3.7. Конечные автоматы-распознаватели

3.7.1. Введение

Теория автоматов относится к числу ключевых разделов современной математики. Она непосредственно связана с математической логикой, теорией алгоритмов, теорией формальных грамматик. Универсальность и сравнительная простота автоматов обусловили широкое использование автоматных моделей на практике. Теория автоматов, например, успешно используется при построении узлов цифровых вычислительных машин, применяется при разработке парсеров для формальных языков (в том числе языков программирования), а также при построении компиляторов и разработке самих языков программирования.

Конечный автомат — это один из самых простых механизмов, используемых при работе с языками. Основная часть конечного автомата — это функция перехода, определяющая возможные переходы для любого текущего состояния и любого входного символа. Подразумевается, что допускается возможность перехода сразу в несколько различных состояний автомата, т. е. управляющее устройство распознавателя может быть и недетерминированным.

Под автоматом в общем смысле обычно понимают дискретный преобразователь информации, который принимает входные сигналы в дискретные моменты времени, изменяет свое состояние (с учетом прежнего состояния) и, возможно, формирует выходные сигналы.

Историческая справка

Основы теории автоматов были заложены выдающимся математиком венгерского происхождения Джоном фон Нейманом (1903–1957). Он родился в Будапеште в семье состоятельного венгерского банкира. Родители назвали его Яношем, в годы учебы и работы в Швейцарии и Германии он именовал себя Иоганном, а после переезда в США (в 1930 г.) — Джоном. В 1952 г. в работе «Вероятностная логика и синтез надежных организмов из ненадежных элементов» он показал путь создания надежных ЭВМ и других автоматов. Это даже не книга, а 5 лекций, написанных Нейманом для своего друга Ричарда Пирса. На русском языке работа была опубликована в 1956 г. в сборнике «Автоматы» [43].

Джон фон Нейман — один из создателей атомной бомбы (математически доказал осуществимость взрывного способа детонации атомной бомбы), он сформулировал основную концепцию логической организации ЭВМ (принцип совместного хранения команд и данных в памяти компьютера, называемый архитектурой фон Неймана, моделью фон Неймана или принстонской архитектурой), что послужило огромным толчком к развитию электронно-вычислительной техники.

3.7.2. Автоматы Мили и Мура

Дадим теперь формальные определения. Начнем с детерминированных автоматов.

Определение 3.128. Конечный детерминированный автомат с выходом (*автомат Мили*[1]) — это система $A = (Q, \Sigma, \Omega, \delta, \lambda, q_0)$, где:

- Q — конечное *непустое* множество состояний;
- $q_0 \in Q$ — начальное состояние;
- Σ — конечное *непустое* множество входных символов (входной алфавит);
- Ω — конечное *непустое* множество выходных символов (выходной алфавит);
- $\delta : Q \times \Sigma \to Q$ — *всюду определенное* отображение множества $Q \times \Sigma$ в множество Q, определяющее поведение автомата (эту функцию часто называют *функцией переходов*);
- $\lambda : Q \times \Sigma \to \Omega$ — *всюду определенное* отображение множества $Q \times \Sigma$ в множество Ω, определяющее выходную последовательность автомата (эту функцию часто называют *функцией выходов*).

Автомат начинает работу в состоянии q_0, считывая по одному символу входной строки. Считанный символ переводит автомат в новое состояние из Q в соответствии с функцией переходов δ и возвращает соответствующий выходной символ согласно функции выходов λ.

Существуют два основных способа описания конечного автомата.

1. *Диаграмма состояний* (или иногда граф переходов) — графическое представление множества состояний и функции переходов. Представляет собой нагруженный ориентированный граф, вершины которого — состояния автомата, дуги — переходы из одного состояния в другое, а нагрузка — входные/выходные символы, при которых осуществляется данный переход. Если переход из состояния q_i в q_j может быть осуществлен при появлении одного из нескольких символов, то над дугой должны быть надписаны все они.

2. *Таблица переходов* (табличное представление) функции δ. Обычно в такой таблице каждой строке соответствует одно состояние, а столбцу — один допустимый входной символ. В ячейке на пересечении строки и столбца записывается состояние, в которое должен перейти автомат, если в ситуации, когда он находился в данном состоянии, на входе он получил данный символ и соответствующий выходной символ.

[1] Иногда его называют автоматом I рода.

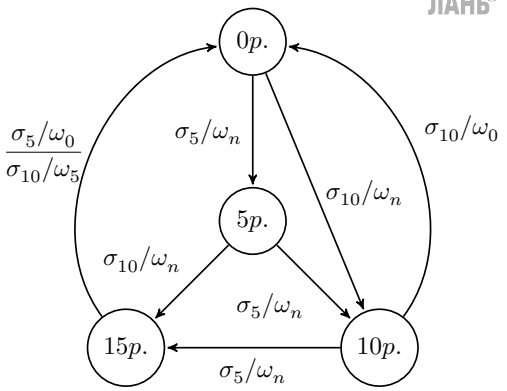
Пример 3.182. Рассмотрим автомат Мили по продаже шоколадок стоимостью 20 рублей, принимающий монеты номиналом в 5 и 10 рублей и возвращающий сдачу, если это необходимо[1].

Состояний автомата четыре: q_0, q_5, q_{10}, q_{15} — 0, 5, 10 и 15 рублей.

Входных сигналов два: σ_5 — 5 рублей и σ_{10} — 10 рублей.

Выходных сигналов три: ω_n — ничего не выдавать, ω_0 — выдать шоколадку и 0 рублей сдачи, ω_5 — выдать шоколадку и 5 рублей сдачи.

На рисунке 3.38*a* приведена диаграмма состояний, а на рисунке 3.38*b* — таблица переходов рассматриваемого автомата.



(a)

	σ_5	σ_{10}
q_0	q_5/ω_n	q_{10}/ω_n
q_5	q_{10}/ω_n	q_{15}/ω_n
q_{10}	q_{15}/ω_n	q_0/ω_0
q_{15}	q_0/ω_0	q_0/ω_5

(b)

Рис. 3.38.

Определение 3.129. Пусть

$$A = (Q_A, \Sigma, \Omega, \delta_A, \lambda_A, q_0), \quad B = (Q_B, \Sigma, \Omega, \delta_B, \lambda_B, p_0)$$

два автомата с одинаковыми входными и выходными алфавитами. Состояние q автомата A и состояние p автомата B называются *неотличимыми*, если для любого слова $\alpha \in \Sigma^*$ выходные слова при достижении q и p совпадают.

Автоматы A и B называются *эквивалентными*, если для любого состояния автомата A найдется неотличимое от него состояние автомата B и наоборот.

Иногда удобным бывает более простое определение автомата, в котором функция переходов задает следующее состояние, а выход автомата зависит лишь от его текущего состояния.

Определение 3.130. Конечный автомат называется *автоматом Мура* или *автоматом II рода*, если для любого $q \in Q$ значения функции выхо-

[1] Пример взят из *Викиконспекты: https://neerc.ifmo.ru/wiki/...*

дов на дугах, входящих в q, совпадают. То есть функция выходов не зависит от входного символа, а только от состояния $\lambda : Q \to \Omega$.

В графическом представлении автоматов Мура часто вершину состояния $q_i \in Q$ помечают также соответствующим выходным символом $w_j \in \Omega$.

Пример 3.183. Пусть на вход автомата подается последовательность из нулей и единиц. Построим автомат Мура, который возвращает накопленное число единиц по модулю 3.

Как и в предыдущем примере 3.182, начальное состояние обозначим q_0. Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.39. Выходной сигнал однозначно определяется состоянием, в которое автомат перешел, поэтому выходные сигналы приписаны прямо к состояниям.

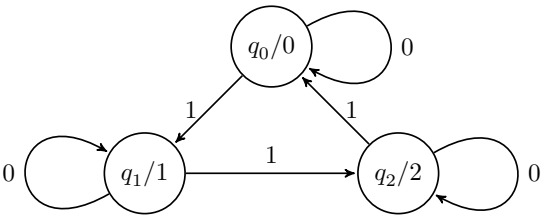


Рис. 3.39.

3.7.3. Примеры автоматов Мили и Мура

Пример 3.184. Реализуем с помощью автомата Мили двоичный последовательный сумматор для одного разряда входного кода.

Автомат имеет два состояния: q_0 — нет переноса единицы, начальное состояние, q_1 — есть перенос. Входной алфавит автомата — это пары битов обоих операндов, последовательно поступающих на сумматор: $(0,0)$, $(0,1)$, $(1,0)$ и $(1,1)$. Выходной алфавит, результат работы сумматора: $\{0,1\}$.

Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.40.

Пример 3.185. Построим автомат Мили, управляющий лифтом в трехэтажном доме.

Входной алфавит автомата состоит из нажатия кнопки вызова соответствующего этажа: $\{C_1, C_2, C_3\}$. Выходной алфавит состоит из перемещений на один или два этажа вверх или вниз, а также остановки лифта:

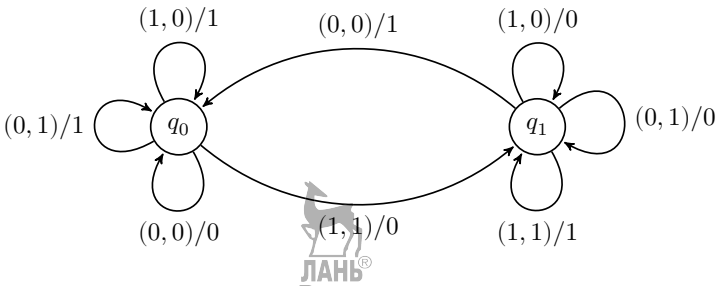


Рис. 3.40.

$\{U_1, U_2, D_1, D_2, S\}$; состояние соответствует этажу, на котором находится автомат: $\{q_1, q_2, q_3\}$; для определенности выберем начальное состояние q_1.

Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.41.

