующий данную грамматику, приведен на рис. 23.

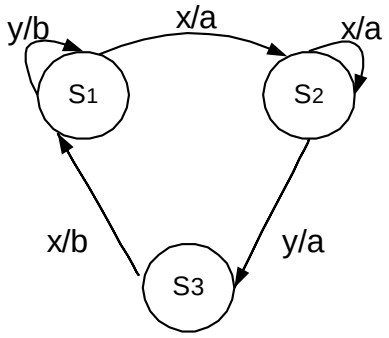


Рис. 23. Граф, реализующий грамматику G_4

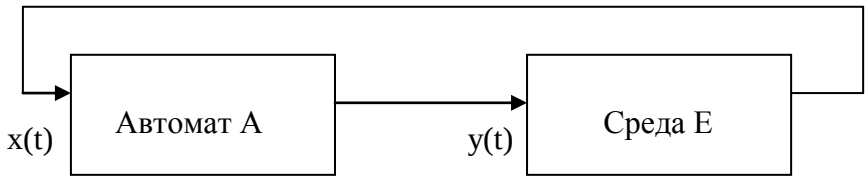
Рисунок иллюстрирует следующие соответствия:

- нетерминалы соответствуют состояниям;
- терминалы соответствуют входным и выходным символам;
- продукции определяют переходы.

Полученный автомат является преобразователем входных слов в выходные. Так, слово 'yxyx' преобразуется в слово 'baab'. Множество выходных слов образует язык, который порождается данной грамматикой.

Взаимодействие автомата со средой

В теории автоматов значительное место занимали экспериментальные методы: исследование "поведения" автоматов на моделях, отражающих взаимодействие автоматов с различными видами внешней среды, имеющей дискретное, в частности, автоматное описание.



Среду E можно рассматривать как детерминированный или стохастический автомат, поведение которого зависит от его внутреннего состояния, внешних сигналов и времени.

Введем следующие обозначения:

y(t) – действия автомата;

x(t) – реакция среды;

x(t)= +1 – благоприятная реакция (выигрыш);

x(t)= -1 – неблагоприятная реакция (проигрыш).

В стационарной среде каждое действие автомата вызывает с вероятностью p_m значение реакции x(t)= -1 и с вероятностью q_m значение x(t)= +1. При этом, $q_m = 1 - p_m$; $b_m = q_m - p_m$ – оценка математического ожидания выигрыша за действие y_m.

Принято считать, что автомат обладает *целесообразным* поведением, если на множестве экспериментальных результатов (при различных соотношениях параметров q_m и p_m) математическое ожидание выигрыша $M > M_0$, где M_0 – математическое ожидание выигрыша при $q_m = p_m = 0{,}5$.

Исторически сложились такие направления исследований с применением моделирования:

Поведение автоматов в детерминированных и случайных средах.

Игры автоматов.

Коллективное поведение автоматов.

Клеточные автоматы.

Большую роль в развитии фундаментальной теории автоматов сыграли такие исследователи, как Дж.фон Нейман, К. Шеннон, М. Л. Цетлин, В. М. Глушков, М. Минский.

Подходящим руководством для первоначального изучения теории формальных грамматик является книга М. Гросса и А. Лантена [4].

1.8. Условия автоматности отображения

Как уже отмечалось, автомат может рассматриваться как преобразователь слов в алфавитах; однако не всякое преобразование слов может быть реализовано автоматом. Другими словами, не всякое отображение входных слов в выходные является *автоматным*.

Условия автоматности отображения:

1. Отображение должно быть однозначным: каждому входному слову сопоставляется единственное выходное слово.

2. Область определения отображения должна удовлетворять условиям полноты. Это означает, что если в область определения входит некоторое слово, то и все его начальные отрезки также входят в область определения.

3. Отображение должно сохранять длину слова (каждое входное слово отображается в выходное слово той же длины).

4. Отображение должно переводить любой начальный отрезок входного слова в начальный отрезок выходного слова той же длины.

Условия автоматности накладывают жесткие ограничения на класс отображений, реализуемых автоматом. На практике большинство отображений не удовлетворяют условиям автоматности. Однако существует стандартный прием, который позволяет превратить любое алфавитное отображение в автоматное. При этом используются две операции:

- Операция выравнивания длин слов. Во входной и выходной алфавит добавляется по одной "пустой" букве (пропуск такта): ε, δ, соответственно;
  ε – приписывается один раз или многократно к концу входных слов,
  δ – приписывается один раз или многократно к началу выходных слов.
  В *последовательной* процедуре выравнивания на каждом шаге добавляется не более чем по одной букве ε и δ, с последующей операцией пополнения.
- Операция пополнения. Применяется к выровненным отображениям и заключается в распространении отображения на начальные отрезки слов.

Пример: Проверим автоматность отображения:

000 –> 001
001 –> 011

Решение:

0 –> 0
00 –> 00
00 –> 01

Из двух последних строк следует, что отображение не однозначно.

Добавим по одной пустой букве к входным и выходным словам:

000ε –> δ001
001ε –> δ011

Проверим выполнение однозначности:

0 –> δ
00 –> δ0
00 –> δ00
001 –> δ01

Неоднозначность устранена – отображение удовлетворяет условиям автоматности. В данном примере мы применили последовательную процедуру приведения отображения

к автоматному, при этом оказалось достаточным однократное приписывание пустых букв к входным и выходным словам.

Существует *стандартная* процедура выравнивания: к входному слову приписывается столько букв ε, какова длина выходного слова, и, соответственно, к выходному слову приписывается столько букв δ, какова длина входного слова. Применительно к рассмотренному примеру автоматное отображение будет в этом случае следующим:

000εεε –> δδδ001

001εεε –> δδδ011 .

1.9. Построение графа автомата по входо-выходным последовательностям

Проектирование автомата в ряде случаев может начинаться с общего или отчасти детализированного представления о его действии. В частности, первоначально алгоритм функционирования автомата может быть представлен словесным описанием, временными диаграммами или входо-выходными последовательностями (перечислением пар слов, отражающим зависимость выходных слов от входных). Рассмотрим подробнее последний случай.

Пусть дано отображение:

000 –> 000

001 –> 001

010 –> 010

011 –> 011

100 –> 110

101 –> 111.

Предварительный этап построения графа автомата заключается в проверке условий автоматности отображения. Легко видеть, что в данном случае эти условия выполнены.

Основной этап состоит из следующих шагов.

1. Построим граф в виде "ориентированного" дерева (рис.24); q_0 – обозначение начальной вершины, с которой начинается построение.

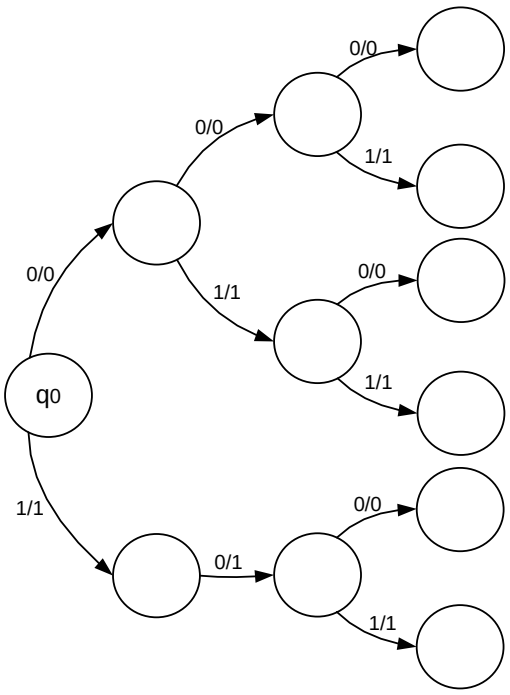


Рис. 24. Граф, представленный в виде дерева

2. Выделим и объединим одинаковые поддеревья путем склеивания вершин третьего яруса (рис. 25).

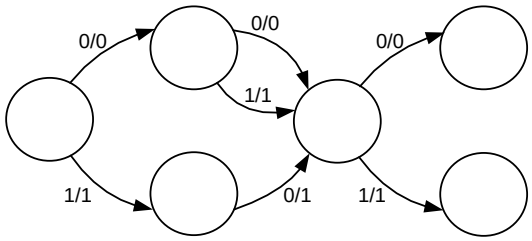


Рис. 25. Граф после склеивания вершин

4. Объединим (путем склеивания) вершины последнего яруса с начальной вершиной и введем идентификаторы состояний.

В результате получен граф автомата Мили, реализующий заданное отображение (рис. 26).