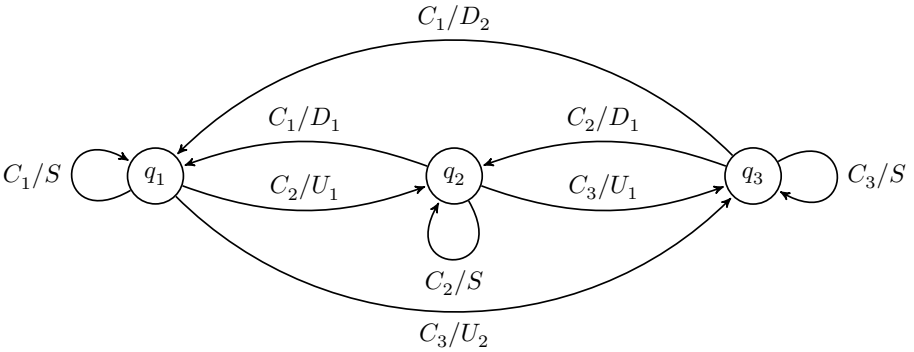


Рис. 3.41.

Пример 3.186. Построим автомат Мили, который удаляет повторяющиеся подряд символы в словах в алфавите $\{a, b, c\}$ (оставляя только один). Например, слово *abbbaccaa* должно преобразоваться в слово *abca*.

В выходной алфавит добавим пустой символ ϵ. Пустой символ будем возвращать при удалении символов входного слова. Начальное состояние обозначим q_0. Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.42.

3.7.4. Эквивалентность автоматов Мили и Мура

Согласно определениям 3.128 и 3.130 автомат Мура есть частный случай автомата Мили. Однако оба автомата являются эквивалентными.

Теорема 3.49 (об эквивалентности автоматов Мили и Мура). Для любого автомата Мили существует эквивалентный ему автомат Мура.

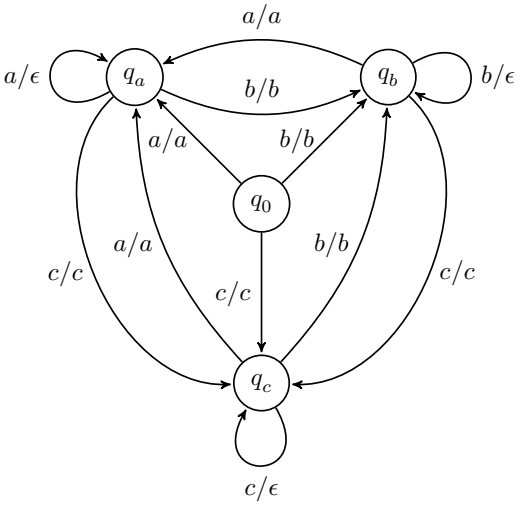


Рис. 3.42.

Доказательство. Преобразование автомата Мура в эквивалентный автомат Мили легко провести в их графическом представлении. Для этого для каждой вершины-состояния символ выходного алфавита Ω, которыми помечена данная вершина, переносим на все дуги, входящие в нее. Очевидно, что получившийся автомат Мили эквивалентен исходному.

На рисунке 3.43 показано данное преобразование для автомата Мура из примера 3.183.

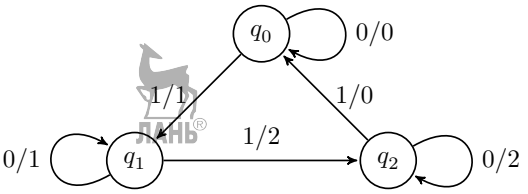


Рис. 3.43.

Рассмотрим алгоритм обратного преобразования автомата Мили в автомат Мура. Для этого выполним действия, обратные действиям при преобразовании автомата Мура в автомат Мили, т. е. выходной символ, которым помечена дуга графа автомата, перенесем в вершину, в которую эта дуга входит. Однако такое преобразование невозможно для случая нескольких входящих в вершину дуг с разными выходными символами. Тогда это состояние необходимо «расщепить» на такое количество состояний, сколько различных входных символов имеется на всех входных ребрах этого состояния, и сопоставить каждому из них свой выходной символ.

Схематически идея «расщепления» состояний показана на рисунке 3.44.

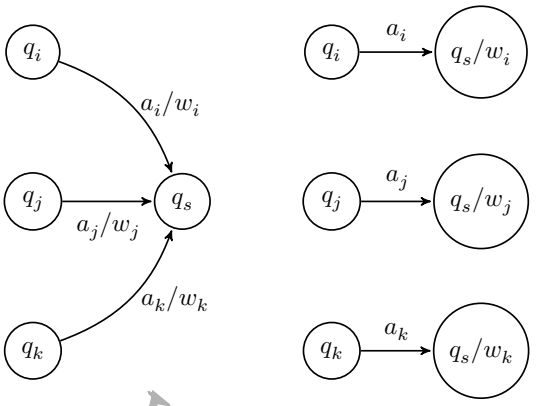


Рис. 3.44.

Рассмотрим эту процедуру на примере автомата Мили на рисунке 3.45.

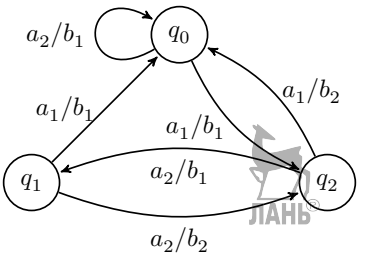


Рис. 3.45.

Каждому состоянию автомата Мили $q_i \in Q$ поставим в соответствие множество пар A_i:

$$A_i = \left\{(q_i, \omega_j) \,|\, q_i = \delta(q_m, \alpha_s), \omega_j = \lambda(q_m, \alpha_s),\ q_m \in Q, \alpha_s \in \Sigma, \omega_j \in \Omega\right\}. \tag{3.95}$$

Согласно (3.95) для автомата на рисунке 3.45 имеем:

$$A_0 = \begin{cases} (q_0, b_1) = p_0, \\ (q_0, b_2) = p_1, \end{cases} \quad A_1 = \{(q_1, b_1) = p_2\}, A_2 = \begin{cases} (q_2, b_1) = p_3, \\ (q_2, b_2) = p_4. \end{cases} \tag{3.96}$$

Таким образом, получаем состояния автомата Мура: $\{p_0, p_1, p_2, p_3, p_4\}$. Обозначим функции переходов и выходов для автомата Мура как $\hat{\delta}$ и $\overline{\lambda}$. Функция выхода определяется вторым элементом соответствующей пары в (3.96).

$$\hat{\lambda}(p_0) = b_1, \hat{\lambda}(p_1) = b_2, \hat{\lambda}(p_2) = b_1, \hat{\lambda}(p_3) = b_1, \hat{\lambda}(p_4) = b_2.$$

Определим функцию переходов для автомата Мура. Рассмотрим произвольное состояние $q_i \in Q$ и произвольный символ входного алфавита $\alpha \in \Sigma$. Пусть $\delta(q_i, \alpha) = q_s$ и $\lambda(q_i, \alpha) = \omega_m \in \Omega$. Тогда в автомате Мура должны быть переходы из всех состояний A_i в состояние $(q_s, \omega_m) \in A_s$, определенное согласно (3.95), под воздействием символа α. Например, в автомате Мили $\delta(q_0, a_1) = q_2, \lambda(q_0, a_1) = b_1$. Тогда для автомата Мура для состояний $p_0, p_1 \in A_0$ имеем $\hat{\delta}(p_0, a_1) = (q_2, b_1) = p_3$ и $\hat{\delta}(p_1, a_1) = p_3$.

В общем виде можно записать таблицу переходов (таблица 3.59).

Таблица 3.59.

$\delta(q_0, a_1) = q_2, \lambda(q_0, a_1) = b_1$	$(q_2, b_1) = p_3$	$\hat{\delta}(p_0, a_1) = \hat{\delta}(p_1, a_1) = p_3$
$\delta(q_0, a_2) = q_0, \lambda(q_0, a_2) = b_1$	$(q_0, b_1) = p_0$	$\hat{\delta}(p_0, a_2) = \hat{\delta}(p_1, a_2) = p_0$
$\delta(q_1, a_1) = q_0, \lambda(q_1, a_1) = b_1$	$(q_0, b_1) = p_0$	$\hat{\delta}(p_2, a_1) = p_0$
$\delta(q_1, a_2) = q_2, \lambda(q_1, a_2) = b_2$	$(q_2, b_2) = p_4$	$\hat{\delta}(p_2, a_2) = p_4$
$\delta(q_2, a_1) = q_0, \lambda(q_2, a_1) = b_2$	$(q_0, b_2) = p_1$	$\hat{\delta}(p_3, a_1) = \hat{\delta}(p_4, a_1) = p_1$
$\delta(q_2, a_2) = q_1, \lambda(q_2, a_2) = b_1$	$(q_1, b_1) = p_2$	$\hat{\delta}(p_3, a_2) = \hat{\delta}(p_4, a_2) = p_2$

В итоге получаем эквивалентный автомат Мура (рисунок 3.46).

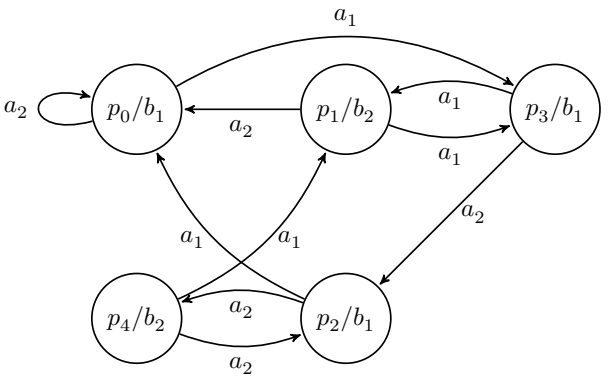


Рис. 3.46.

В качестве начального состояния автомата Мура можно взять любое из состояний p_0 или p_1, так как оба они порождены состоянием q_0 автомата Мили.

■

Таким образом, можно рассматривать только автоматы Мура.

Для задачи определения принадлежности слова языку достаточно ограничиться случаем $|\Omega| = 2$. Тогда каждому состоянию может соответствовать один из двух выходных символов, будем считать, что тем самым со-

стояния разбиваются на два класса, назовем их *заключительным* и *незаключительным*.

Приходим к понятию детерминированного конечного автомата-распознавателя (раздел 3.7.5).

3.7.5. Основные понятия автоматов-распознавателей

Определение 3.131. Детерминированный конечный автомат-распознаватель A — это пятерка объектов $A = (Q, \Sigma, \delta, q_0, F)$, где:

- Q — конечное *непустое* множество состояний;
- $q_0 \in Q$ — начальное состояние;
- Σ — конечное *непустое* множество входных символов (входной алфавит);
- $\delta : Q \times \Sigma \to Q$ — *всюду определенное* отображение множества $Q \times \Sigma$ в множество Q, определяющее поведение автомата. Как и ранее (см. определение 3.128), эту функцию будем называть *функцией переходов*);
- $F \subseteq Q$ — множество заключительных (финальных) состояний.

Автомат начинает работу в состоянии q_0, считывая по одному символу входной строки. Считанный символ переводит автомат в новое состояние из Q в соответствии с функцией переходов δ.

Замечание 3.57

Автомат, описанный в определении 3.131, называется детерминированным, так как для любой пары $q \in Q, a \in \Sigma$ существует единственное состояние $p \in Q$: $p = \delta(q, a)$.

В принципе, для определения последующих действий конечного автомата достаточно знать его текущее состояние и последовательность еще необработанных символов на входной ленте. Этот набор данных называется *конфигурацией автомата*.

Определение 3.132. Слово $w = a_1 \dots a_k$ над алфавитом Σ *допускается* конечным автоматом $M = (Q, \Sigma, \delta, q_0, F)$, если существует последовательность состояний $q_1, q_2, \dots, q_n$ такая, что

$$q_1 = q_0,\ q_n \in F,\ \delta(q_i, a_j) = q_{i+1},\ 1 \leqslant i < n,\ 1 \leqslant j < k.$$

Определение 3.133. Язык L распознается конечным автоматом A, если каждое слово языка L допускается этим конечным автоматом. При этом язык называется *автоматным* (или *регулярным*) и обозначается L_A.

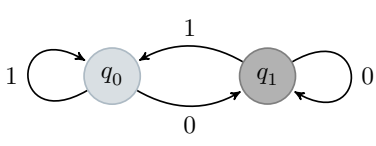
Определение 3.134. Два состояния автомата называются *эквивалентными*, если для любой строки $\alpha \in \Sigma^*$ результаты ее обработки (допуск или недопуск автоматом), начиная с указанных состояний, совпадают.

Определение 3.135. Автоматы A и B называются эквивалентными, если совпадают распознаваемые ими языки, т. е. $L_A = L_B$.

3.7.6. Примеры автоматов-распознавателей

Рассмотрим диаграммы состояний простейших автоматов. Далее, на рисунках начальное состояние будем отмечать светло-серым цветом, конечные состояния — темно-серым.

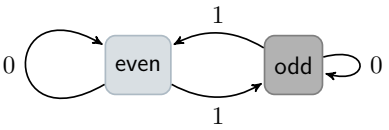
Пример 3.187. На рисунке 3.47*a* приведена диаграмма состояний, а на рисунке 3.47*b* — таблица переходов детерминированного конечного автомата, допускающего цепочки из 0 и 1, заканчивающиеся символом 0.



(a)

	0	1
q_0	q_1	q_0
q_1	q_1	q_0

(b)



(c)

Рис. 3.47.

Пример 3.188. Рассмотрим диаграмму состояний автомата, контролирующего нечетность единиц, то есть распознающего цепочки, состоящих из 0 и 1 и имеющие нечетное число единиц (рисунок 3.47*c*). Начальное состояние: $q_0 = \text{even}$.

Пример 3.189. Рассмотрим автомат, распознающий слова, содержащие только парные вхождения букв a и b, например $aa, aabbaaaa, bbaa$ и т. д. Диаграмма состояний автомата приведена на рисунке 3.48.

Замечание 3.58

Заметим, что автомат из примера 3.189, попав в незаключительное состояние q_3 под воздействием сигналов a, b, не может выйти из него под воздействием тех же входных сигналов (состояние и переходы отмечены на диаграмме 3.48 пунктиром). Будем считать, что переходы автомата в это состояние под воздействием сигналов a, b запрещены. То есть запрещены сигнал b в состоянии q_1 и сигнал a в состоянии q_2. Далее запрещенные состояния и переходы в диаграмме состояний не отображаем, считая, что если символ входного слова привел к запрещенному переходу, то данное слово не принимается автоматом. Такие состояния (в которые

переходят при обработке любого недопустимого символа) в теории часто называют невозвратными.

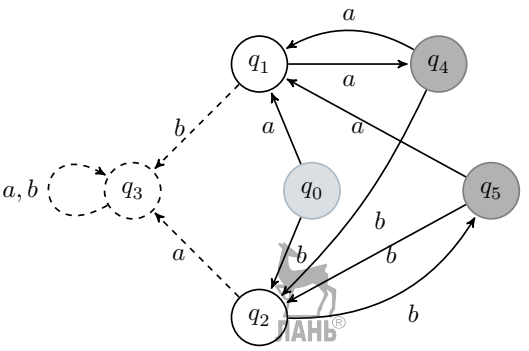


Рис. 3.48.

Пример 3.190. С учетом замечания 3.58 построим автомат, распознающий слова над алфавитом $\{\alpha, \beta\}$, начинающиеся с символов $\alpha\alpha$ и содержащие нечетное число вхождений символа β. Его диаграмма представлена на рисунке 3.49.

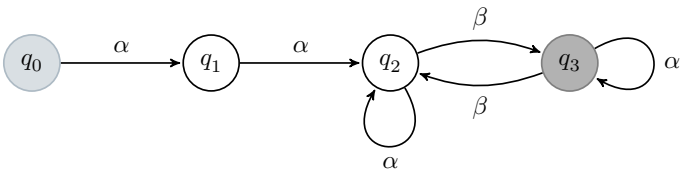


Рис. 3.49.

Пример 3.191. Построим автомат, распознающий слова над алфавитом $\{0, 1\}$, содержащие подслово 00, например $\alpha = 01001$. Его диаграмма представлена на рисунке 3.50.

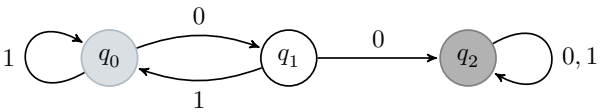


Рис. 3.50.

Пример 3.192. Рассмотрим автомат, распознающий множество вещественных констант ($V_T = \{+, -, 0, \ldots, 9, .\}$) (рисунок 3.51). При этом в константе может быть опущен знак и отсутствовать дробная или целая часть (но не обе сразу). Десятичная точка как признак вещественного числа обязательна. При построении диаграммы учитываем замечание 3.58.

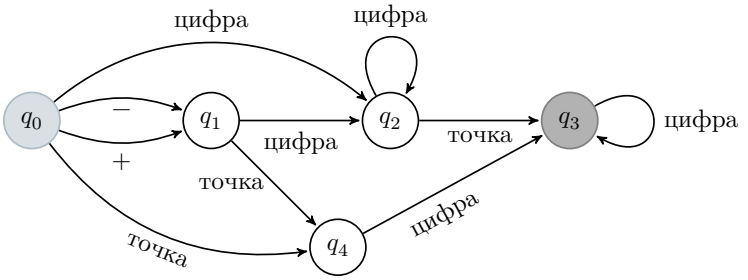


Рис. 3.51.

3.7.7. Недетерминированные автоматы-распознаватели. Детерминизация

Перейдем теперь к недерминированным автоматам-распознавателям, которые являются обобщением детерминированных.

Определение 3.136. Недетерминированный конечный автомат-распознаватель A — это пятерка объектов $A = (Q, \Sigma, \delta, H, F)$, где:

- Q — конечное *непустое* множество состояний;
- Σ — конечное *непустое* множество входных символов;
- $\delta : Q \times \Sigma \to 2^Q$ — функция перехода: всюду определенное отображение множества $Q \times \Sigma$ в множество подмножеств Q:

$$\delta(q_i, \alpha_j) = \left\{q_{i_1}, \dots, q_{i_n}\right\} \subseteq Q,$$

означает, что из состояния q_i по входному символу α_j можно осуществить переход в любое из состояний $q_{i_1}, \dots, q_{i_n}$;

- $H \subseteq Q$ — множество начальных состояний;
- $F \subseteq Q$ — множество заключительных (финальных) состояний.

Замечание 3.59

Отличия от детерминированного автомата (см. определение 3.131) состоят в задании множества начальных состояний H вместо одного q_0 и выходному значению функции перехода δ, которая возвращает подмножество выходных состояний вместо одного.

Для дальнейшего изучения свойств конечных автоматов важное значение имеет следующая теорема.

Теорема 3.50 (о детерминизации). Для любого конечного автомата может быть построен эквивалентный ему детерминированный конечный автомат.

Доказательство. В качестве доказательства приведем два алгоритма: детеминизация объединением и детерминизация вытягиванием для построения детерминированного конечного автомата по недерминированному. ■

Введем необходимые определения.

Определение 3.137. Состояние $q_i \in Q$ автомата A называется *недостижимым*, если под воздействием любого слова автомат A не переходит в состояние q_i.

Состояния, недостижимые из начального, можно найти поиском в ширину, например, используя следующий алгоритм.

Алгоритм 3.23. Нахождение недостижимых состояний

// Инициализация. Заносим в список L начальное состояние
$q_0 \to L$

// Основной цикл
while существуют $q_i \in L$, $q_j \notin L$ и $\alpha_k \in \Sigma$, $q_j = \delta(q_i, \alpha_k)$ **do**
  // Добавляем в список L новые состояния, достижимые из состояний L
  $L \leftarrow L \cup \{q_j\}$
end while
Состояния, не включенные в список L, удаляем со всеми инцидентными им дугами

Рассмотрим теперь алгоритм построения детерминированного конечного автомата по недерминированному.