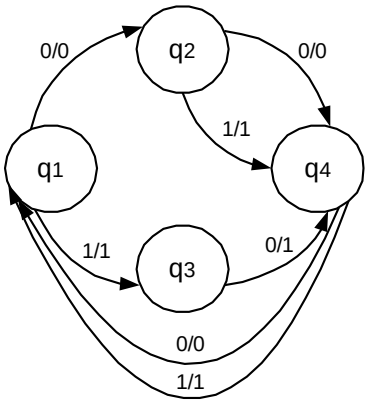


Рис. 26. Граф автомата Мили, реализующий заданное отображение

1.10. Преобразование автоматов

Два автомата называются *эквивалентными*, если они реализуют одинаковые отображения входных слов в выходные на всей области определения отображения. Автоматы Мили и Мура могут быть преобразованы друг в друга, т. е. для всякого автомата Мура может быть построен эквивалентный ему автомат Мили и, наоборот, для всякого автомата Мили – эквивалентный ему автомат Мура.

1. Преобразование автомата Мура в автомат Мили.

Пусть S_A – исходный автомат Мура; S_B – автомат Мили, который является целью преобразования. Автомат Мура характеризуется шестиэлементным множеством:

$S_A = (X_A, Q_A, Y_A, \delta_A, \lambda_A, q_{1A})$

Построим автомат Мили: $S_B = (X_B, Q_B, Y_B, \delta_B, \lambda_B, q_{1B})$.

Четыре первые компоненты автомата Мили, а также начальное состояние определяются исходя из равенств:

$X_B = X_A$,

$Q_B = Q_A$,

$Y_B = Y_A$,

$\delta_B = \delta_A$,

$q_{1B} = q_{1A}$.

Различие состоит в функциях выходов. Они определяются так: если в автомате Мура $\delta_A(q_i, x_j) = q_s$ и $y_k = \lambda_A(q_s)$, то в автомате Мили $\lambda_B(q_i, x_j) = y_k$ (рис. 27).

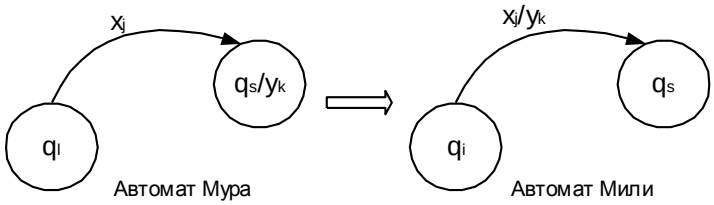


Рис. 27. Фрагмент преобразования автомата Мура в автомат Мили

В общем случае, если в вершину q_s входят несколько дуг, то выходной сигнал y_k, записанный в вершине q_s, переносится в обозначения всех дуг, входящих в данную вершину.

2. Преобразование автомата Мили в автомат Мура

При этом преобразовании в графе автомата Мили не должно быть вершин, в которые не входит ни одна дуга, но которые в то же время имеют хотя бы одну выходящую дугу.

$S_A = (X_A, Q_A, Y_A, \delta_A, \lambda_A, q_{1A})$ – исходный автомат Мили;

$S_B = (X_B, Q_B, Y_B, \delta_B, \lambda_B, q_{1B})$ – целевой автомат Мура.

Для алфавитов автомата S_B будут справедливы следующие равенства:

$X_B = X_A$,

$Y_B = Y_A$.

Для определения множества Q_B каждому состоянию $q_s \in Q_A$ поставим в соответствие множество Q_s всевозможных пар вида (q_s, y_t), где y_t – выходной сигнал, приписанный дуге, входящей в q_s.(фрагмент графа изображен на рис. 28).

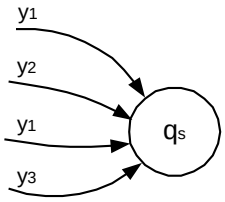


Рис. 28. Вершина q_s с входящими дугами

В этом случае Q_s представляет собой множество пар вида:

$$Q_s = \{(q_s, y_1), (q_s, y_2), (q_s, y_3)\}$$

В общем виде, если D – множество дуг, входящих в вершину q_s, то Q_s определяется следующим образом:

$$Q_s = \{(q_s, y_t) | \, y_t \in D\}$$

Итак, число элементов Q_s равно числу различных выходных сигналов при дугах автомата S_A, входящих в состояние q_s. Множество состояний автомата S_B получим как теоретико-множественное объединение множеств Q_s, ассоциированных со всеми состояниями q_s исходного автомата.

Вывод: число состояний в автомате Мура в среднем больше, чем число состояний в автомате Мили.