На вход алгоритма подается недетерминированный конечный автомат

$$A = (Q, \Sigma, \delta, H, F)\,.$$

В результате работы получаем детерминированный конечный автомат

$$\widetilde{A} = \left(\widetilde{Q}, \widetilde{\Sigma}, \tilde{\delta}, \widetilde{q_0}, \widetilde{F}\right),$$

допускающий тот же язык, что и автомат A.

Алгоритм 3.24. Детерминизация объединением

1: Определяем $\widetilde{Q}$ как множество всех подмножеств Q. Обозначим $\widetilde{q_i} \in \widetilde{Q}$ как

$$\widetilde{q_i} = \left[q_{i_1} \dots q_{i_k}\right],\ q_{i_j} \in Q,\ 1 \leqslant j \leqslant k.$$

2: Определяем отображение $\tilde{\delta}$ как объединение состояний:

$$\tilde{\delta}\left(\left[q_{i_1} \dots q_{i_n}\right], \alpha\right) = \left[q_{j_1} \dots q_{j_m}\right],$$

где

$$\forall q_{j_s} \in \left\{q_{j_1}, \dots, q_{j_m}\right\},\ \exists q_{i_p} \in \left\{q_{i_1}, \dots, q_{i_n}\right\} : \delta\left(q_{i_p}, \alpha\right) = q_{j_s}.$$

3: Определяем начальное состояние $\widetilde{q_0}$ как объединение *всех* начальных состояний H:

$$\widetilde{q_0} = \left[q_{0_1} \dots q_{0_k}\right],\ q_{0_i} \in H,\ 1 \leqslant i \leqslant k,\ k = |H|\,. \tag{3.97}$$

4: Определяем множество заключительных состояний $\widetilde{F}$:

$$\widetilde{F} = \left\{\widetilde{q_i} \in \widetilde{Q} \mid \widetilde{q_i} = [\dots q_l \dots],\ q_l \in F\right\}.$$

5: Используя алгоритм 3.23, исключаем возможные недостижимые состояния в $\widetilde{Q}$.

Замечание 3.60

Если $|Q| = n$, то после первого шага $\left|\widetilde{Q}\right| = 2^n$. Таким образом, на шаге 2 необходимо перебрать 2^n возможных состояний нового детерминированного автомата, а отсечение недостижимых состояний происходит на шаге 5.

Объединение состояний на шаге 2 алгоритма 3.24 иллюстрирует рисунок 3.52

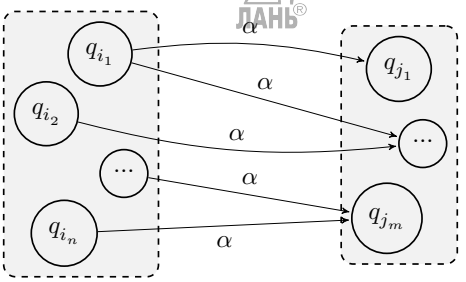
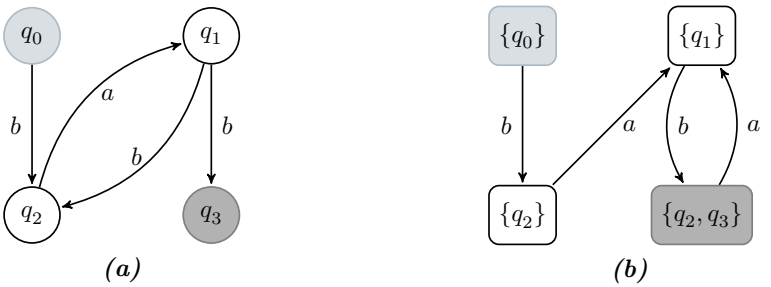


Рис. 3.52.

Пример 3.193. На рисунке 3.53*b* изображен детерминированный автомат $\widetilde{A}$, построенный по алгоритму 3.24 из недетерминированного автомата A на рисунке 3.53*a*.

Всего согласно алгоритму 3.24 у нового автомата $\widetilde{A}$ должно быть $2^4 = 16$ состояний, но только 4 из них оказываются достижимыми.



(a) ***(b)***

Рис. 3.53.

В алгоритме 3.24 отсечение недостижимых состояний происходит на последнем шаге. Рассмотрим другой подход, где достижимость состояния нового детерминированного автомата определяется сразу при построении.

Процесс добавления состояний-множеств напоминает расширение списка L в алгоритме 3.23.

Алгоритм 3.25. Детерминизация вытягиванием

// Инициализация. Добавляем начальное состояние.

Начальное состояние $\widetilde{q_0} \in \widetilde{Q}$ определяем согласно (3.97).

// Основной цикл

while существует $\widetilde{q_j} = \delta(\tilde{q}, \alpha)$, $\tilde{q} \in \widetilde{Q}$, $\alpha \in \Sigma \mid \widetilde{q_j} \notin \widetilde{Q}$ **do**

// Добавляем новые состояния, непосредственно достижимые из текущего $\widetilde{Q}$

$\widetilde{Q} \leftarrow \widetilde{Q} \cup \{\widetilde{q_j}\}$

end while

Замечание 3.61

На шаге k основного цикла добавляются состояния, достижимые из начального состояния $\widetilde{q_0}$ по пути длиной k.

Пример 3.194. Рассмотрим работу алгоритма 3.25 на примере недетерминированного автомата (рисунок 3.53a). Имеем:

$$\delta_1(\{q_0\}, b) = \{q_2\};$$
$$\delta_1(\{q_2\}, a) = \{q_1\};$$
$$\delta_1(\{q_1\}, b) = \{q_2, q_3\};$$
$$\delta_1(\{q_2, q_3\}, a) = \delta(q_2, a) \cup \delta(q_3, a) = \{q_1\} \cup \emptyset = \{q_1\}.$$

Так как новых состояний-множеств больше не появилось, процедура «вытягивания» на этом заканчивается, и мы получаем граф, изображенный на рисунке 3.53b.

Пример 3.195. Рассмотрим работу алгоритма 3.25 на примере недетерминированного автомата (рисунок 3.54). Имеем:

$$\delta_1(\{q_0\}, a) = \{q_0, q_1\};$$
$$\delta_1(\{q_0, q_1\}, b) = \{q_2\};$$
$$\delta_1(\{q_2\}, a) = \{q_0, q_1, q_3\};$$
$$\delta_1(\{q_0, q_1, q_3\}, b) = \{q_0, q_2\};$$
$$\delta_1(\{q_0, q_2\}, a) = \{q_0, q_3\}.$$

Результат работы алгоритма 3.25 показан на рисунке 3.55.

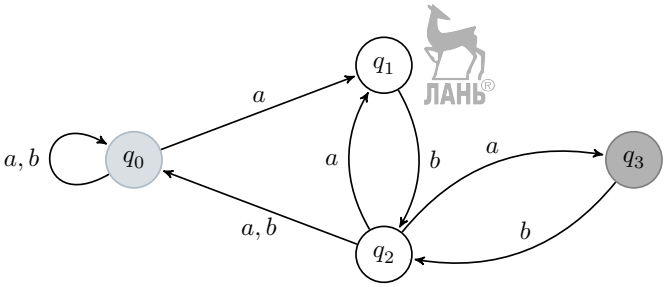


Рис. 3.54.

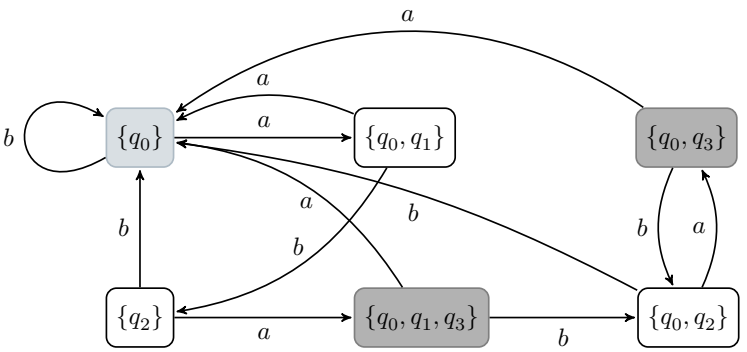


Рис. 3.55.

3.7.8. Теорема о разрастании для автоматных языков

Очевидно, что любой конечный язык может быть распознан конечным автоматом, поэтому *все конечные языки автоматные*. Один и тот же язык может распознаваться разными автоматами. Однако не для всех языков существуют распознающие их конечные автоматы. Иными словами, *существуют и неавтоматные языки*.

Установим некоторое необходимое условие, которому удовлетворяют все автоматные языки. После этого, проверив, что некоторый язык этому условию не удовлетворяет, можно заключить, что он не является автоматным. Очевидно, что это условие имеет смысл формулировать только для бесконечных языков.

Теорема 3.51 (о разрастании для автоматных языков). Пусть L — бесконечный автоматный язык. Тогда существует такая константа n, что любое слово $w \in L$ длиной $|w| > n$ можно разбить на три части $\alpha_1, \alpha_2, \alpha_3$ так, что $w = \alpha_1\alpha_2\alpha_3$ и

1) $|\alpha_1\alpha_2| \leqslant n$;
2) $|\alpha_2| > 0$;
3) $\forall m \geqslant 0 \quad w_m = \alpha_1\alpha_2^m\alpha_3 \in L$.

В последнем условии $\alpha_2^0 = \varepsilon,\ \alpha_2^1 = \alpha_2,\ \alpha_2^{i+1} = \alpha_2^i \alpha_2$.

Доказательство. Так как язык L автоматный, то существует детерминированный конечный автомат $A = (Q, \Sigma, \delta, q_0, F)$, распознающий L. Пусть $|Q| = n$ и слово $w = w_1 w_2 \dots w_k \in L$ имеет длину $k > n$. Рассмотрим путь

$$p = \left(q_0 = q_{i_0}, q_{i_1}, \dots, q_{i_k}\right)$$

в диаграмме автомата A, который принимает слово w. Очевидно, что среди первых $(n+1)$ состояний этого пути хотя бы одно встречается дважды. Выберем первое из таких состояний — $q \in Q$. Тогда для некоторой пары чисел $l < j \leqslant n$ имеем $q_{i_l} = q_{i_j} = q$.

Пусть $\alpha_1 = w_1 w_2 \dots w_l$ — это префикс w, который переводит q_0 в $q_{i_l} = q$, $\alpha_2 = w_{i+1} \dots w_j$ — это подслово w, которое переводит $q_{i_l} = q$ в $q_{i_j} = q$, и $\alpha_3 = w_{j+1} \dots w_k$ — это суффикс w, который переводит $q_{i_j} = q$ в $q_{i_k} \in F$.

Части α_1 и α_3 могут быть пустыми, но $|\alpha_2| = j - l \geqslant 1$. Длина $|\alpha_1 \alpha_2| = j \leqslant n$. Таким образом, условия (1) и (2) теоремы выполнены. Нетрудно убедиться и в выполнении условия (3).

Действительно, выбросив из пути p цикл $q_{i_l} = q, \dots, q_{i_j}$, получим путь p_0 из q_0 в $q_{i_k} \in F$, который принимает слово $\alpha_1 \alpha_3$, а повторив этот цикл m раз, получим путь p_0 из q_0 в $q_{i_k} \in F$, который принимает слово $\alpha_1 \alpha_2^m \alpha_3$. Следовательно,

$$\forall m \geqslant 0 \quad w_m = \alpha_1 \alpha_2^m \alpha_3 \in L.$$

■

Замечание 3.62

Содержательно теорема 3.51 утверждает, что у всякого достаточно длинного слова из автоматного языка имеется непустое подслово, которое можно вырезать или повторить сколько угодно раз, оставаясь внутри языка.

Пример 3.196. Рассмотрим автомат с диаграммой, представленной на рисунке 3.56. Проиллюстрируем теорему 3.51 для слова $w = 1101111$. Имеем $n = 4, |w| = 7 > 4$. Путь p для слова w:

$$p = q_0 \xrightarrow{1} q_2 \xrightarrow{1} \underbrace{q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1}_{\text{цикл}} \xrightarrow{1} q_3 \xrightarrow{1} q_3 \xrightarrow{1} q_3.$$

Повторение состояний $q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1$. Тогда

$$w = \alpha_1 \alpha_2 \alpha_3, \quad \alpha_1 = 11, \quad \alpha_2 = 01, \quad \alpha_3 = 111.$$

При этом

$$|\alpha_1 \alpha_2| = n = 4,\ |\alpha_2| = 2.$$

Очевидно, что слово

$$\alpha_1\alpha_2^m\alpha_3 \in L,\ \forall m \geqslant 0.$$

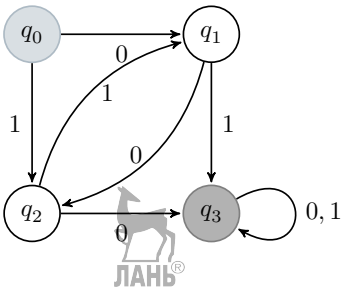


Рис. 3.56.

Замечание 3.63

Теорема 3.51 имеет другие и названия в литературе: лемма о накачке, лемма Огдена, pumping lemma.

Как, используя теорему 3.51, доказать, что некоторый язык L не является автоматным? Это можно сделать, используя схему доказательства от противного.

1. Предположим, что L — автоматный язык. Тогда для него имеется константа n из утверждения теоремы 3.51.

2. Определим по n «специальное» слово $w \in L, |w| > n$ и докажем, что для любого разбиения $w = \alpha_1\alpha_2\alpha_3$, удовлетворяющего условиям (1) и (2) теоремы, найдется такое $k \geqslant 0$, что слово $w_k = \alpha_1\alpha_2^k\alpha_3 \notin L$.

3. На основании полученного противоречия делаем вывод, что L не является автоматным языком.

Замечание 3.64

В этой схеме самым сложным является выбор «специального» слова w в пункте (2). Что касается подбора такого $k \geqslant 0$, для которого $w_k \notin L$, то, как правило, достаточно рассмотреть $k = 0$ или $k = 2$.

Рассмотрим несколько примеров применения данной схемы.

Пример 3.197. Покажем, что язык $L_1 = \{w = 0^k\,1^k \mid k \geqslant 1\}$ не является автоматным.

Предположим, что он автоматный. Тогда для него имеется n из утверждения теоремы 3.51. Рассмотрим «специальное» слово $w = 0^n\,1^n$. Очевидно, что $w \in L_1$. Предположим, что существует разбиение $w = \alpha_1\alpha_2\alpha_3$, удовлетворяющее условиям (1) и (2) теоремы.

Так как по условию (2) $|\alpha_1\alpha_2| \leqslant n$, то $\alpha_2 = 0^i$ для некоторого $i > 0$. Но тогда слово $w_0 = \alpha_1\alpha_3 = 0^{n-i}1^n \notin L_1$, что противоречит условию (3) теоремы. Следовательно, язык L_1 не является автоматным.

Пример 3.198. Покажем, что язык правильных скобочных последовательностей L_2 в алфавите $\{(,)\}$ не является автоматным.

В качестве «специального» слова выберем слово $w = (^n)^n \in L_2$. Тогда для всякого разбиения $w = \alpha_1\alpha_2\alpha_3$ такого, что $|\alpha_1\alpha_2| \leqslant n$, слово $\alpha_2 = (^i$ для некоторого $i > 0$. И, как и в предыдущем примере 3.197, слово $w_0 = \alpha_1\alpha_3 = (^{n-i})^n \notin L_2$, что противоречит условию (3) теоремы. Следовательно, язык L_2 не автоматный.

Упражнение 3.28. С помощью теоремы 3.51 покажите, что язык

$$L_3 = \{w = 0^k\,1\,0^k \,|\, k \geqslant 1\}$$

не является автоматным.

3.7.9. Автоматы и автоматные грамматики

Установим связь между автоматными грамматиками G_3 и недерминированными автоматами-распознавателями.

Определение 3.138. Порождающая грамматика G и конечный автомат-распознаватель A называются *эквивалентными*, если $L(G) = L(A)$.

Теорема 3.52. Для каждой праволинейной грамматики существует эквивалентный ей конечный автомат.

Доказательство. Каждому нетерминальному символу произвольной праволинейной грамматики G поставим в соответствие одно состояние конечного автомата A. Добавим еще одно состояние T — единственное конечное состояние. Состояние, соответствующее аксиоме грамматики S, будем считать начальным состоянием. Каждому правилу $A \to aB$ поставим в соответствие команду $Aa \to B$, а каждому терминальному правилу $A \to a$ поставим в соответствие команду $Aa \to T$. Таким образом, выводу цепочки в грамматике

$$S \Rightarrow a_1A_1 \Rightarrow a_1a_2A_2 \Rightarrow \ldots \Rightarrow a_1a_2\ldots a_{k-1}A_{k-1} \Rightarrow a_1a_2\ldots a_k$$

взаимно однозначно соответствует последовательность команд построенного автомата A:

$$Aa_1 \to A_1;\ A_1a_2 \to A_2;\ \ldots;\ A_{k-1}a_k \to T.$$

Таким образом, язык $L(G) = L(A)$. ■

Пример 3.199. Для грамматики G, заданной следующими правилами вывода:

$$S \to aS|bB; \quad A \to aA|bS; \quad B \to bB|c|cA,$$

построим эквивалентный ей конечный автомат A.

Граф автомата (рисунок 3.57) будет иметь четыре вершины, три из них помечены нетерминальными символами грамматики S, A, B, четвертая вершина, помеченная символом T, является единственным заключительным состоянием. Начальным состоянием является вершина, соответствующая аксиоме грамматики S.

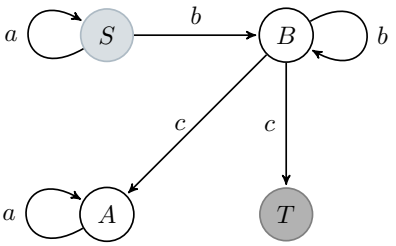


Рис. 3.57.

Согласно теореме 3.52 каждому правилу грамматики поставим в соответствие команду конечного автомата:

1) $Sa \to S$ — в начальном состоянии при поступлении на вход терминального символа a автомат остается в том же состоянии S;

2) $Sb \to B$ — из начального состояния при поступлении на вход терминального символа b автомат переходит в состояние B;

3) $Bb \to B$ — в состоянии B при поступлении на вход терминального символа b автомат остается в том же состоянии B;

4) $Bc \to F$ — из состояния B при поступлении на вход терминального символа c автомат переходит в заключительное состояние T;

5) $Bc \to A$ — из состояния B при поступлении на вход терминального символа c автомат переходит в состояние A;

6) $Aa \to A$ — в состоянии A при поступлении на вход терминального символа a автомат остается в этом же состоянии A;

7) $Ab \to S$ — из состояния A при поступлении на вход терминального символа b автомат переходит в состояние S.

Полученный *недетерминированный* конечный автомат распознает цепочки языка, порождаемые праворекурсивной грамматикой G.

Пример 3.200. Для грамматики G_4 из примера 3.163 построим эквивалентный ей конечный автомат A_4. Правила вывода для этой грамматики согласно (3.90) имеют вид:

$$S \to aA,\ A \to aA \,|\, bC,\ C \to cC \,|\, c.$$

Грамматика G_4 является регулярной (праволинейной) грамматикой и порождает язык $L(G_4) = \{a^n bc^m \,|\, n, m > 0\}$.

Согласно теореме 3.52 каждому правилу грамматики G_4 поставим в соответствие команду конечного автомата. Рассуждая аналогично примеру 3.199, получаем следующий автомат (рисунок 3.58).