Функция λ_B определяется так: каждому состоянию автомата S_B, представляющему собой пару (q_s, y_t), ставится соответствие выходной сигнал y_t.

Функция δ_B определяется следующим образом: если в автомате Мили S_A есть переход $\delta_A(q_i, x_j) = q_s$ и при этом выдается выходной сигнал $\lambda_A(q_i, x_j) = y_p$, то в автомате Мура S_B будет переход из множества Q_i состояний, порожденных состоянием q_i, в состояние (q_s, y_p) под воздействием входного сигнала x_j (рис. 29).

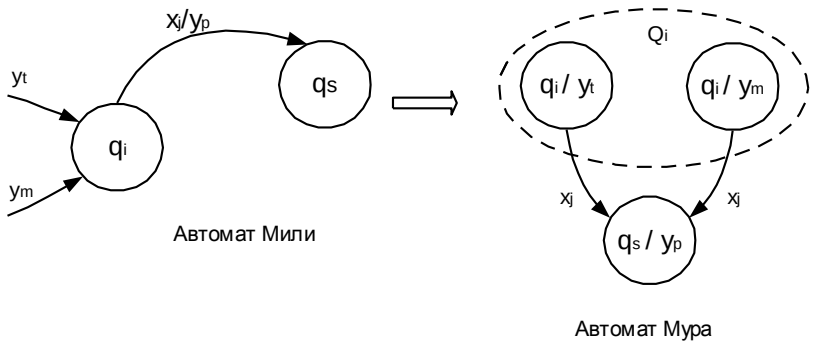


Рис. 29. Преобразование автомата Мили в автомат Мура

В качестве начального состояния q_{1B} можно взять любое состояние из множества Q_1, порожденного состоянием q_{1A}.

ЗАДАЧИ И УПРАЖНЕНИЯ

1. Дать рекурсивное определение цепочки над некоторым алфавитом, используя понятия пустой цепочки и конкатенации.

2. На вход конечного автомата подается последовательность из нулей и единиц. Изобразить граф, описывающий поведение распознающего автомата, и записать регулярное выражение для распознаваемой последовательности для случаев:

а) число единиц делится на 3;

б) все единицы появляются сериями не менее чем из трех единиц;

в) единица не появляется в момент времени, делящийся на 2 или 3.

3. Заменить регулярное выражение $(a \vee b)^*$ таким, в котором не используется знак "$\vee$".

4. Совпадают ли множества последовательностей, представляемые регулярными выражениями

а) $b(ab \vee b)^*a$ и $bb^*a(bb^*a)^*$;

б) $(a^*ab \vee ba)^*a^*$ и $(a \vee ab \vee ba)^*$?

5. Какие из следующих равенств верны:

а) $E^*F = (E \vee E^*)^*F$;

б) $E^*F^* = (E \vee F)^*(EF)^*$;

в) $E^*F^* = E^*EF^* \vee E^*FF^*$;

г) $E(FGE)^*FG = EF(GEF)^*G$?

Здесь E, F, G – метасимволы, обозначающие определенные последовательности символов алфавита.

6. Какие из следующих множеств последовательностей могут быть распознаны конечным автоматом:

а) множество всех последовательностей: 0, 1, 00, 01, 10, 11, 000, 001, 010, … ;

б) числа 1, 2, 4, 8, … , 2^n, … , записанные в двоичной системе счисления;

в) то же самое множество чисел, записанных в унарном коде: 1, 11, 1111, 11111111, 1111111111111111, … ;

г) множество последовательностей, в которых число нулей равно числу единиц;

д) последовательности: 0, 101, 11011, … , 1^k01^k (k – число повторений единицы)?

Литература

1. Минский М. Вычисления и автоматы. – М.: Мир, 1971. - 364 с.
2. Карпов Ю. Г. Теория автоматов. – СПб.: Питер, 2002. - 206 с.
3. Кузнецов О. П., Адельсон-Вельский Г. М. Дискретная математика для инженера. – М.: Энергия, 1986. - 336 с.
4. Гросс М., Лантен А. Теория формальных грамматик. – М.: Мир, 1971. - 290 с.
5. Горбатов В. А. Основы дискретной математики. – М.: Высш. школа, 1984. - 310 с.

Романов Владимир Федорович

При написании настоящего учебного пособия использовались конспекты лекций автора, подготовленные студентами Власенко В. В., Владимировой Н. В.