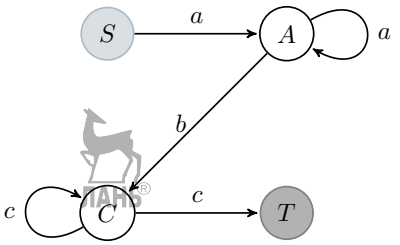


Рис. 3.58.

Полученный *недетерминированный* конечный автомат распознает цепочки языка, порождаемые праворекурсивной грамматикой G_4, например слово *aaabcc* из примера 3.163.

Пример 3.201. Построим конечный автомат, распознающий язык

$$L(A) = \{(ab)^n \,|\, n \in N\}.$$

Сначала построим саму грамматику G, которая порождает язык $L(A)$:

$$S \to aA; \quad A \to bS|b.$$

Проверим, действительно ли эта грамматика порождает язык $L(A)$. Для этого построим несколько выводов возможных вариантов цепочек:

1) $S \Rightarrow aA \Rightarrow ab$;
2) $S \Rightarrow aA \Rightarrow abS \Rightarrow abaA \Rightarrow abab$;
3) $S \Rightarrow aA \Rightarrow abS \Rightarrow abaA \Rightarrow ababS \Rightarrow ababaA \Rightarrow abababa$.

и т.д.

Таким образом, грамматика G действительно порождает язык $L(A)$, следовательно, можно построить соответствующий этой грамматике конеч-

ный автомат (рисунок 3.59). Для этого введем заключительное состояние T, начальное состояние соответствует аксиоме грамматики S.

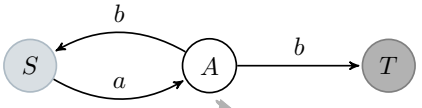


Рис. 3.59.

Запишем преобразование правил вывода грамматики G в команды автомата A:

1) $Sa \to A$ — из состояния S при поступлении на вход терминального символа a автомат переходит в состояние A;

2) $Ab \to S$ — из состояния A при поступлении на вход терминального символа b автомат переходит в состояние S;

3) $Ab \to F$ — из состояния A при поступлении на вход терминального символа b автомат переходит в заключительное состояние T.

Таким образом, построенный недетерминированный конечный автомат A распознает заданный язык $L(G)$.

Легко установить и утверждение, обратное к теореме 3.52.

Теорема 3.53. Для произвольного конечного автомата-распознавателя существует эквивалентная порождающая автоматная праволинейная грамматика.

Доказательство. Каждому состоянию произвольного автомата поставим в соответствие нетерминальный символ грамматики, причем начальному состоянию будет соответствовать аксиома грамматики. Тогда для каждой команды $Ac \to B$ в множество правил грамматики включим правило $A \to cB$, причем в случае, если B — заключительное состояние, добавим правило $A \to c$. Эквивалентность исходного конечного автомата и построенной грамматики очевидна. ■

Пример 3.202. Построим порождающую автоматную праволинейную грамматику по недерминированному конечному автомату-распознавателю на рисунке 3.60.

Введя согласно теореме 3.53 обозначения для порождающей грамматики G, получаем набор правил:

$$q_0 \to S,\ q_1 \to A, \qquad G:\ S \to aS \,|\, aA,\ A \to bA \,|\, b.$$

Построенная грамматика G порождает язык

$$L(G) = \{a^n b^m \,|\, n \geqslant 0, m > 0\}.$$

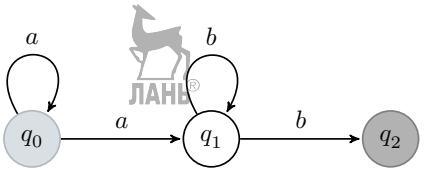


Рис. 3.60.

3.7.10. Минимизация автоматов-распознавателей

Поставим задачу нахождения наименьшего (по количеству состояний) конечного автомата, распознающего данный язык. Введем необходимые определения.

Определение 3.139. На множестве состояний Q автомата A зададим семейство отношений эквивалентности следующим образом:

а) 0-*эквивалентность*: для произвольных состояний $q_1, q_2 \in Q$ полагаем $q_1 \overset{0}{\equiv} q_2$ тогда и только тогда, когда они оба являются заключительными или оба не являются заключительными;

б) n-*эквивалентность*: при $n \geqslant 1$ полагаем $q_1 \overset{n}{\equiv} q_2$ тогда и только тогда, когда

$$q_1 \overset{n-1}{\equiv} q_2, \quad \delta(q_1, \alpha) \overset{n-1}{\equiv} \delta(q_2, \alpha), \ \forall \alpha \in \Sigma. \tag{3.98}$$

Замечание 3.65

Предлагаемый ниже алгоритм 3.26 нахождения минимального автомата был предложен Ауфенкампом и Хоном в [70].

Алгоритм 3.26. Построение минимального автомата

// Инициализация

Удаляем все недостижимые состояния, используя алгоритм 3.23.

Разбиваем множество состояний Q на два подмножества 0-эквивалентных состояний: $S_1 = F$ и $S_2 = Q - F$.

$n \leftarrow 0$

// Основной цикл

do

Разбиваем каждое из подмножеств n-эквивалентных состояний на подмножества $(n+1)$-эквивалентных состояний.

$n \leftarrow n + 1$

while пока удается разбить хотя бы одно подмножество

// Получаем набор подмножеств эквивалентных состояний $S_1, \ldots, S_k$

Для множества состояний минимизированного автомата берем по одному представителю каждого из множеств S_i.

При работе алгоритма 3.26 возможны два крайних случая:

1) все состояния исходного автомата окажутся эквивалентными, и тогда в минимальном автомате останется только одно состояние; этот случай проиллюстрирован на рисунке 3.61a (исходный автомат) и b (минимизированный автомат) — это автомат, допускающий произвольные цепочки символов a, b;

2) итоговое разбиение будет состоять из одноэлементных классов эквивалентности, это означает, что исходный автомат нельзя минимизировать, он уже минимален.

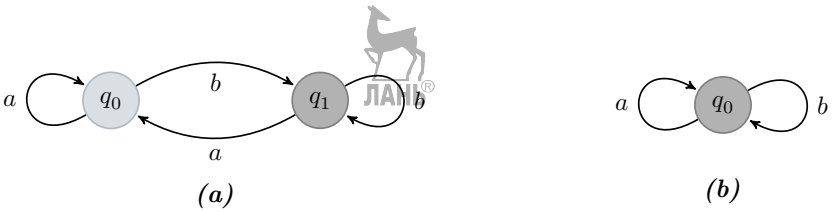


Рис. 3.61.

Замечание 3.66

Алгоритм 3.26 часто называют методом разбиения, а сам процесс приведения автомата к минимальному — редукцией конечного автомата.

Пример 3.203. Рассмотрим процесс минимизации автомата, диаграмма состояний которого представлена на рисунке 3.62.

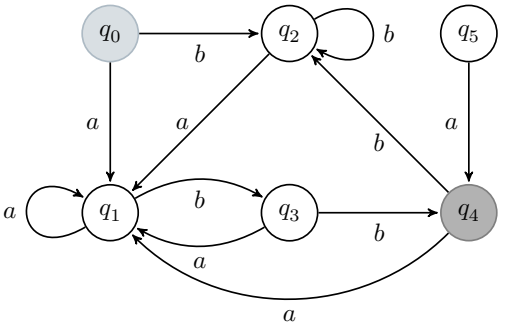


Рис. 3.62.

1 Согласно алгоритму 3.23 вначале произведем поиск и удаление недостижимых состояний. Получаем список достижимых состояний:

$$\{q_0\} \to \{q_0, q_2\} \to \{q_0, q_2, q_1\} \to \{q_0, q_2, q_1, q_3\} \to \{q_0, q_2, q_1, q_3, q_4\}.$$

Состояние q_5 недостижимо, удаляем его вместе с дугой $[q_5, q_4]$ и производим разбиение множества достижимых состояний автомата на классы эквивалентности.

2 Запишем начальное разбиение множества состояний автомата для отношения $\overset{0}{\equiv}$:

$$\{q_4\},\ \{q_0, q_1, q_2, q_3\}.$$

3 Заметим, что для состояний q_0, q_1, q_2 автомат переходит в одно из состояний класса эквивалентности предыдущего уровня — $\{q_0, q_1, q_2, q_3\}$, поэтому все они 1-эквивалентны.

Но состояние $\delta(q_3, b) = q_4 \notin \{q_0, q_1, q_2, q_3\}$, поэтому согласно (3.98) состояние q_3 не 1-эквивалентно состояниям q_0, q_1, q_2. В разбиении для 1-эквивалентности они «разойдутся» по разным классам; разбиение, определяемое отношением $\overset{1}{\equiv}$, будет иметь вид:

$$\{q_4\},\ \{q_0, q_1, q_2\},\ \{q_3\}.$$

4 Далее, для состояний q_0 и q_2 автомат переходит в одно из состояний класса эквивалентности предыдущего уровня $\{q_0, q_1, q_2\}$, поэтому они 2-эквивалентны.

Однако состояние $\delta(q_1, b) = q_3 \notin \{q_0, q_1, q_2\}$, поэтому q_1 не 2-эквивалентно q_0 и q_2. Произойдет разделение; разбиение, определяемое отношением $\overset{2}{\equiv}$, будет иметь вид:

$$\{q_4\},\ \{q_0, q_2\},\ \{q_1\},\ \{q_3\}.$$

5 Заметим, что согласно (3.98) $q_0 \overset{3}{\equiv} q_2$. Действительно,

$$q_0, q_2, \delta(q_0, b), \delta(q_2, b) \in \{q_0, q_2\},\quad \delta(q_0, a), \delta(q_2, a) \in \{q_1\}.$$

Поэтому разбиение на классы 2-эквивалентности и есть искомое разбиение. Таким образом, состояния q_0 и q_2 неразличимы. Полученный минимизированный автомат представлен на рисунке 3.63.

Пример 3.204. Проведем минимизацию автомата, диаграмма состояний которого представлена на рисунке 3.64.

1 Используя алгоритм 3.23, удаляем недостижимые состояния: q_6, q_5.

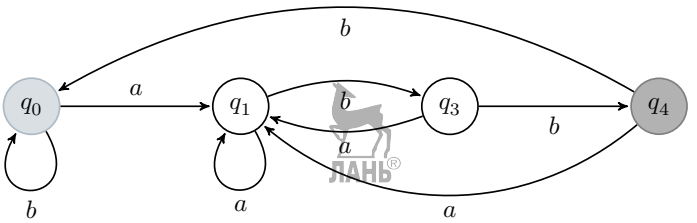


Рис. 3.63.

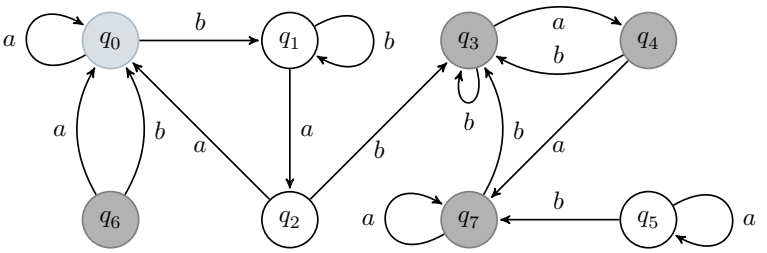


Рис. 3.64.

2 Определяем начальное разбиение множества состояний автомата для отношения $\overset{0}{\equiv}$:

$$\{q_0, q_1, q_2\},\ \{q_3, q_4, q_7\}.$$

3 Для состояний q_0 и q_1 автомат переходит в одно из состояний класса эквивалентности предыдущего уровня — $\{q_0, q_1, q_2\}$, поэтому они 1-эквивалентны. Но состояние $\delta(q_2, b) = q_3 \notin \{q_0, q_1, q_2\}$, следовательно, q_3 не 1-эквивалентно q_0 и q_1. Для отношения $\overset{1}{\equiv}$ получаем новое разбиение:

$$\{q_0, q_1\},\ \{q_2\},\ \{q_3, q_4, q_7\}.$$

4 Далее, состояния $\delta(q_0, a) = q_0$ и $\delta(q_1, a) = q_2$ не 1-эквивалентны, для отношения $\overset{2}{\equiv}$ получаем разбиение:

$$\{q_0\},\ \{q_1\},\ \{q_2\},\ \{q_3, q_4, q_7\}.$$

5 Поскольку для всех состояний из множества $\{q_3, q_4, q_7\}$ автомат переходит в одно из этих же состояний, то разбиение на классы 2-эквивалентности и есть искомое разбиение. Таким образом, состояния q_3, q_4, q_7 неразличимы. Минимизированный автомат представлен на рисунке 3.65.

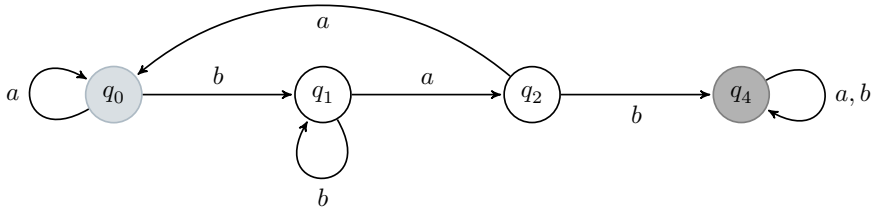


Рис. 3.65.

3.7.11. Конечные автоматы с эпсилон-переходами

Определение 3.140. *Эпсилон-переходом* называется переход между состояниями, который может быть выполнен автоматом «просто так», без входного символа. На графах и в таблицах такие переходы обычно помечаются символом ϵ.

Одно из наиболее полезных свойств ϵ-переходов — возможность простого объединения нескольких автоматов в один. При этом если на вход подаются детерминированные автоматы, то при объединении может получиться недерминированный автомат. Рассмотрим этот прием на простом примере.

Пример 3.205. На рисунке 3.66 изображены два исходных конечных автомата.

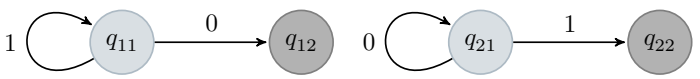


Рис. 3.66.

На рисунке 3.67 показан результат объединения двух входных автоматов.

Никаких дополнительных ограничений на ϵ-переходы при работе конечного автомата не накладывается.

1. Из одного состояния недерминированного конечного автомата может выходить сколько угодно как обычных, так и ϵ-переходов.

2. Может существовать цепочка из нескольких последовательных эпсилон-переходов.

3. Автомат может проходить цепочку целиком, частично или не проходить ее вообще (если у автомата есть другие варианты поведения).

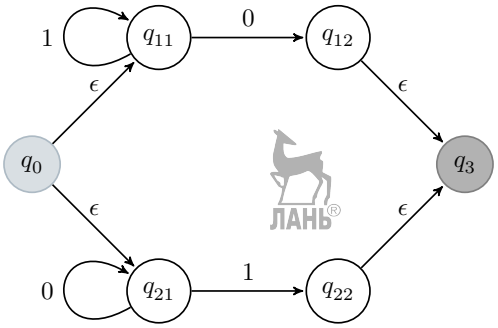


Рис. 3.67.

В общем, граф/таблица переходов автомата выглядят так, как будто во входном алфавите появился еще один символ — ϵ, а его работа так, будто в любом месте обрабатываемой цепочки (в начале, в конце, между символами) находится произвольное количество этих символов. При этом для каждого состояния есть как минимум один переход по ϵ — в себя.

Замечание 3.67

Нетрудно показать, что по недерминированному автомату с ϵ-переходами можно построить эквивалентный ему конечный детерминированный автомат, но проектировать ϵ-автомат гораздо проще и удобнее.

Действительно, уберем ϵ-переходы следующим образом. Найдем все пары состояний (q_k, q_l) такие, что q_l достижимо из q_k по ϵ-переходам:

$$q_k \xrightarrow{\epsilon} \dots q_i \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} q_l$$

Для каждого не-ϵ-перехода $\delta(q_l, \alpha) = q_s$ добавим переход $\delta(q_k, \alpha) = q_s$. Кроме того, если $q_l \in F$, добавим и q_k в F. Далее, при необходимости нужно применить один из рассмотренных выше алгоритмов детерминизации (3.24 или 3.25).

Пример 3.206. Уберем ϵ-переходы для объединенного автомата из примера 3.205 (рисунок 3.67). На рисунке 3.68 представлен результат. Детерминизация не потребовалась.

3.7.12. Конечные автоматы и регулярные выражения

Регулярные выражения являются достаточно удобным средством для построения «алгебраических» описаний языков. Они строятся из элементарных выражений с помощью операций *объединения* $(+, |, \cup)$[1], *конкатенации*

[1] В различных описаниях возможно использование одного из данных символов.

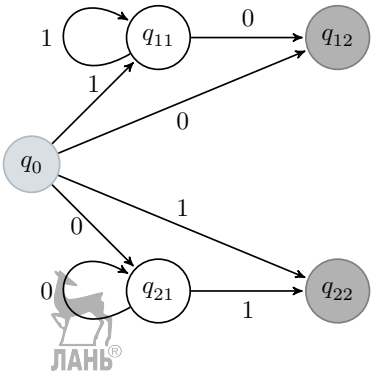


Рис. 3.68.

(сцепления) $\hat{}$ и *итерации* $(*)$. Каждому такому выражению r соответствует представляемый им язык L_r.

Замечание 3.68

Другие общеупотребительные названия регулярных выражений: Regular Expressions, RegExp. Активно используются в языках программирования Python, Perl, Ruby, Java, .Net и в текстовых редакторах Vim, EmEdit.

Введем необходимые определения.

Определение 3.141. *Объединение* языков $L_1 + L_2$ содержит все слова, содержащиеся или в L_1, или в L_2.

Конкатенация (сцепление) $L_1 \hat{} L_2$ содержит все слова, удовлетворяющие форме vw, где $v \in L_1$, а $w \in L_2$.

Итерацию[1] $(L)^*$ языка L образуют все слова, которые можно разбить на несколько подряд идущих слов из L:

$$(L)^* = \{\varepsilon\} \cup \{w \mid (\exists n \geqslant 0)(w = w_1 w_2 \dots w_n),\ w_i \in L, i \in 1 : n\}.$$

Ее можно представить с помощью степеней:

$$(L)^* = \bigcup_{i=0}^{\infty} L^i.$$

Заметим, что $(L)^*$ включает и пустое слово ϵ, так как $n = 0$ допустимо по условию.

Введем некоторые соглашения. При записи регулярных выражений будем опускать знак конкатенации $\hat{}$. Расставим приоритеты операций по уменьшению: итерация, конкатенация, объединение. Это позволит опустить многие скобки.

[1] Иногда ее называют замыканием Клини.

Пример 3.207. Выражение $(((1\hat{}0)\hat{}((1)^* + 0))$ можно записать как $10(1^* + 0)$.

Дадим строгое определение регулярного выражения. Регулярное выражение над алфавитом Σ определяется рекурсивно следующим образом.

Определение 3.142.

1) Пустой символ ϵ является регулярным выражением;

2) для любого $a \in \Sigma$, a является регулярным выражением;

3) если r и p являются регулярными выражениями, то $(r+p), (rp)$ и r^* тоже являются регулярными выражениями.

Определение 3.143. Два регулярных выражения r и p называются *эквивалентными*, если совпадают представляемые ими языки, т. е. $L_r = L_p$. В этом случае будем писать $r = p$.

Теорема 3.54. Справедливы следующие свойства регулярных операций:

1) $r + p = p + r$ (коммутативность объединения);

2) $(r + p) + q = r + (p + q)$ (ассоциативность объединения);

3) $(rp)q = r(pq)$ (ассоциативность конкатенации);

4) $(r^*)^* = r^*$ (идемпотентность итерации);

5) $(r + p)q = rq + pq$ (дистрибутивность).

Доказательство. Перечисленные свойства очевидным образом следуют из введенных выше определений. ■

Следствие 3.11. Класс регулярных языков замкнут относительно операций конкатенации и итерации.

Рассмотрим примеры регулярных выражений и представляемых ими языков.

1. Регулярное выражение $(0 + 1)^*$ представляет множество всех слов в алфавите $\{0, 1\}$, включая пустое слово.

2. Множество всех слов в алфавите $\{0, 1\}$, содержащих ровно одно вхождение 1, соответствует регулярному выражению 0^*10^*.

3. Регулярное выражение $11(0 + 1)^*001$ представляет язык, состоящий из всех слов в алфавите $\{0, 1\}$, которые начинаются на 11, а заканчиваются на 001.

Теорема 3.55. Для каждого регулярного выражения r можно эффективно построить такой недетерминированный конечный автомат A, который распознает язык, задаваемый r, т. е. $L_A = L_r$.

Доказательство этой теоремы достаточно техническое, построение автомата A по выражению r проводится индукцией по длине регулярного выражения r.

Следствие 3.12. Для каждого регулярного выражения можно эффективно построить детерминированный конечный автомат, который распознает язык, представляемый этим выражением.

Рассмотрим алгоритм построения недерминированного конечного автомата на основе регулярного выражения. Используем автоматы с эпсилон-переходами, рассмотренными в разделе 3.7.11.

Будем строить недерминированный конечный автомат (НКА) как композицию таких автоматов для более простых выражений исходя из определения 3.142. Комбинирование выполняется по правилам, представленным ниже.

1 **Правило 1.** Для ϵ строим НКА, i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.69a) распознает язык $L = \{\epsilon\}$.

2 **Правило 2.** Для $a \in \Sigma$ строим НКА, i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.69b) распознает язык $L = \{a\}$.

(a) *(b)*

Рис. 3.69.

3 **Правило 3.** Пусть A_s и A_t представляют собой НКА для регулярных выражений s и t. Для регулярного выражения $s + t$ строим НКА A_{s+t}, i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.70a) распознает язык $L_s \cup L_t$.

4 **Правило 4.** Для регулярного выражения st строим НКА A_{st}. Здесь начальное состояние НКА A_s становится новым начальным состоянием A_{st}, заключительное состояние A_t становится заключительным состоянием A_{st}. А конечное состояние A_s и начальное состояние A_t объединяются. Этот НКА (рисунок 3.70b) распознает язык $L_s L_t$.

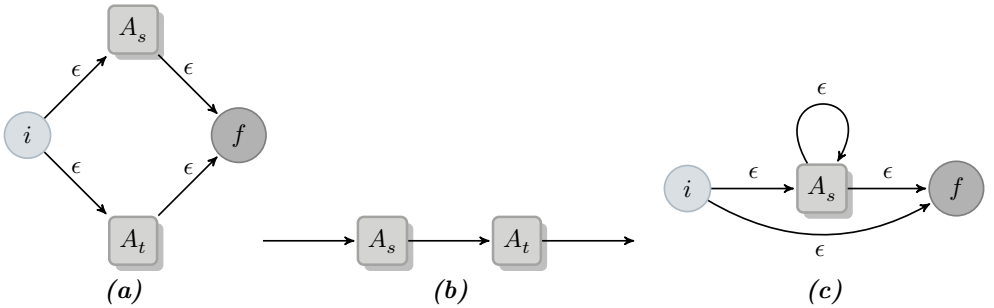


Рис. 3.70.

5 **Правило 5.** Для регулярного выражения s^* строим НКА A_{s^*}, i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.70c) распознает язык $L\,(s^*)$.

Пример 3.208. Используя перечисленные правила, построим НКА по регулярному выражению $r = (ab)^*a^*ba$.

Используя правила 2, 4 и 5, строим НКА A_r (рисунок 3.71) для регулярного выражения $r = (ab)^*$.

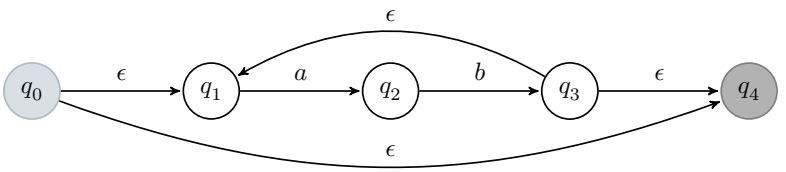


Рис. 3.71.

Используя правила 2 и 5, строим НКА A_s (рисунок 3.72) для регулярного выражения $s = a^*$.

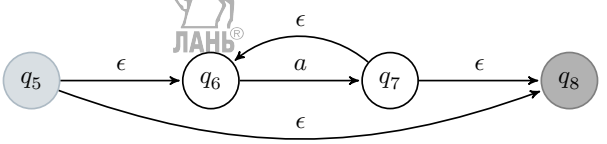


Рис. 3.72.

Используя правила 2 и 4, строим НКА A_t (рисунок 3.73) для регулярного выражения $t = ba$.

Соединяем построенные автоматы A_r, A_s и A_t (рисунок 3.74).

Пример 3.209. Построим детерминированный конечный автомат для регулярного выражения $(a + b)^*a(b + a)$.

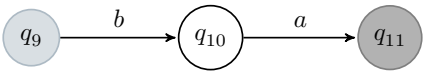


Рис. 3.73.

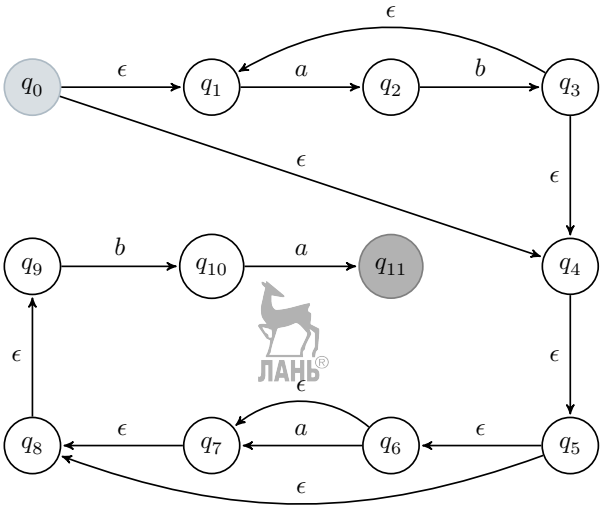
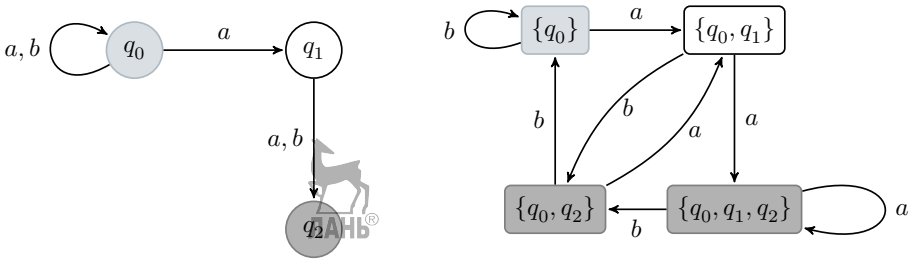


Рис. 3.74.

Построим сначала недерминированный автомат (рисунок 3.75a), затем детерминизируем его. В данном случае при построении можно обойтись без эпсилон-переходов.



(a) ***(b)***

Рис. 3.75.

Применяя алгоритм 3.25 детерминизации вытягиванием, получаем:

$\delta_1(\{q_0\}, a) = \{q_0, q_1\}$; $\delta_1(\{q_0\}, b) = \{q_0\}$;
$\delta_1(\{q_0, q_1\}, a) = \{q_0, q_1, q_2\}$; $\delta_1(\{q_0, q_1\}, b) = \{q_0, q_2\}$;
$\delta_1(\{q_0, q_2\}, a) = \{q_0, q_1\}$; $\delta_1(\{q_0, q_2\}, b) = \{q_0\}$;
$\delta_1(\{q_0, q_1, q_2\}, a) = \{q_0, q_1, q_2\}$; $\delta_1(\{q_0, q_1, q_2\}, b) = \{q_0, q_3\}$;

$$\delta_1(\{q_0, q_3\}, a) = \{q_0, q_1\}; \qquad \delta_1(\{q_0, q_3\}, b) = \{q_0\}.$$

Полученный детерминированный автомат представлен на рисунке 3.75b.

Задачи и вопросы

1. Дайте определение автоматов Мили и Мура. В чем их различие?

2. Как преобразовать автомат Мура в эквивалентный ему автомат Мили?

3. Дайте определение недетерминированного конечного автомата-распознавателя.

4. Опишите алгоритм построения недетерминированного конечного автомата по автоматной грамматике.

5. Постройте конечный автомат Мили, который для заданных двух целых двоичных чисел N_1 и N_2 одинаковой длины (меньшее может быть дополнено слева нулями) выдает $\max(X_1, X_2)$. Числа вводятся со старших разрядов, вход автомата — пары разрядов исходных чисел.

6. Приведите пример конечного автомата с ϵ-переходами. Для чего нужны такие автоматы?

7. Постройте конечный автомат, распознающий язык L в алфавите $\{a, b, c\}$, где слово принадлежит языку L, если в нем не встречается буква a.

8. Постройте конечный автомат, распознающий язык

$$L = \{ucv \mid u \in \{a, b\}^*,\ v \in \{a, b\}^*\}.$$

9. Постройте конечный автомат с входным алфавитом $\{0, 1\}$, допускающий цепочки:

а) число единиц четное, а нулей — нечетное;

б) между вхождениями единиц четное число нулей;

в) за каждым вхождением пары 11 следует 0;

г) каждый третий символ — единица;

д) имеется по крайней мере одна единица;

е) начинаются с 0 и заканчиваются на 1.

10. Постройте конечный автомат с входным алфавитом $\Sigma = \{a, b, c\}$, допускающий все цепочки: