

# **Конечные автоматы и регулярные грамматики**

## **Лекция 3.**

# **План лекции**

**§ 3.1. Конечный автомат**

**§ 3.2. Отношения эквивалентности и  
конечные автоматы**

**§ 3.3. Недетерминированные конечные  
автоматы**

**§ 3.4. Конечные автоматы и языки типа 3**

**§ 3.5. Свойства языков типа 3**

**§ 3.6. Алгоритмически разрешимые  
проблемы, касающиеся конечных  
автоматов**

## § 3.1. Конечный автомат

**Определение.** Конечным автоматом называется формальная система

$$M = (Q, \Sigma, \delta, q_0, F),$$

где  $Q$  — конечное непустое множество состояний;

$\Sigma$  — конечный входной алфавит;

$\delta$  — отображение типа  $Q \times \Sigma \rightarrow Q$ ;

$q_0 \in Q$  — начальное состояние;

$F \subseteq Q$  — множество конечных состояний.

Запись  $\delta(q, a) = p$ , где  $q, p \in Q$  и  $a \in \Sigma$ , означает, что конечный автомат  $M$  в состоянии  $q$ , сканируя входной символ  $a$ , продвигает свою входную головку на одну ячейку вправо и переходит в состояние  $p$ .

Область определения отображения  $\delta$  можно расширить до  $Q \times \Sigma^*$  следующим образом:

$$\delta'(q, \varepsilon) = q, \delta'(q, xa) = \delta(\delta'(q, x), a)$$

для любого  $x \in \Sigma^*$  и  $a \in \Sigma$ .

Таким образом, запись  $\delta'(q, x) = p$  означает, что fa  $M$ , начиная в состоянии  $q \in Q$  чтение цепочки  $x \in \Sigma^*$ , записанной на входной ленте, оказывается в состоянии  $p \in Q$ , когда его входная головка продвинется правее цепочки  $x$ .

Далее мы будем использовать одно и то же обозначение  $\delta$  для обоих отображений, так как это не приведёт к путанице.

Определенная таким образом модель конечного автомата называется *детерминированной* (см. рис. 3.0).

Для обозначения детерминированного автомата часто используют аббревиатуру dfa (**d**eterministic **f**inite **a**utomaton).

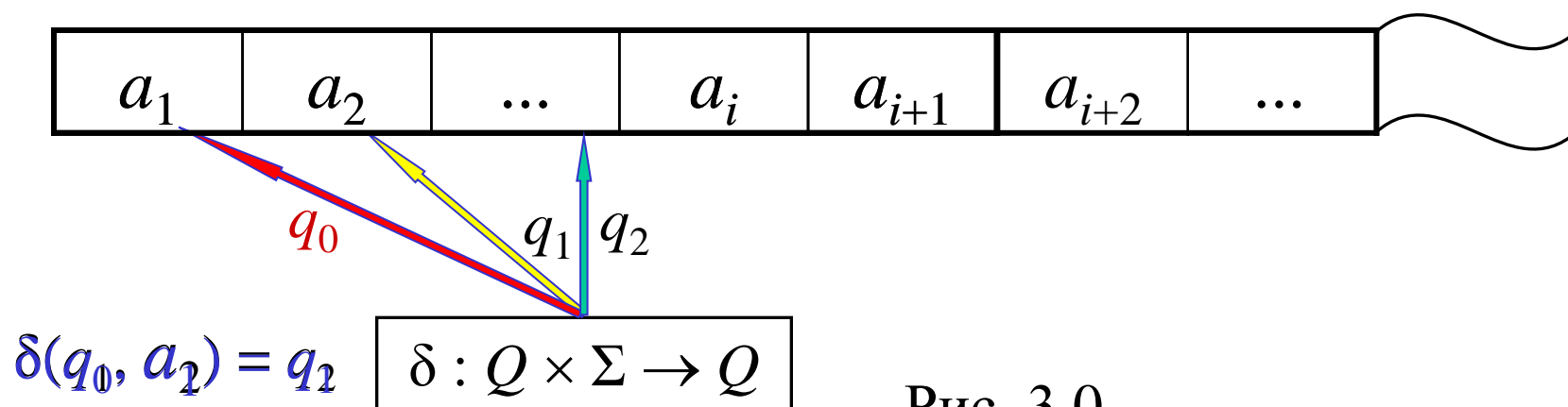


Рис. 3.0.

**Определение.** Цепочка  $x \in \Sigma^*$  принимается конечным автоматом  $M$ , если  $\delta(q_0, x) = p$  для некоторого  $p \in F$ .

Множество всех цепочек  $x \in \Sigma^*$ , принимаемых конечным автоматом  $M$ , называется *языком, распознаваемым конечным автоматом  $M$* , и обозначается как  $T(M)$ , т. е.

$$T(M) = \{x \in \Sigma^* \mid \delta(q_0, x) = p \text{ при } p \in F\}.$$

Любое множество цепочек, принимаемых конечным автоматом, называется *регулярным*.



**Пример 3.1.** Пусть задан dfa

$$M = (Q, \Sigma, \delta, q_0, F),$$

где  $Q = \{q_0, q_1, q_2, q_3\}$ ,  $\Sigma = \{0, 1\}$ ,  $F = \{q_0\}$ ,

$$\delta(q_0, 0) = q_2, \quad \delta(q_0, 1) = q_1,$$

$$\delta(q_1, 0) = q_3, \quad \delta(q_1, 1) = q_0,$$

$$\delta(q_2, 0) = q_0, \quad \delta(q_2, 1) = q_3,$$

$$\delta(q_3, 0) = q_1, \quad \delta(q_3, 1) = q_2.$$

Рассмотрим *диаграмму переходов* этого конечного автомата.

# Диаграмма переходов

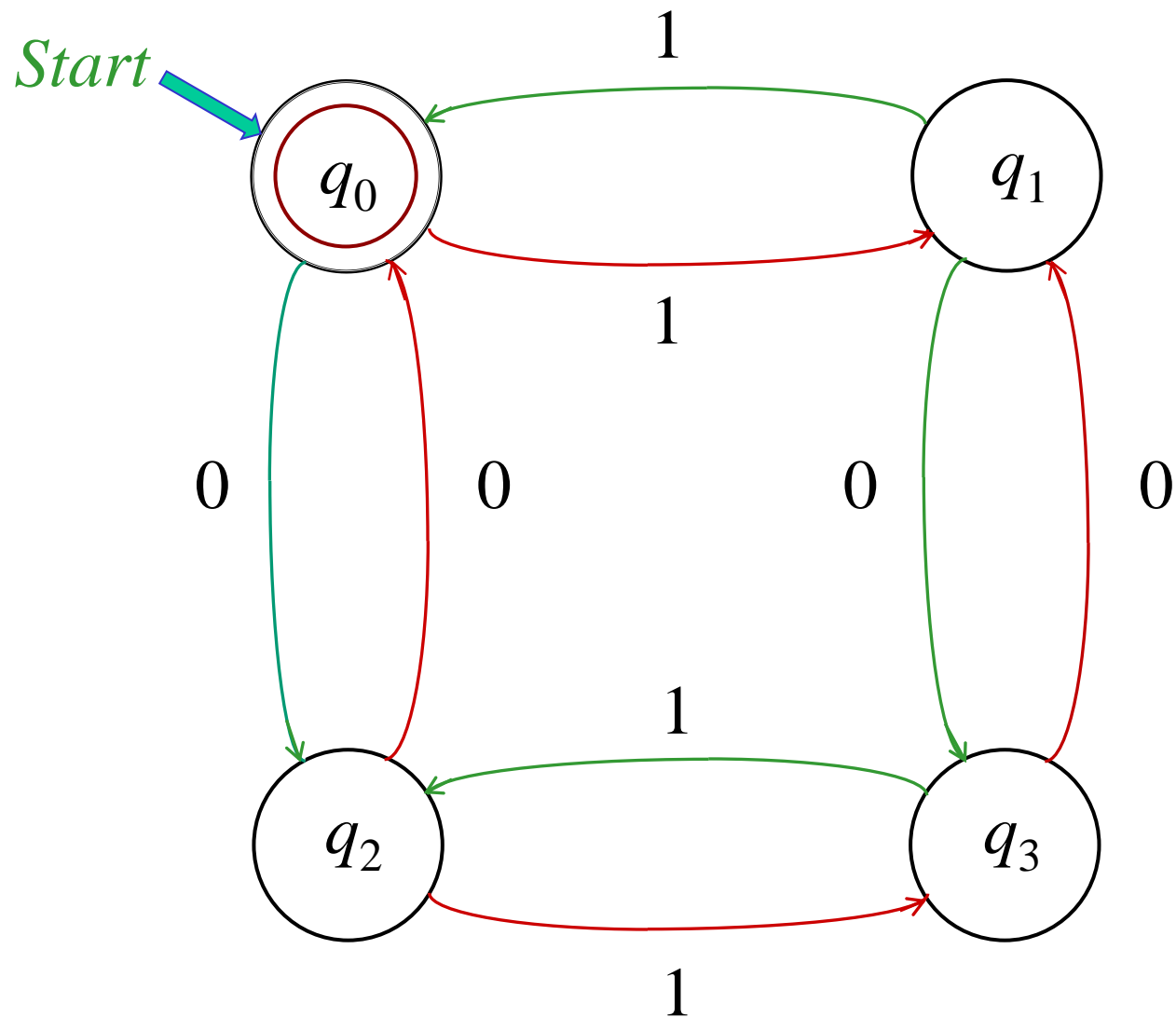


Рис. 3.1.

Диаграмма переходов конечного автомата состоит из узлов, представляющих состояния, и ориентированных дуг, определяющих возможные переходы, которые зависят от входных символов.

Так, если  $\delta(q, a) = p$ , то из узла, представляющего состояние  $q$ , в узел, представляющий состояние  $p$ , проводится дуга, помеченная входным символом  $a$ .

Двойным кружком выделено единственное в данном примере *конечное* состояние, которое является одновременно и *начальным*.

Предположим, что на входе автомата  $M$  находится цепочка **11**0101. Поскольку

$$\delta(q_0, 1) = q_1, \text{ а } \delta(q_1, 1) = q_0 \text{ и } q_0 \in F,$$

то цепочка **11** находится в языке, распознаваемом данным конечным автоматом  $M$ .

Но мы интересуемся всей входной цепочкой. Сканируя остаток 0101 входной цепочки, автомат переходит последовательно в состояния  $q_2, q_3, q_1, q_0$ .

Поэтому  $\delta(q_0, 110101) = q_0$ , и потому цепочка 110101 тоже находится в  $T(M)$ .

Очевидно, что  $T(M)$  есть множество всех цепочек из  $\{0, 1\}^*$ , содержащих чётное число нулей и чётное число единиц.

***Вопрос:***  $\varepsilon \in T(M)$ ?


## § 3.2. Отношения эквивалентности и конечные автоматы

Пусть  $M$  — dfa. Определим отношение  $R$  на множестве  $\Sigma^*$  следующим образом:

$(x, y) \in R$  тогда и только тогда, когда  
$$\delta(q_0, x) = \delta(q_0, y).$$

Отношение  $R$  *рефлексивно, симметрично*  
и *транзитивно*, т. е.  $R$  — *отношение эквивалентности*.

Транзитивность:  $(x, y) \in R, (y, z) \in R \Rightarrow (x, z) \in R$ :

$$\underline{\delta(q_0, x) = \delta(q_0, y) = p}; \quad \delta(q_0, y) = \underline{\delta(q_0, z) = q = p} \Rightarrow (x, z) \in R$$


Автомат  $M$  из примера 3.1 индуцирует отношение эквивалентности  $R$ , которое делит множество  $\{0, 1\}^*$  на четыре класса эквивалентности, соответствующие четырём состояниям этого автомата.

Кроме того, если  $xRu$ , то для всех  $z \in \{0, 1\}^*$  имеет место  $xzRuz$ , поскольку

$$\begin{aligned}\delta(q_0, xz) &= \delta(\delta(q_0, x), z) = \delta(\delta(q_0, u), z) = \\ &= \delta(q_0, uz).\end{aligned}$$

Такое отношение эквивалентности называется *право-инвариантным*.

**Теорема 3.1.** *Следующие три высказывания эквивалентны:*

- 1) *Язык  $L \subseteq \Sigma^*$  распознаётся некоторым fa.*
- 2) *Язык  $L$  есть объединение некоторых классов эквивалентности право-инвариантного отношения эквивалентности конечного индекса;*
- 3) *Пусть отношение эквивалентности  $R$  определяется через язык  $L$  следующим образом:  
 $xRu$  тогда и только тогда, когда для всех  $z \in \Sigma^*$  имеет место принадлежность  $xz \in L$  точно тогда, когда  $uz \in L$ .*

*Отношение  $R$  имеет конечный индекс.*



Доказательство.

**1→2.** Предположим, что язык  $L$  принимается некоторым конечным автоматом  $M' = (Q', \Sigma', \delta', q'_0, F')$ .

Пусть  $R'$  — отношение эквивалентности, определяемое следующим образом:

$xR'y$  тогда и только тогда, когда

$$\delta'(q'_0, x) = \delta'(q'_0, y).$$

Отношение  $R'$  — право-инвариантно, поскольку, если  $x R' y$ , т.  $\delta'(q'_0, x) = \delta'(q'_0, y)$ , то для любого  $z \in \Sigma^*$  имеем

$$\begin{aligned}\delta'(q'_0, xz) &= \delta'(\delta'(q'_0, x), z) = \delta'(\delta'(q'_0, y), z) = \\ &= \delta'(q'_0, yz), \text{ т. е. } xz R' yz.\end{aligned}$$

Индекс отношения  $R'$  конечен, поскольку (самое большее) он равен числу состояний fa  $M'$ .

Кроме того, язык  $L$  есть объединение лишь тех классов эквивалентности, которые включают цепочки  $x \in \Sigma^*$ , переводящие автомат  $M'$  в конечные состояния:

$$\delta'(q'_0, x) = p, \text{ где } p \in F'.$$

Итак, из высказывания (1) следует (2).

**2→3.** Пусть язык  $L$  есть объединение некоторых классов эквивалентности в отношении  $R'$ , которое является *правоинвариантным* и имеет *конечный индекс*.

Покажем сначала, что каждый класс эквивалентности  $R'$  целиком содержится в некотором классе эквивалентности  $R$ , определённом в высказывании (3).

Пусть  $xR'y$ . Так как отношение  $R'$  правоинвариантно, то для любого  $z \in \Sigma^*$  имеет место  $xzR'yz$  и, таким образом,  $xz \in L$  точно тогда, когда  $yz \in L$ , т. е.  $x R y$ .

Следовательно,  $R' \subseteq R$ , и потому

$$[x]_{R'} \subseteq [x]_R.$$

Это и значит, что любой класс эквивалентности отношения эквивалентности  $R'$  содержится в некотором классе эквивалентности отношения эквивалентности  $R$ .

Поскольку отношения  $R$  и  $R'$  разбивают на классы эквивалентности *одно* и то же множество цепочек  $\Sigma^*$ , то индекс отношения эквивалентности  $R$  не может быть больше индекса отношения эквивалентности  $R'$  (чем крупнее классы, тем их меньше).

Согласно высказыванию (2) индекс отношения эквивалентности  $R'$  *конечен*. Следовательно, индекс отношения эквивалентности  $R$ , тем более, *конечен*.

Итак, из высказывания (2) следует (3).

**3→1.** Пусть  $xRy$ . Тогда для любых  $w, z \in \Sigma^*$  цепочка  $xwz \in L$  в точности тогда, когда цепочка  $ywz \in L$ . Следовательно,  $xwRyw$ , и потому  $R$  — *правоинвариантно*.

Построим конечный автомат

$$M = (Q, \Sigma, \delta, q_0, F),$$

где в качестве  $Q$  возьмем конечное множество классов эквивалентности  $R$ , т. е.

$$Q = \{[x]_R \mid x \in \Sigma^*\};$$

ПОЛОЖИМ

$$\delta([x]_R, a) = [xa]_R,$$

и это определение непротиворечиво, так как  $R$  — право-инвариантно;

ПОЛОЖИМ

$$q_0 = [\varepsilon]_R \text{ и } F = \{[x]_R \mid x \in L\}.$$

Очевидно, что конечный автомат  $M$  *распознаёт язык  $L$* , поскольку

$$\delta(q_0, x) = \delta([\varepsilon]_R, x) = [x]_R,$$

и, таким образом,  $x \in T(M)$  *тогда и только тогда, когда  $[x]_R \in F$* .

Итак, из высказывания (3) следует (1).



**Теорема 3.2.** *Конечный автомат с минимальным числом состояний, распознающий язык  $L$ , единствен с точностью до изоморфизма (т. е. переименования состояний), и есть  $f\alpha M$  из теоремы 3.1 (см.  $3 \rightarrow 1$ ).*

Доказательство. При доказательстве теоремы 3.1 мы установили, что любой конечный автомат  $M' = (Q', \Sigma', \delta', q'_0, F')$ , распознающий язык  $L$ , индуцирует отношение эквивалентности  $R'$ , индекс которого не меньше индекса отношения эквивалентности  $R$ , определённого при формулировке утверждения  $3 \rightarrow 1$ . Поэтому число состояний  $\text{fa } M'$  больше или равно числу состояний  $\text{fa } M$ , построенного в третьей части доказательства теоремы 3.2.

Если  $M'$  — fa с *минимальным* числом состояний, то число его состояний равно числу состояний fa  $M$  и между состояниями  $M'$  и  $M$  можно установить однозначное соответствие.

Действительно, пусть  $q' \in Q'$ .

Должна существовать некоторая цепочка  $x \in \Sigma^*$ , такая, что  $\delta'(q'_0, x) = q'$ , ибо в противном случае состояние  $q'$  без какого-нибудь ущерба для языка, распознаваемого этим автоматом, можно было бы исключить из множества состояний  $Q'$  как недостижимое.

Отбросив такое недостижимое состояние, мы получили бы автомат с меньшим числом состояний, который распознавал бы всё тот же самый язык. Но это противоречило бы предположению, что  $M'$  является конечным автоматом с *минимальным* числом состояний.

Пусть  $q' \in Q'$  и  $q' = \delta'(q'_0, x)$ .

Сопоставим с  $q'$  состояние  $q \in Q$ , достижимое автоматом  $M$  по прочтении той же цепочки  $x$ :

$$q = \delta(q_0, x) = \delta([\varepsilon]_R, x) = [x]_R.$$

Это сопоставление является непротиворечивым, так как отображение  $R$  право-инвариантное.

Действительно, если  $q', p' \in Q'$  и  $q' = p'$ , причём  $q' = \delta'(q'_0, x)$ , а  $p' = \delta'(q'_0, y)$ , то их образы есть соответственно

$$q = \delta(q_0, x) = \delta([\varepsilon]_R, x) = [x]_R \text{ и} \\ p = \delta(q_0, y) = \delta([\varepsilon]_R, y) = [y]_R.$$

Учитывая, что  $x$  и  $y$  принадлежат одному и тому же классу эквивалентности отношения  $R'$ , и что  $R' \subseteq R$ , заключаем, что  $x$  и  $y$  также находятся в одном и том же классе эквивалентности отношения  $R$ , т. е.

$$[x]_R = [y]_R, \text{ и потому } q = p.$$

Другими словами, если прообразы состояний ( $q'$  и  $p'$ ) равны, то равны и их образы ( $q$  и  $p$ ).

$$\begin{array}{ccc}
 M' : q' = \delta'(q'_0, x) & \xrightarrow{a} & p' = \delta'(q', a) = \\
 \downarrow x & & \downarrow xa \\
 & & = \delta'(\delta'(q'_0, x), a) = \\
 & & = \delta'(q'_0, xa) \\
 M : q = \delta([\varepsilon]_R, x) = [x]_R & \xrightarrow{a} & p = \delta([\varepsilon]_R, xa) = [xa]_R
 \end{array}$$

Рис. 3.2.



Кроме того, если  $\text{fa } M'$  совершает переход из состояния  $q'$  в состояние  $p'$ , прочитав символ  $a \in \Sigma$ , то  $\text{fa } M$  переходит из состояния  $q$ , являющегося образом  $q'$ , в состояние  $p$ , являющееся образом  $p'$ , прочитав тот же самый символ  $a \in \Sigma$ , так как  $\delta([x]_R, a) = [xa]_R$  (рис. 3.2).

Это и значит, что мы имеем *изоморфное* отображение множества состояний  $\text{fa } M'$  на множество состояний  $\text{fa } M$ .

## § 3.3. Недетерминированные конечные автоматы

Теперь мы введём понятие недетерминированного конечного автомата (ndfa — **n**ondeterministic **f**inite **a**utomaton). От своего детерминированного аналога, определённого ранее (см. первое определение), он отличается только типом управляющего отображения  $\delta$ .

Когда важно подчеркнуть, что имеется в виду именно детерминированная модель конечного автомата, мы будем использовать обозначение dfa, а когда это не важно, будем использовать нейтральное fa.

Мы увидим, что любое множество, распознаваемое *недетерминированным* конечным автоматом, может также распознаваться *детерминированным* конечным автоматом.

Недетерминированный конечный автомат является полезным понятием при доказательстве теорем.

Кроме того, с этого простейшего понятия легче начать знакомство с недетерминированными устройствами, которые не эквивалентны своим детерминированным аналогам.

**Определение.** *Недетерминированным конечным автоматом называется формальная система*

$$M = (Q, \Sigma, \delta, q_0, F),$$

где  $Q$  — конечное непустое множество состояний;

$\Sigma$  — входной алфавит;

$\delta$  — отображение типа  $Q \times \Sigma \rightarrow 2^Q$ ,

$q_0 \in Q$  — начальное состояние;

$F \subseteq Q$  — множество конечных состояний.

Существенная разница между детерминированной и недетерминированной моделями конечного автомата состоит в том, что значение  $\delta(q, a)$  является (возможно пустым) множеством состояний, а не одним состоянием.

Запись  $\delta(q, a) = \{p_1, p_2, \dots, p_k\}$  означает, что недетерминированный конечный автомат  $M$  в состоянии  $q$ , сканируя символ  $a$  на входной ленте, продвигает входную головку вправо к следующей ячейке и выбирает любое из состояний  $p_1, p_2, \dots, p_k$  в качестве следующего.

Область определения  $\delta$  может быть расширена на  $Q \times \Sigma^*$  следующим образом:

$$\begin{aligned}\delta(q, \varepsilon) &= \{q\}, \\ \delta(q, xa) &= \bigcup_{p \in \delta(q, x)} \delta(p, a)\end{aligned}$$

для каждого  $x \in \Sigma^*$  и  $a \in \Sigma$ .

Область определения  $\delta$  может быть расширена далее до  $2^Q \times \Sigma^*$  следующим образом:

$$\delta(\{p_1, p_2, \dots, p_k\}, x) = \bigcup_{i=1}^k \delta(p_i, x).$$

**Определение.** Цепочка  $x \in \Sigma^*$  принимается недетерминированным конечным автоматом  $M$ , если существует состояние  $p$ , такое, что  $p \in F$  и  $p \in \delta(q_0, x)$ . Множество всех цепочек  $x$ , принимаемых ndfa  $M$ , обозначается  $T(M)$ .

**Замечание 3.1.** Напомним, что  $2^Q$ , где  $Q$  — любое множество, обозначает *степенное множество* или множество всех подмножеств множества  $Q$ .



**Пример 3.2.** Рассмотрим ndfa, который распознает множество  $\{0,1\}^*\{00,11\}\{0,1\}^*$ :

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2, q_4\})$ ,  
где

$$\begin{array}{ll} \delta(q_0, 0) = \{q_0, q_3\}, & \delta(q_0, 1) = \{q_0, q_1\}, \\ \delta(q_1, 0) = \emptyset, & \delta(q_1, 1) = \{q_2\}, \\ \delta(q_2, 0) = \{q_2\}, & \delta(q_2, 1) = \{q_2\}, \\ \delta(q_3, 0) = \{q_4\}, & \delta(q_3, 1) = \emptyset, \\ \delta(q_4, 0) = \{q_4\}, & \delta(q_4, 1) = \{q_4\}. \end{array}$$

На рис. 3.3 приведена диаграмма переходов этого автомата. Фактически он принимает любые цепочки, составленные из нулей и единиц, в которых встречаются два подряд идущих нуля или единицы.

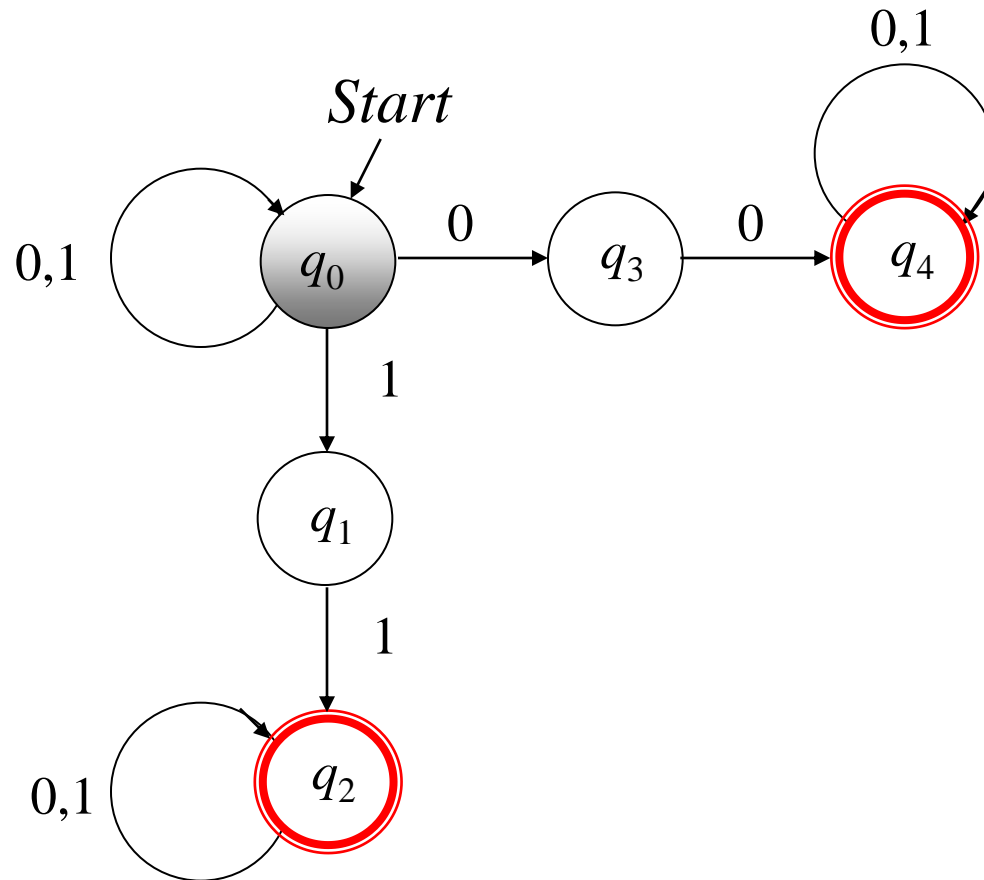


Рис. 3.3. Диаграмма переходов.

**Теорема 3.3.** Пусть  $L$  — язык, распознаваемый недетерминированным конечным автоматом. Тогда существует детерминированный конечный автомат, который распознаёт  $L$ .

Доказательство. Пусть  $M = (Q, \Sigma, \delta, q_0, F)$  — ndfa и  $L = T(M)$ .

Определим dfa  $M' = (Q', \Sigma', \delta', q'_0, F')$  следующим образом.

Положим  $Q' = \{[s] \mid s \in 2^Q\}$ .

Состояние из множества  $Q'$  будем представлять в виде  $[q_1, q_2, \dots, q_i]$ , где  $q_i \in Q$  ( $i > 0$ ).

Обозначение  $\varphi$  будем использовать в случае, когда  $i = 0$ , то есть  $s = \emptyset$ .

Начальное состояние  $q_0' = [q_0]$ .

Таким образом, dfa  $M'$  будет хранить след всех состояний, в которых ndfa  $M$  мог бы быть в любой данный момент.

Пусть  $F'$  — множество всех состояний из  $Q'$ , содержащих хотя бы одно состояние из множества конечных состояний  $F$ .

Входной алфавит  $\Sigma' = \Sigma$  — такой же, как в данном ndfa  $M$ .

## Эквивалентность недетерминированных и детерминированных конечных автоматов

---

Определим

$$\delta'([q_1, q_2, \dots, q_i], a) = [p_1, p_2, \dots, p_j]$$

тогда и только тогда, когда

$$\delta(\{q_1, q_2, \dots, q_i\}, a) = \{p_1, p_2, \dots, p_j\}.$$

Индукцией по длине  $l$  входной цепочки  $x \in \Sigma^*$  легко показать, что

$$\delta'(q_0, x) = [q_1, q_2, \dots, q_i]$$

тогда и только тогда, когда

$$\delta(q_0, x) = \{q_1, q_2, \dots, q_i\}.$$

База. Пусть  $l = 0$ . Утверждение выполняется,  
ибо

$$\delta'(q_0', \varepsilon) = q_0' = [q_0] \text{ и } \delta(q_0, \varepsilon) = \{q_0\}.$$

Индукционная гипотеза. Предположим, что  
утверждение выполняется для всех  $l \leq n$   
( $n \geq 0$ ).

Индукционный переход. Докажем, что тогда  
утверждение выполняется и для  $l = n + 1$ .



Пусть  $x = za$ , где  $z \in \Sigma^*$ ,  $|z| = n$ ,  $a \in \Sigma$ .

Тогда

$$\delta'(q_0', x) = \delta'(q_0', za) = \delta'(\delta'(q_0', z), a).$$

По индукционному предположению

$$\left. \begin{array}{l} \delta'(q_0', z) = [q_1, q_2, \dots, q_i] \\ \text{тогда и только тогда, когда} \\ \delta(q_0, z) = \{q_1, q_2, \dots, q_i\}. \end{array} \right\} \quad (1)$$

В то же время по построению

$$\left. \begin{array}{l} \delta'([q_1, q_2, \dots, q_i], a) = [p_1, p_2, \dots, p_j] \\ \text{тогда и только тогда, когда} \\ \delta(\{q_1, q_2, \dots, q_i\}, a) = \{p_1, p_2, \dots, p_j\}. \end{array} \right\} \quad (2)$$

Таким образом, учитывая (1) и (2), имеем

$$\begin{aligned}\delta'(q_0', x) &= \delta'(q_0', za) = \\ &= \delta'([q_1, q_2, \dots, q_i], a) = [p_1, p_2, \dots, p_j]\end{aligned}$$

тогда и только тогда, когда

$$\begin{aligned}\delta(q_0, x) &= \delta(q_0, za) = \\ &= \delta(\{q_1, q_2, \dots, q_i\}, a) = \{p_1, p_2, \dots, p_j\}.\end{aligned}$$

Чтобы закончить доказательство, остается добавить, что

$$\delta'(q'_0, x) \in F'$$

точно тогда, когда  $\delta(q_0, x)$  содержит состояние из множества конечных состояний  $F$ .

Следовательно,  $T(M) = T(M')$ .

Что и требовалось доказать.

**Пример 3.3.** Пусть

$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$  — ndfa, где

$$\delta(q_0, 0) = \{q_0, q_1\}, \quad \delta(q_0, 1) = \{q_1\},$$

$$\delta(q_1, 0) = \emptyset, \quad \delta(q_1, 1) = \{q_0, q_1\}.$$

См. рис. 2.4 (а).

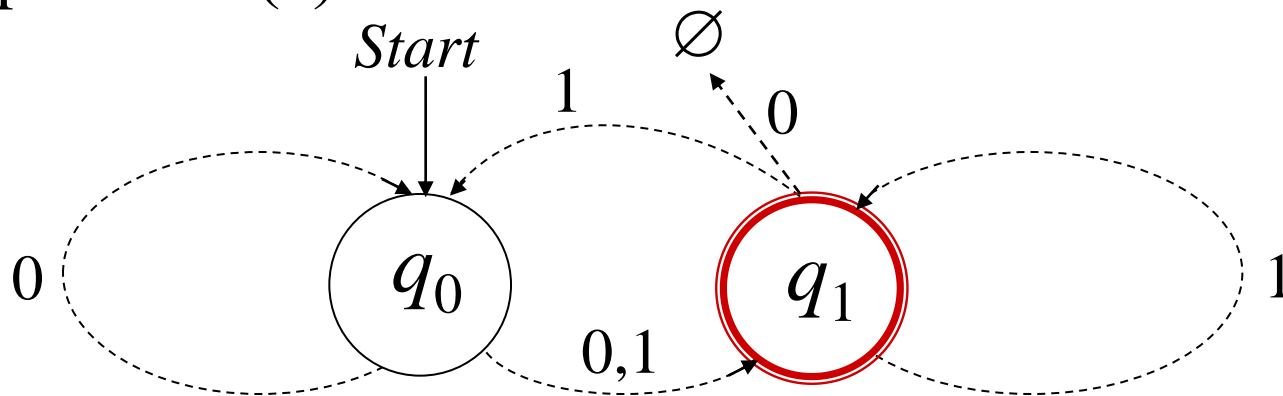


Рис. 3.4 (а). Недетерминированный автомат  $M$ .

### Пример 3.3.

---

Построим детерминированный конечный автомат, эквивалентный данному.

Положим  $M' = (Q', \{0, 1\}, \delta', q'_0, F')$ .

Согласно теореме 3.3 в качестве состояний детерминированного автомата следует взять все подмножества множества  $\{q_0, q_1\}$ , включая пустое:

### Пример 3.3.

---

$Q' = \{\varphi, [q_0], [q_1], [q_0, q_1]\},$   
причём  $q'_0 = [q_0].$

Конечные состояния автомата  $M'$  представлены теми подмножествами, которые содержат конечные состояния данного автомата (в нашем случае —  $q_1$ ), т. е.

$$F' = \{[q_1], [q_0, q_1]\}.$$

Наконец,

$$\begin{aligned}\delta'([q_0], 0) &= [q_0, q_1], & \delta'([q_0], 1) &= [q_1], \\ \delta'([q_1], 0) &= \varnothing, & \delta'([q_1], 1) &= [q_0, q_1], \\ \delta'([q_0, q_1], 0) &= [q_0, q_1], & \delta'([q_0, q_1], 1) &= [q_0, q_1], \\ \delta'(\varnothing, 0) &= \varnothing, & \delta'(\varnothing, 1) &= \varnothing.\end{aligned}$$

Поясним, что  $\delta'([q_0, q_1], 0) = [q_0, q_1]$ , так как  $\delta(q_0, 0) = \{q_0, q_1\}$ ,  $\delta(q_1, 0) = \varnothing$ , и  $\{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$ .

Аналогично,  $\delta'([q_0, q_1], 1) = [q_0, q_1]$ , ибо  $\delta(q_0, 1) = \{q_1\}$ ,  $\delta(q_1, 1) = \{q_0, q_1\}$  и  $\{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$ . См. [рис. 3.4 \(б\)](#).

### Пример 3.3.

---

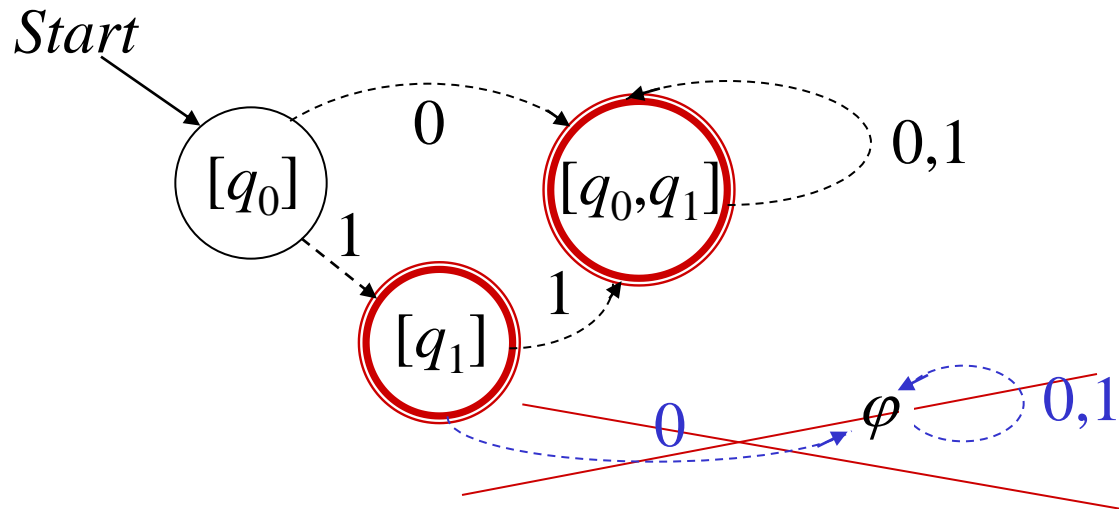


Рис. 3.4 (б). Детерминированный автомат  $M'$ .

Состояние  $\varphi$  можно удалить как бесполезное (*состояние ошибки*), ибо из него не достижимо никакое *конечное*.



## § 3.4. Конечные автоматы и языки типа 3

Теперь мы возвращаемся к связи языков, порождаемых грамматиками типа 3, с множествами, которые распознаются конечными автоматами.

Для удобства рассуждений введём понятие *конфигурации конечного автомата*, которое относится как dfa, так и к ndfa.

**Определение.** Пусть  $M = (Q, \Sigma, \delta, q_0, F)$  — конечный автомат.

*Конфигурацией* конечного автомата  $M$  назовём состояние управления  $q \in Q$  в паре с непрочитанной частью входной цепочки от текущего символа до конца входной цепочки  $x \in \Sigma^*$ :  $(q, x)$ .

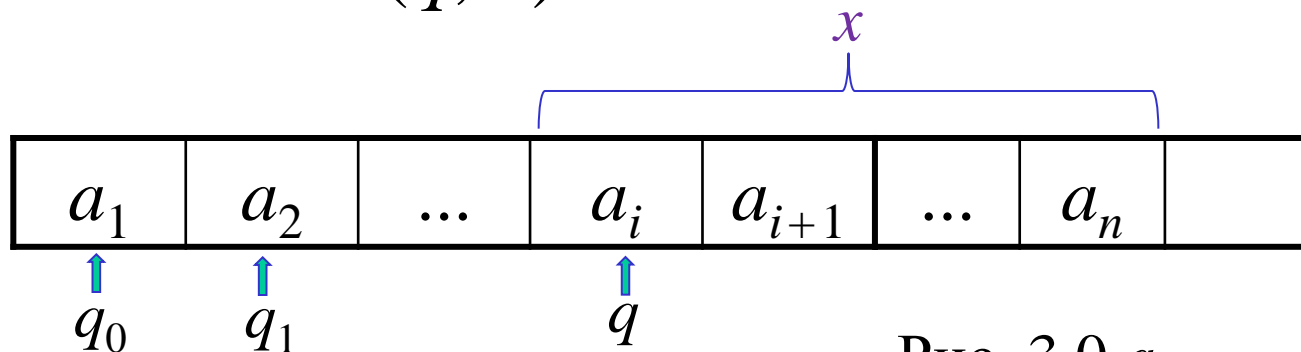


Рис. 3.0 а

Пусть  $(q, ax)$  — конфигурация fa  $M$ , где  $q \in Q$ ,  $a \in \Sigma$ ,  $x \in \Sigma^*$ , и пусть  $p = \delta(q, a)$  в случае, если  $M$  — dfa, или  $p \in \delta(q, a)$  в случае, когда  $M$  — ndfa.

Тогда fa  $M$  может перейти из конфигурации  $(q, ax)$  в конфигурацию  $(p, x)$ , и этот факт мы будем записывать как

$$(q, ax) \vdash (p, x).$$

Далее символом  $\vdash^*$  обозначается *рефлексивно-транзитивное* замыкание отношения непосредственного следования конфигураций  $\vdash$  на множестве конфигураций.

Запись

$$(q_0, x) \vdash^* (p, \varepsilon),$$

где  $p \in F$ , равнозначна записи  $x \in T(M)$ .

**Теорема 3.4.** Пусть  $G = (V_N, V_T, P, S)$  — грамматика типа 3. Тогда существует конечный автомат  $M = (Q, \Sigma, \delta, q_0, F)$ , такой, что  $T(M) = L(G)$ .

Доказательство. Построим ndfa  $M$ , о котором идёт речь. В качестве состояний возьмём нетерминалы грамматики и ещё одно дополнительное состояние  $A \notin V_N$ :

$Q = V_N \cup \{A\}$ , причём начальное состояние автомата  $M$  есть  $q_0 = S$ .

$$L = L(G) \implies L = T(M)$$


---

$$F = \begin{cases} \{S, A\}, & \text{если } S \rightarrow \varepsilon \in P ; \\ \{A\}, & \text{если } S \rightarrow \varepsilon \notin P. \end{cases}$$

Предполагается, что начальный нетерминал  $S$  не будет появляться в правых частях правил, если  $S \rightarrow \varepsilon \in P$

Включим  $A$  в  $\delta(B, a)$ , если  $B \rightarrow a \in P$ . Кроме того, в  $\delta(B, a)$  включим все  $C \in V_N$ , такие, что  $B \rightarrow aC \in P$ .

Положим  $\delta(A, a) = \emptyset$  для каждого  $a \in V_T$ .

$$L = L(G) \implies L = T(M)$$

---

Построенный автомат  $M$ , принимая цепочку  $x$ , моделирует её вывод в грамматике  $G$ .

Требуется показать, что  $T(M) = L(G)$ .

Если  $S \xRightarrow{*} x$ , то  $(q_0, x) \vdash^* (p, \varepsilon)$

---

I. Пусть  $x = a_1 a_2 \dots a_n$  и  $x \in L(G)$ ,  $n \geq 1$ .

Тогда существует вывод вида

$$S \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \dots \Rightarrow a_1 a_2 \dots a_{n-1} A_{n-1} \Rightarrow \\ \Rightarrow a_1 a_2 \dots a_{n-1} a_n, \text{ где } A_1, \dots, A_{n-1} \in V_N.$$

Очевидно, что в нём используются следующие правила:

$$S \rightarrow a_1 A_1, A_1 \rightarrow a_2 A_2, \dots, A_{n-2} \rightarrow a_{n-1} A_{n-1}, \\ A_{n-1} \rightarrow a_n \in P.$$

По построению  $\delta$  имеем:

$$A_1 \in \delta(S, a_1), A_2 \in \delta(A_1, a_2), \dots, \\ A_{n-1} \in \delta(A_{n-2}, a_{n-1}), A \in \delta(A_{n-1}, a_n).$$



Если  $S \xRightarrow{*} x$ , то  $(q_0, x) \vdash^* (p, \varepsilon)$

---

Следовательно, существует последовательность конфигураций

$$(S, a_1 a_2 \dots a_n) \vdash (A_1, a_2 \dots a_n) \vdash \dots \\ \dots \vdash (A_{n-1}, a_n) \vdash (A, \varepsilon),$$

причём  $A \in F$  и потому  $x \in T(M)$ .

Если же  $x = \varepsilon$ , то есть в данном выводе используется правило  $S \rightarrow \varepsilon \in P$ , и по построению автомата в этом случае  $S \in F$ , то

$$(S, \varepsilon) \vdash^* (S, \varepsilon) \text{ (рефлексивность!),}$$

т. е.  $x \in T(M)$ .

Если  $(q_0, x) \vdash^* (p, \varepsilon)$ , то  $S \xRightarrow{*} x$ .

---

II. Пусть теперь  $x = a_1 a_2 \dots a_n$  и  $x \in T(M)$ ,  $n \geq 1$ .

Тогда существует последовательность конфигураций вида

$$(S, a_1 a_2 \dots a_n) \vdash (A_1, a_2 \dots a_n) \vdash \dots$$

$$\dots \vdash (A_{n-2}, a_{n-1}) \vdash (A_{n-1}, a_n) \vdash (A, \varepsilon),$$

где  $A \in F$ .

Очевидно, что

$$A_1 \in \delta(S, a_1), \quad A_2 \in \delta(A_1, a_2), \quad \dots,$$

$$A_{n-1} \in \delta(A_{n-2}, a_{n-1}), \quad A \in \delta(A_{n-1}, a_n).$$

Если  $(q_0, x) \vdash^* (p, \varepsilon)$ , то  $S \Rightarrow^* x$ .

---

Но это возможно лишь при условии, что существуют правила

$$S \rightarrow a_1 A_1, A_1 \rightarrow a_2 A_2, \dots, A_{n-2} \rightarrow a_{n-1} A_{n-1}, \\ A_{n-1} \rightarrow a_n \in P.$$

Используя их, легко построить вывод вида  $S \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \dots \Rightarrow a_1 a_2 \dots a_{n-1} A_{n-1} \Rightarrow a_1 a_2 \dots a_{n-1} a_n = x$ , т. е.  $x \in L(G)$ .

Если же  $x = \varepsilon$  и  $x \in T(M)$ , то  $(S, \varepsilon) \vdash^* (S, \varepsilon)$  и  $S \in F$ . Но это возможно, если только существует правило  $S \rightarrow \varepsilon \in P$ . А тогда  $S \Rightarrow \varepsilon$  и  $x \in L(G)$ . Что и требовалось доказать.

**Теорема 3.5.** Пусть  $M = (Q, \Sigma, \delta, q_0, F)$  — конечный автомат.

Существует грамматика  $G$  типа 3, такая, что  $L(G) = T(M)$ .

Доказательство. Без потери общности можно считать, что  $M$  — dfa. Построим грамматику  $G = (V_N, V_T, P, S)$ , положив  $V_N = Q$ ,  $V_T = \Sigma$ ,  $S = q_0$ ,  
 $P = \{q \rightarrow ap \mid \delta(q, a) = p\} \cup$   
 $\cup \{q \rightarrow a \mid \delta(q, a) = p \text{ и } p \in F\}$ . Очевидно, что  $G$  — грамматика типа 3.

Если  $x \in T(M)$ , то  $x \in L(G)$ .

---

I. Пусть  $x \in T(M)$  и  $|x| > 0$ .

Покажем, что  $x \in L(G)$ .

Предположим, что  $x = a_1 a_2 \dots a_n$ ,  $n > 0$ .

Существует последовательность конфигураций автомата  $M$

$(q_0, a_1 a_2 \dots a_n) \vdash (q_1, a_2 \dots a_n) \vdash \dots \vdash (q_{n-1}, a_n) \vdash$   
 $\vdash (q_n, \varepsilon)$ , причём  $q_n \in F$ .

Соответственно

$\delta(q_0, a_1) = q_1, \delta(q_1, a_2) = q_2, \dots, \delta(q_{n-1}, a_n) = q_n$ .

Если  $x \in T(M)$ , то  $x \in L(G)$

---

По построению грамматики в множестве правил  $P$  существуют правила вида

$q_i \rightarrow a_{i+1}q_{i+1}$  ( $i = 0, 1, \dots, n-2$ ) и правило  $q_{n-1} \rightarrow a_n$ .

С их помощью можно построить вывод

$$\begin{aligned} S = q_0 &\Rightarrow a_1q_1 \Rightarrow a_1a_2q_2 \Rightarrow \dots \Rightarrow \\ &\Rightarrow a_1a_2 \dots a_{n-1}q_{n-1} \Rightarrow a_1a_2 \dots a_n. \end{aligned}$$

А это значит, что  $x \in L(G)$ .

Если  $x \in L(G)$ , то  $x \in T(M)$

---

II. Пусть  $x \in L(G)$  и  $|x| > 0$ .

Покажем, что  $x \in T(M)$ .

Предположим, что  $x = a_1 a_2 \dots a_n$ ,  $n > 0$ .

Существует вывод вида

$$q_0 \Rightarrow a_1 q_1 \Rightarrow a_1 a_2 q_2 \Rightarrow \dots \Rightarrow a_1 a_2 \dots a_{n-1} q_{n-1} \Rightarrow \\ \Rightarrow a_1 a_2 \dots a_{n-1} a_n.$$

Соответственно существуют правила

$$q_i \rightarrow a_{i+1} q_{i+1} \quad (i = 0, 1, \dots, n-2) \text{ и правило} \\ q_{n-1} \rightarrow a_n.$$

Если  $x \in L(G)$ , то  $x \in T(M)$

---

Очевидно, что они обязаны своим существованием тому, что

$\delta(q_i, a_{i+1}) = q_{i+1}$  ( $i = 0, 1, \dots, n-2$ ) и  $q_n \in F$ .

А тогда существует последовательность конфигураций  $\text{fa } M$  вида

$$(q_0, a_1 a_2 \dots a_n) \vdash (q_1, a_2 \dots a_n) \vdash \dots$$
$$\dots (q_{n-2}, a_{n-1}) \vdash (q_{n-1}, a_n) \vdash (q_n, \varepsilon),$$

причём  $q_n \in F$ . Это значит, что  $x \in T(M)$ .



Если  $x \in L(G)$ , то  $x \in T(M)$

---

Если  $q_0 \notin F$ , то  $\varepsilon \notin T(M)$  и  $L(G) = T(M)$ .

Если  $q_0 \in F$ , то  $\varepsilon \in T(M)$ . В этом случае  $L(G) = T(M) \setminus \{\varepsilon\}$ . По теореме [2.2](#) мы можем получить из  $G$  новую грамматику  $G_1$  типа 3, такую, что

$$L(G_1) = L(G) \cup \{\varepsilon\} = T(M).$$

Что и требовалось доказать.

**Пример 3.4.** Рассмотрим грамматику типа 3

$G = (\{S, B\}, \{0, 1\}, P, S)$ , где

$P = \{S \rightarrow 0B, B \rightarrow 0B, B \rightarrow 1S, B \rightarrow 0\}$ .

Мы можем построить ndfa

$M = (\{S, B, A\}, \{0, 1\}, \delta, S, \{A\})$ ,

где  $\delta$  определяется следующим образом:

- |                                 |                                 |
|---------------------------------|---------------------------------|
| 1) $\delta(S, 0) = \{B\}$ ,     | 2) $\delta(S, 1) = \emptyset$ , |
| 3) $\delta(B, 0) = \{B, A\}$ ,  | 4) $\delta(B, 1) = \{S\}$ ,     |
| 5) $\delta(A, 0) = \emptyset$ , | 6) $\delta(A, 1) = \emptyset$ . |

## Пример 3.4

По теореме 3.5  $T(M) = L(G)$ , в чём легко убедиться и непосредственно, ибо:

$$(1) S = 0B;$$

$$(2) B = 0B \mid 0 \mid 1S;$$

Подставим (1) в (2), получаем:

$$(2') B = 0B \mid 0 \mid 10B = 0 \mid (0 \mid 10) B = (0 \mid 10)^* 0.$$

Подставим (2') в (1), получаем:

$$(1') S = 0(0 \mid 10)^* 0, \text{ т. е. } L = L(G) = 0(0 \mid 10)^* 0.$$

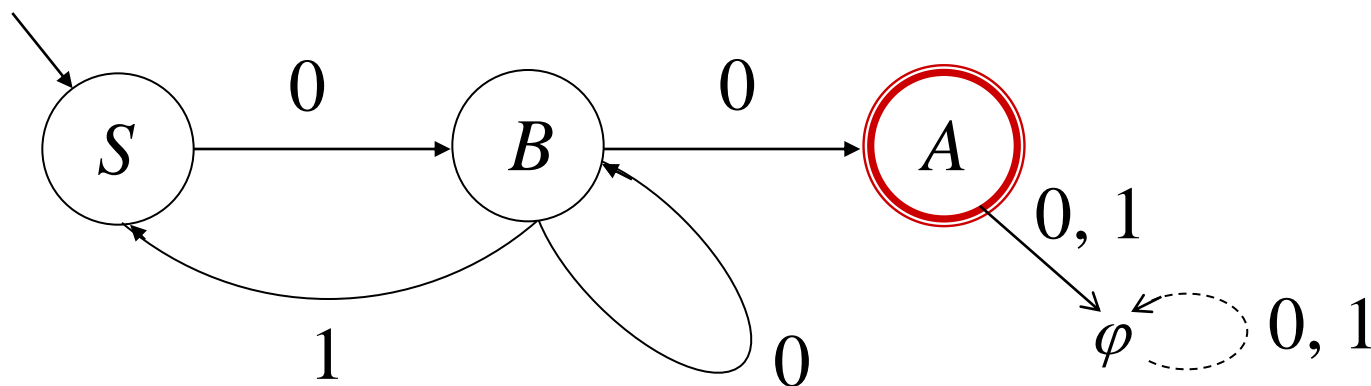


Рис. 3.5 (а). ndfa  $M$ .

### Пример 3.4

---

Теперь мы используем построения теоремы 3.4, чтобы найти dfa  $M'$ , эквивалентный ndfa  $M$ .

Положим  $M' = (Q', \{0, 1\}, \delta', [S], F')$ , где  
 $Q' = \{\varphi, [S], [B], [A], [S, B], [S, A], [B, A],$   
 $[S, B, A]\},$   
 $F' = \{[A], [S, A], [B, A], [S, B, A]\};$

### Пример 3.4

---

Отображение  $\delta'$  определяется следующим образом:

- |   |   |
|---|---|
| 1) $\delta'([S], 0) = [B]$ ,                              | 2) <del><math>\delta'([S], 1) = \varphi</math>,</del>     |
| 3) $\delta'([B], 0) = [B, A]$ ,                           | 4) $\delta'([B], 1) = [S]$ ,                              |
| 5) $\delta'([B, A], 0) = [B, A]$ ,                        | 6) $\delta'([B, A], 1) = [S]$ ,                           |
| 7) <del><math>\delta'(\varphi, 0) = \varphi</math>,</del> | 8) <del><math>\delta'(\varphi, 1) = \varphi</math>.</del> |

*Намним, что в ndfa M:*

- |                                 |                                 |
|---------------------------------|---------------------------------|
| 1) $\delta(S, 0) = \{B\}$ ,     | 2) $\delta(S, 1) = \emptyset$ , |
| 3) $\delta(B, 0) = \{B, A\}$ ,  | 4) $\delta(B, 1) = \{S\}$ ,     |
| 5) $\delta(A, 0) = \emptyset$ , | 6) $\delta(A, 1) = \emptyset$ . |

### Пример 3.4

---

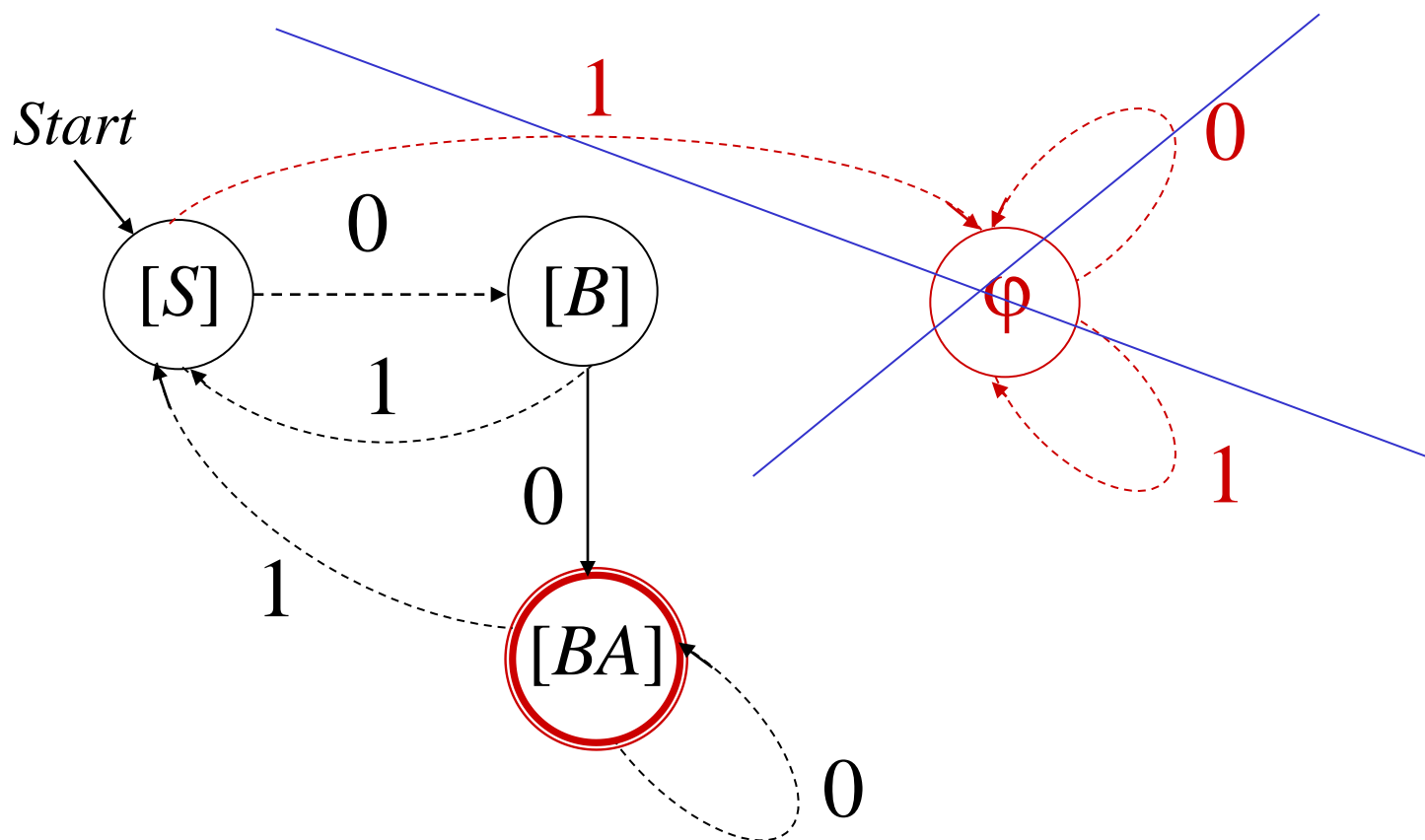


Рис. 3.5 (б). dfa  $M'$ .

### Пример 3.4

---

Ни в какие другие состояния, кроме  $\varphi$ ,  $[S]$ ,  $[B]$ ,  $[B, A]$ , автомат  $M'$  никогда не входит, и они могут быть удалены из множеств состояний  $Q'$ ,  $F'$  и  $\delta'$  как бесполезные.

Кроме того, ясно, что из состояния  $\varphi$  нет пути к единственному конечному состоянию  $[B, A]$ , т. е.  $\varphi$  — *состояние ошибки*. Его можно было бы отбросить со всеми инцидентными дугами.

Обозначим полученный таким образом dfa  $M'' = (\{[S], [B], [BA]\}, \{0, 1\}, \delta'', \{[B, A]\})$ , где  $\delta''$  отличается от  $\delta'$  отсутствием равенств (2), (7) и (8).

## Пример 3.4

Теперь согласно построениям теоремы 3.5 по автомату  $M''$  построим грамматику типа 3:

$$G_1 = (V_N, V_T, P, S),$$

где  $V_N = \{[S], [B], [B, A]\},$

$$V_T = \{0, 1\},$$

$$S = [S],$$

$$P = \{(1) [S] \rightarrow 0[B],$$

$$(2) [B] \rightarrow 0[B, A], \quad (3) [B] \rightarrow 0, \quad (4) [B] \rightarrow 1[S],$$

$$(5) [B, A] \rightarrow 0[B, A], \quad (6) [B, A] \rightarrow 0, \quad (7) [B, A] \rightarrow 1[S].$$

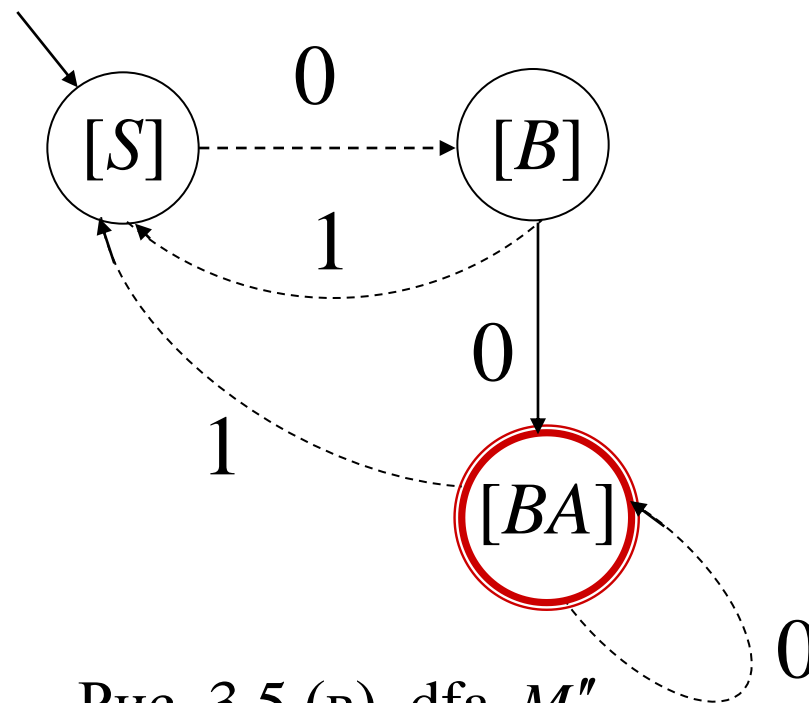


Рис. 3.5 (в). dfa  $M''$ .



Грамматика  $G_1$  сложнее грамматики  $G$ , но согласно трём предыдущим теоремам  $L(G_1) = L(G)$ .

Однако, грамматику  $G_1$  можно упростить, если заметить, что правила для  $[B, A]$  порождают в точности те же цепочки, что и правила для  $[B]$ .

И в самом деле, правило (2) приписывает 0 к цепочкам, порождаемым нетерминалом  $[B, A]$ , но правило (5) даёт сколько угодно 0 в начале этих цепочек и без этого правила (2).

Поэтому нетерминал  $[B, A]$  можно заменить всюду на  $[B]$  и исключить появившиеся дубликаты правил для  $[B]$ .

В результате получим грамматику  $G_2 = (\{[S], [B]\}, \{0, 1\}, P_2, [S])$ , где  $P_2 = \{(1) [S] \rightarrow 0[B],$   
(2)  $[B] \rightarrow 0[B],$   
(3)  $[B] \rightarrow 0,$   
(4)  $[B] \rightarrow 1[S]\}.$

Очевидно, что полученная грамматика  $G_2$  отличается от исходной грамматики  $\underline{G}$  лишь обозначениями нетерминалов.

Другими словами,  $L(G_2) = L(G).$

## § 3.5. Свойства языков типа 3

Поскольку класс языков, порождаемых грамматиками типа 3, равен классу множеств, распознаваемых конечными автоматами, мы будем использовать обе формулировки при описании свойств класса языков типа 3.

Прежде всего покажем, что языки типа 3 образуют *булеву алгебру*.

**Определение.** Булева алгебра множеств есть совокупность множеств, замкнутая относительно операций объединения, дополнения и пересечения.

**Определение.** Пусть  $L \subseteq \Sigma_1^*$  — некоторый язык и  $\Sigma_1 \subseteq \Sigma_2$ .

Под дополнением языка  $L$  подразумевается множество  $\bar{L} = \Sigma_2^* \setminus L$ .

**Пояснение.** Вложение  $\Sigma_1 \subseteq \Sigma_2$  предполагает, что дополнению принадлежат не только цепочки из “законных” символов из  $\Sigma_1$ , но и цепочки, содержащие “незаконные” символы, т. е. символы не из  $\Sigma_1$ .

**Лемма 3.1.** *Класс языков типа 3 замкнут относительно объединения.*

Доказательство. Возможны два подхода: один использует недетерминированные конечные автоматы, другой основывается на грамматиках. Мы будем пользоваться вторым подходом.

[Ret 92](#)   [Ret 112](#)

Пусть  $L_1 = L(G_1)$  и  $L_2 = L(G_2)$ , где  $G_1 = (V_N^{(1)}, V_T^{(1)}, P_1, S_1)$  и  $G_2 = (V_N^{(2)}, V_T^{(2)}, P_2, S_2)$ .  
— грамматики типа 3.

Можно предположить, что

$$V_N^{(1)} \cap V_N^{(2)} = \emptyset,$$

ибо в противном случае этого всегда можно достичь путём переименования нетерминалов данных грамматик.

Предположим также, что  $S \notin V_N^{(1)} \cup V_N^{(2)}$ .

Построим новую грамматику

$G_3 = (\{S\} \cup V_N^{(1)} \cup V_N^{(2)}, V_T^{(1)} \cup V_T^{(2)}, P_3, S)$ , где

$$P_3 = (P_1 \cup P_2) \setminus \{S_1 \rightarrow \varepsilon, S_2 \rightarrow \varepsilon\} \cup \\ \cup \{S \rightarrow \alpha \mid \exists S_1 \rightarrow \alpha \in P_1 \text{ или } \exists S_2 \rightarrow \alpha \in P_2\}.$$

Очевидно, что  $S \xRightarrow{G_3} \alpha$  тогда и только тогда, когда  $S_1 \xRightarrow{G_1} \alpha$  или  $S_2 \xRightarrow{G_2} \alpha$ .

В первом случае из  $\alpha$  могут выводиться только цепочки в алфавите  $V_N^{(1)} \cup V_T^{(1)}$ , во втором — в алфавите  $V_N^{(2)} \cup V_T^{(2)}$ .

Формально, если  $S_1 \xRightarrow{G_1} \alpha$ , то  $\alpha \xRightarrow{G_3}^* x$   
тогда и только тогда, когда  $\alpha \xRightarrow{G_1}^* x$ .

Аналогично, если  $S_2 \xRightarrow{G_2} \alpha$ , то  $\alpha \xRightarrow{G_3}^* x$   
тогда и только тогда, когда  $\alpha \xRightarrow{G_2}^* x$ .



Сопоставив всё сказанное, заключаем, что  $S \xRightarrow[G_3]{*} x$  тогда и только тогда, когда либо  $S_1 \xRightarrow[G_1]{*} x$ , либо  $S_2 \xRightarrow[G_2]{*} x$ .

А это и значит, что  $L(G_3) = L(G_1) \cup L(G_2)$ .

Что и требовалось доказать.

**Лемма 3.2.** *Класс множеств, распознаемых конечными автоматами (порождаемых грамматиками типа 3), замкнут относительно дополнения.*

Доказательство. Пусть

$M = (Q, \Sigma_1, \delta, q_0, F)$  — dfa и  $T(M) = L$ .

Пусть  $\Sigma_2$  — конечный алфавит, содержащий  $\Sigma_1$ , т. е.  $\Sigma_1 \subseteq \Sigma_2$ .

Мы построим fa  $M'$ , который распознаёт

$$\bar{L} = \Sigma_2^* \setminus L.$$

Положим

$$M' = (Q', \Sigma_2, \delta', q'_0, F'),$$

где  $Q' = Q \cup \{d\}$ ,  $F' = (Q \setminus F) \cup \{d\}$ , где  $d \notin Q$  — новое состояние “ловушка”.

(1)  $\delta'(q, a) = \delta(q, a)$  для каждого  $q \in Q$  и  $a \in \Sigma_1$ , если  $\delta(q, a)$  определено;

(2)  $\delta'(q, a) = d$  для тех  $q \in Q$  и  $a \in \Sigma_2$ , для которых  $\delta(q, a)$  не определено;

(3)  $\delta'(d, a) = d$  для каждого  $a \in \Sigma_2$ .

Интуитивно  $fa\ M'$  получается расширением входного алфавита  $\Sigma_1$   $fa\ M$  до алфавита  $\Sigma_2$ , добавлением состояния “ловушки”  $d$  и затем перестановкой конечных и неконечных состояний.

Очевидно, что  $fa\ M'$  распознаёт язык

$$\bar{L} = \Sigma_2^* \setminus L.$$

**Теорема 3.6.** *Класс множеств, принимаемых конечными автоматами, образует булеву алгебру.*

Доказательство непосредственно следует из лемм 3.1 и 3.2 и того факта, что

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}.$$

**Теорема 3.7.** *Все конечные множества распознаются конечными автоматами.*

Доказательство. Рассмотрим множество, содержащее только одну цепочку  $x = a_1a_2\dots a_n$ ,  $n > 0$ .

Мы можем построить конечный автомат  $M$ , принимающий только эту цепочку.

Положим  $M = (\{q_0, q_1, q_2, \dots, q_n, p\}, \{a_1, a_2, \dots, a_n\}, \delta, q_0, \{q_n\})$ , где

1.  $\delta(q_i, a_{i+1}) = q_{i+1}$ ,
2.  $\delta(q_i, a) = p$ , если  $a \neq a_{i+1}$  ( $i = 0, 1, \dots, n-1$ ),
3.  $\delta(q_n, a) = \delta(p, a) = p$  для всех  $a$ .

Очевидно, что  $\text{fa } M$  принимает только цепочку  $x$ .

Множество, содержащее только *пустую цепочку*, распознаётся конечным автоматом

$$M = (\{q_0, p\}, \Sigma, \delta, q_0, \{q_0\}), \text{ где} \\ \delta(q_0, a) = \delta(p, a) = p \text{ для всех } a \in \Sigma.$$

Действительно, только пустая цепочка оставит автомат в состоянии  $q_0$ , являющееся конечным.

*Пустое множество* распознаётся конечным автоматом

$$M = (\{q_0\}, \Sigma, \delta, q_0, \emptyset), \text{ где} \\ \delta(q_0, a) = q_0 \text{ для всех } a \in \Sigma.$$

Утверждение теоремы немедленно следует из свойства замкнутости языков типа 3 относительно объединения. Что и требовалось доказать.

**Определение.** Произведением или конкатенацией языков  $L_1$  и  $L_2$  называется множество  $L_1L_2 = \{z \mid z = xy, x \in L_1, y \in L_2\}$ .

Другими словами, каждая цепочка в языке  $L_1L_2$  есть конкатенация цепочки из  $L_1$  с цепочкой из  $L_2$ .

Например, если

$$L_1 = \{01, 11\} \text{ и } L_2 = \{0, 1, 101\},$$

то множество

$$L_1L_2 = \{010, 011, 01101, 110, 111, 11101\}.$$



**Теорема 3.8.** *Класс множеств, распознаваемых конечными автоматами, замкнут относительно произведения.*

Доказательство. Пусть

$$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1) \text{ и}$$

$$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$$

— детерминированные конечные автоматы, распознающие языки  $L_1$  и  $L_2$  соответственно.

Без потери общности можно предполагать, что

(1)  $Q_1 \cap Q_2 = \emptyset$  и (2)  $\Sigma_1 = \Sigma_2 = \Sigma$ ,

ибо противном случае мы могли бы добавить в  $Q_1$  и  $Q_2$  “мёртвые” состояния, вроде состояния “ловушки”  $d$ , как при доказательстве леммы [3.2](#).

Именно, сохраняя прежние обозначения для fa  $M_1$  и  $M_2$ , положим

$$M_1 = (Q_1 \cup \{d_1\}, \Sigma, \delta_1, q_1, F_1),$$

$$M_2 = (Q_2 \cup \{d_2\}, \Sigma, \delta_2, q_2, F_2),$$

где  $\Sigma = \Sigma_1 \cup \Sigma_2$ , а исходные отображения  $\delta_1$  и  $\delta_2$  пополним равенствами:

$$\delta_1(q, a) = d_1 \text{ для каждого } q \in Q_1 \text{ и } a \in \Sigma_2 \setminus \Sigma_1,$$

$$\delta_1(d_1, c) = d_1 \text{ для каждого } c \in \Sigma,$$

$$\delta_2(p, b) = d_2 \text{ для каждого } p \in Q_2 \text{ и } b \in \Sigma_1 \setminus \Sigma_2,$$

$$\delta_2(d_2, c) = d_2 \text{ для каждого } c \in \Sigma.$$

Мы построим ndfa  $M$ , распознающий язык  $L_1L_2$ .

Положим  $M = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F)$ , где

- 1)  $\delta(q, a) = \{\delta_1(q, a)\}$  для любого  $q \in Q_1 \setminus F_1$ ,
- 2)  $\delta(q, a) = \{\delta_1(q, a), \delta_2(q_2, a)\}$  для любого  
 $q \in F_1$ ,
- 3)  $\delta(q, a) = \{\delta_2(q, a)\}$  для любого  $q \in Q_2$ .

Если  $\varepsilon \notin L_2$ , то  $F = F_2$ , иначе  $F = F_1 \cup F_2$ .

Правило 1 воспроизводит движения автомата  $M_1$  до тех пор, пока он не достигает какого-нибудь из его конечных состояний, приняв некоторую (возможно пустую) начальную часть входной цепочки, принадлежащей языку  $L_1$ .

Затем согласно правилу 2 он может продолжать повторять движения автомата  $M_1$  или перейти в режим воспроизведения движений автомата  $M_2$ , начиная с его начального состояния (на том же текущем входном символе).

В последнем случае все дальнейшие движения, благодаря правилу 3, повторяют движения автомата  $M_2$ .

Если fa  $M_2$  принимает (возможно пустое) окончание входной цепочки, принадлежащее языку  $L_2$ , то и автомат  $M$  принимает всю входную цепочку. Другими словами,

$$T(M) = L_1 L_2.$$

**Определение.** *Замыкание языка  $L$*

есть множество

$$L^* = \bigcup_{k=0}^{\infty} L^k.$$

Предполагается, что

$$L^0 = \{\varepsilon\}, L^n = L^{n-1}L = LL^{n-1} \text{ при } n > 0.$$

**Пример 3.5.** Если  $L = \{01, 11\}$ , то

$$L^* = \{\varepsilon, 01, 11, 0101, 0111, 1101, 1111, \dots\}.$$

**Теорема 3.9.** *Класс множеств, принимаемых конечными автоматами, замкнут относительно замыкания.*

Доказательство. Пусть  $M = (Q, \Sigma, \delta, q_0, F)$  — dfa и  $L = T(M)$ .

Построим ndfa  $M'$ , который распознаёт язык  $L^*$ .

Положим

$$M' = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', F \cup \{q_0'\}),$$

где  $q_0' \notin Q$  — новое состояние;



для всех  $a \in \Sigma$ :

$$\delta'(q_0', a) = \begin{cases} \{\delta(q_0, a), q_0\}, & \text{если } \delta(q_0, a) \in F, \\ \{\delta(q_0, a)\} & \text{в противном случае;} \end{cases}$$

для всех  $q \in Q$  и  $a \in \Sigma$  :

$$\delta'(q, a) = \begin{cases} \{\delta(q, a), q_0\}, & \text{если } \delta(q, a) \in F, \\ \{\delta(q, a)\} & \text{в противном случае.} \end{cases}$$

Предназначение нового начального состояния  $q_0'$  — принимать пустую цепочку.

Если  $q_0 \notin F$ , мы не можем просто сделать  $q_0$  конечным состоянием, поскольку автомат  $M$  может снова прийти в состояние  $q_0$ , прочитав некоторую непустую цепочку, не принадлежащую языку  $L$ .

Докажем теперь, что  $T(M') = L^*$ .

I. Предположим, что  $x \in L^*$ . Тогда либо  $x = \varepsilon$ , либо  $x = x_1x_2...x_n$ , где  $x_i \in L$  ( $i = 1, 2, ..., n$ ).

Очевидно, что автомат  $M'$  принимает  $\varepsilon$ .

Ясно также, что из  $x_i \in L$  следует, что  $\delta(q_0, x_i) \in F$  и множества  $\delta'(q_0', x_i)$  и  $\delta'(q_0, x_i)$  каждое содержит состояние  $q_0$  и некоторое состояние  $p \in F$  (возможно  $p = q_0$ ).

Итак,

$$(q_0', x) = (q_0', x_1 x_2 \dots x_n) \mid_{M'}^* (q_0, x_2 \dots x_n) \mid_{M'}^* \dots \\ \dots \mid_{M'}^* (q_0, x_n) \mid_{M'}^* (p, \varepsilon), \text{ где } p \in F.$$

Следовательно,  $x \in T(M')$ .

II. Предположим теперь, что

$$x = a_1 a_2 \dots a_n \in T(M'):$$

$$(q_0', a_1 a_2 \dots a_n) \vdash_{M'} (q_1', a_2 \dots a_n) \vdash_{M'} \dots \vdash_{M'} \\ \vdash_{M'} (q_{n-1}', a_n) \vdash_{M'} (q_n', \varepsilon), q_i' \in Q \ (i = 1, 2, \dots, n), \\ \text{причём } q_n' \in F \cup \{q_0'\}.$$

Ясно, что  $q_n' = q_0'$  только в случае  $n = 0$ .

В общем случае существует подпоследовательность состояний

$$q'_{i_1}, q'_{i_2}, \dots, q'_{i_m} \ (m \geq 1)$$

такая, что значение  $q'_{i_k} = q_0, k = 1, 2, \dots, m-1,$   
 $q'_{i_m} = q'_n \in F.$

$$\begin{array}{ccccccc}
 x = a_1 a_2 \dots a_{j_1} & a_{j_1+1} \dots a_{j_2} & a_{j_2+1} \dots a_{j_{m+1}} & \dots & a_n & \in T(M') \\
 \uparrow & \underbrace{\hspace{1.5cm}}_{x_1 \in L} & \uparrow & \underbrace{\hspace{1.5cm}}_{x_2 \in L} & \uparrow & \underbrace{\hspace{1.5cm}}_{x_m \in L} & \uparrow \\
 q_0' & & q_0 & & q_0 & & q_0 \\
 & & \parallel & & \parallel & & \parallel \\
 & & q_{i_1}' & & q_{i_2}', \dots, & & q_{i_{m-1}}' \\
 & & & & & & q_{i_m}' = \\
 & & & & & & q_n' \in F
 \end{array}$$

$$x = x_1 x_2 \dots x_m \in L^m \subset L^*.$$

Это значит, что в такие моменты ndfa  $M'$  заканчивает просмотр очередного фрагмента  $x$ , принимаемого dfa  $M$ .

Формально, при некоторых  $j = j_k$  ( $1 \leq k \leq m$ ) имеет место

$\delta$ , но не  $\delta'$  !

$$q'_{j_k} \in \delta'(q'_{j_k - 1}, a_{j_k}), q'_{j_k} \in \delta(q'_{j_k - 1}, a_{j_k}) \text{ при } q'_{j_k} \in F.$$

Поэтому  $x = x_1 x_2 \dots x_m$ , где  $\delta(q_0, x_k) \in F$  для  $1 \leq k \leq m$ .

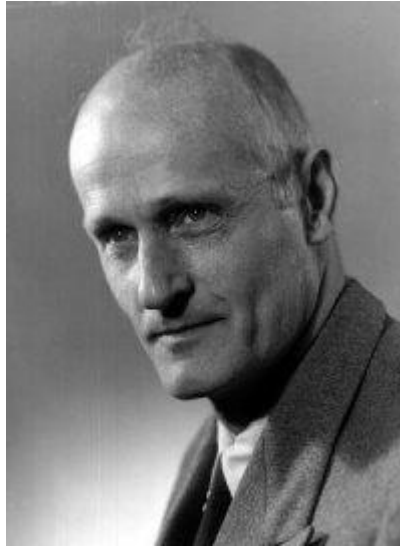
Это означает, что  $x_k \in L$ , а  $x \in L^m \subset L^*$ .

Что и требовалось доказать.

**Теорема 3.10.** (С. Клини). *Класс множеств, принимаемых конечными автоматами, является наименьшим классом, содержащим все конечные множества, замкнутым относительно объединения, произведения и замыкания.*

Доказательство. Обозначим класс регулярных множеств буквой  $M$ , а наименьший класс множеств, принимаемых конечными автоматами, содержащий все конечные множества и замкнутый относительно объединения, произведения и замыкания, буквой  $\mathcal{M}$ .

# Stephen Cole Kleene



*January 5, 1909, Hartford, Connecticut, United States –  
January 25, 1994, Madison, Wisconsin*



То, что  $\mathcal{M} \subseteq M$ , является непосредственным следствием из леммы 3.1 (объединение) и теорем 3.7 (конечные множества), 3.8 (произведение) и 3.9 (замыкание).

Остается показать обратное вложение, то есть, что  $M \subseteq \mathcal{M}$ .

Пусть  $L_1$  — множество, принимаемое некоторым dfa

$$M = (\{q_1, q_2, \dots, q_n\}, \Sigma, \delta, q_1, F).$$

Требуется показать, что  $L_1 = T(M) \in \mathcal{M}$ .

Пусть  $R_{ij}^k$  обозначает множество всех цепочек  $x$ , таких, что  $\delta(q_i, x) = q_j$ , причём, если  $y$  является непустым префиксом  $x$ , не совпадающим с  $x$  ( $x = yz$ ,  $y \neq \varepsilon$ ,  $z \neq \varepsilon$ ), то  $\delta(q_i, y) = q_l$ , где  $l \leq k$ .

Другими словами,  $R_{ij}^k$  — множество всех цепочек, которые переводят fa  $M$  из состояния  $q_i$  в состояние  $q_j$ , не проходя через какое-либо состояние  $q_l$ , где  $l > k$ .

Под “прохождением через состояние  $q_l$ ” мы подразумеваем вход в состояние  $q_l$  и выход из него. Но  $i$  и  $j$  могут быть больше  $k$ .

Мы можем определить  $R_{ij}^k$  рекурсивно:

$$R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} R_{kk}^{k-1*} R_{kj}^{k-1},$$

$$R_{ij}^0 = \{a \mid a \in \Sigma, \delta(q_i, a) = q_j\}.$$

Приведённое определение  $R_{ij}^k$  неформально означает, что цепочки, которые переводят автомат  $M$  из состояния  $q_i$  в состояние  $q_j$  без перехода через состояния выше, чем  $q_k$ ,

(1) либо все находятся во множестве  $R_{ij}^{k-1}$ ,  
т. е. никогда не приводят автомат в состояние столь высокое, как  $q_k$ ,

(2) либо каждая такая цепочка начинается с цепочки из множества  $R_{ik}^{k-1}$  (которая переводит автомат  $M$  в состояние  $q_k$  первый раз), за которой следует сколько-то цепочек из множества  $R_{kk}^{k-1}$ , переводящих автомат  $M$  из состояния  $q_k$  снова в состояние  $q_k$  без перехода через состояние  $q_k$  и высшие по номеру состояния, за которыми следует цепочка из множества  $R_{kj}^{k-1}$ , переводящая автомат  $M$  из состояния  $q_k$  в состояние  $q_j$ , не достигая состояния  $q_k$  и большие по номеру состояния.

Индукцией по параметру  $k$  мы можем показать, что множество  $R_{ij}^k$  для всех  $i$  и  $j$  находятся в пределах класса  $\mathcal{M}$ .

База. Пусть  $k = 0$ .

Утверждение очевидно, поскольку все множества  $R_{ij}^0$  являются конечными.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех  $k$ , таких, что  $0 \leq k \leq m$  ( $0 \leq m < n$ ).

Индукционный переход. Докажем, что утверждение верно и для  $k = m + 1$ .

Это так, поскольку  $R_{ij}^{m+1}$  выражается через объединение, конкатенацию и замыкание различных множеств вида  $R_{pq}^m$ , каждое из которых по индукционному предположению находится в классе  $\mathcal{M}$ .

Остается заметить, что  $L_1 = \bigcup_{q_j \in F} R_{1j}^n$ .

Итак, из  $L_1 = T(M)$  следует, что  $L_1 \in \mathcal{M}$ .

Что и требовалось доказать.

***Следствие 3.1*** (из теоремы Клини). Из теоремы 3.10 следует, что любое выражение, построенное из конечных подмножеств множества  $\Sigma^*$ , где  $\Sigma$  — конечный алфавит, и конечного числа операций объединения ' $\cup$ ', произведения ' $\cdot$ ' (символ обычно опускаемый) и замыкания ' $*$ ' со скобками, которые определяют порядок действий, обозначает множество, принимаемое некоторым конечным автоматом.



И наоборот, каждое множество, принимаемое некоторым конечным автоматом, может быть представлено в виде такого выражения, которое называется *регулярным выражением*.

### **Пример 3.5:** числа языка Паскаль.

Пусть  $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ . Тогда любое число языка Паскаль можно представить в виде следующего регулярного выражения:

$$D^+(\{.\}D^+ \cup \{\varepsilon\}) (\{e\}(\{+, -\} \cup \{\varepsilon\}) D^+ \cup \{\varepsilon\}).$$

Здесь использован символ *плюс Клини* (+), определяемый следующими равенствами:

$$A^+ = A^*A = AA^* = \bigcup_{k=1}^{\infty} A^k.$$

Напомним, что *звездочка Клини* (\*), обозначающая замыкание, определяется следующим равенством:

$$A^* = \bigcup_{k=0}^{\infty} A^k.$$

В примере 3.5 члены  $\{\varepsilon\}$  обеспечивают необязательность дробной части, знака порядка и самого порядка.

Минимальная цепочка, представляющая число на языке Паскаль, состоит из одной цифры.

Если ввести обозначение  $[R] = R \cup \{\varepsilon\}$ , где  $R$  — регулярное выражение, то множество чисел языка Паскаль можно представить следующей регулярной формулой:

$$D^+ [.D^+] [ e[+, -]D^+ ].$$

## **§ 3.6. Алгоритмически разрешимые проблемы, касающиеся конечных автоматов**

В этом параграфе мы покажем, что существуют алгоритмы, отвечающие на многие вопросы, касающиеся конечных автоматов и языков типа 3.

**Теорема 3.11.** *Множество цепочек, принимаемых конечным автоматом с  $n$  состояниями,*

*1) не пусто, тогда и только тогда, когда он принимает цепочку, длина которой меньше  $n$ ;*

*2) бесконечно, тогда и только тогда, когда он принимает цепочку, длина  $l$  которой удовлетворяет неравенству:  $n \leq l < 2n$ .*

Доказательство.

*Необходимость* условия 1 вытекает из следующих рассуждений *от противного*.

Предположим, что множество  $T(M) \neq \emptyset$ , но ни одной цепочки длиной меньше  $n$  в этом множестве не существует.

Пусть  $x \in T(M)$ , где  $M = (Q, \Sigma, \delta, q_0, F)$  — конечный автомат с  $n$  состояниями, и  $|x| \geq n$ , и пусть  $x$  — одна из самых коротких таких цепочек.

Очевидно, что существует такое состояние  $q \in Q$ , что  $x = x_1x_2x_3$ , где  $x_2 \neq \varepsilon$ , и  $\delta(q_0, x_1) = q$ ,  $\delta(q, x_2) = q$ ,  $\delta(q, x_3) \in F$ .

Но тогда  $x_1x_3 \in T(M)$ , поскольку

$$\delta(q_0, x_1x_3) = \delta(q_0, x_1x_2x_3) \in F.$$

В то же время  $|x_1x_3| < |x_1x_2x_3| = |x|$ , но это противоречит предположению, что  $x$  — одна из самых коротких цепочек, принимаемых  $\text{fa } M$ .



*Достаточность* условия 1 очевидна. Действительно, если конечный автомат принимает цепочку с меньшей длиной, чем  $n$ , то множество  $T(M)$  уже не пусто (какой бы длины цепочка ни была).

Докажем теперь утверждение 2.

*Необходимость* условия 2 доказывается способом *от противного*.

Пусть  $\mathcal{L}_A$   $M$  принимает бесконечное множество цепочек, и ни одна из них не имеет длину  $l$ ,  $n \leq l < 2n$ .

Если бы в множестве  $T(M)$  существовали только цепочки длиной  $l < n$ , то язык был бы конечен, но это не так. Поэтому существуют и цепочки длиной  $l \geq 2n$ .

Пусть  $x$  — одна из самых коротких цепочек, таких, что  $x \in T(M)$  и  $|x| \geq 2n$ .

Очевидно, что существует такое состояние  $q \in Q$ , что  $x = x_1x_2x_3$ , где  $1 \leq |x_2| \leq n$ , и

$$\delta(q_0, x_1) = q, \delta(q, x_2) = q, \delta(q, x_3) \in F.$$

Но тогда  $x_1x_3 \in T(M)$ , поскольку

$$\delta(q_0, x_1x_3) = \delta(q_0, x_1x_2x_3) \in F$$

при том, что  $|x_1x_3| \geq n$ , ибо  $|x| = |x_1x_2x_3| \geq 2n$  и  $1 \leq |x_2| \leq n$ .

Поскольку по предположению в  $T(M)$  цепочек длиной  $n \leq l < 2n$  не существует, то  $|x_1x_3| \geq 2n$ .

Следовательно, вопреки предположению, что  $x = x_1x_2x_3 \in T(M)$  — одна из *самых коротких* цепочек, длина которой больше или равна  $2n$ , нашлась *более короткая* цепочка  $x_1x_3 \in T(M)$  и тоже с длиной, большей или равной  $2n$ . Это противоречие доказывает необходимость условия 2.

*Достаточность* условия 2 вытекает из следующих рассуждений. Пусть существует  $x \in T(M)$ , причём  $n \leq |x| < 2n$ .

Как и ранее, можем утверждать, что существует  $q \in Q$ ,  $x = x_1x_2x_3$ , где  $x_2 \neq \varepsilon$ , и

$$\delta(q_0, x_1) = q, \delta(q, x_2) = q, \delta(q, x_3) \in F.$$

Но тогда цепочки вида  $x_1x_2^ix_3 \in T(M)$  при любом  $i \geq 0$ . Очевидно, что множество  $T(M)$  бесконечно.

Что и требовалось доказать.

***Следствие 3.2.*** Из доказанной теоремы следует существование алгоритмов, разрешающих вопрос о *пустоте*, *конечности* и *бесконечности* языка, принимаемого любым данным конечным автоматом.

Действительно, алгоритм, проверяющий *непустоту* языка, может систематически генерировать все цепочки постепенно увеличивающейся длины, но меньшей  $n$ .

Каждая из этих цепочек пропускается через автомат. Либо автомат примет какую-нибудь из этих цепочек, и тогда алгоритм завершится с положительным ответом, либо ни одна из этих цепочек не будет принята, и тогда алгоритм завершится с отрицательным результатом. В любом случае процесс завершается за конечное время.

Алгоритм для проверки *бесконечности* языка можно построить аналогичным образом, только он должен генерировать и тестировать цепочки длины от  $n$  до  $2n - 1$  включительно.



**Теорема 3.12.** *Существует алгоритм для определения, являются ли два конечных автомата эквивалентными (т. е. распознающими один и тот же язык).*

Доказательство. Пусть  $M_1$  и  $M_2$  — конечные автоматы, распознающими языки  $L_1$  и  $L_2$  соответственно.

По теореме 3.6 множество

$$L = (L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$$

распознаётся некоторым конечным автоматом  $M_3$ .

Легко видеть, что множество  $L = T(M_3)$  не пусто тогда и только тогда, когда  $L_1 \neq L_2$ .

Следовательно, согласно теореме 3.12 существует алгоритм для определения, имеет ли место  $L_1 = L_2$ .

Что и требовалось доказать.

**Пояснение.** *Необходимость* утверждения  $L = \emptyset \Leftrightarrow L_1 = L_2$  можно доказать следующим образом *от противного*.

Пусть  $L = \emptyset$ , но  $L_1 \neq L_2$ . Это значит, например, что существует цепочка  $x \in L_1$ , такая что  $x \notin L_2$ , т. е.  $x \in \bar{L}_2$ . Следовательно,  $x \in L_1 \cap \bar{L}_2$ . Но тогда  $L_1 \cap \bar{L}_2 \neq \emptyset$  и  $L \neq \emptyset$ , что противоречит предположению ( $L = \emptyset$ ).

*Достаточность* очевидна, ибо при  $L_1 = L_2$  имеем  $L = (L_2 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_1) = \emptyset$ .

Приложение.

**ПРОГРАММНАЯ  
РЕАЛИЗАЦИЯ ДКА!!!!!!**

# Приложение. Программная реализация FSA

---

В электронике, да и в программировании, в простейшем случае мы имеем дело с так называемыми «переключательными функциями».

Это относительно примитивная абстракция не имеющая собственной памяти: на вход подается аргумент, а на выход некое значение.

**Выходное значение всегда зависит только от входного.**

## Приложение. Программная реализация FSA

---

А если нам необходимо, чтобы последующее значение функции зависело от предыдущего? От нескольких предыдущих? Тут мы уже приходим к некой абстракции с собственной памятью. Это и будет автомат.

**Выходное значение на автомате зависит от значения на входе и текущего состояния автомата.**

Конечный автомат соответственно потому так называется, что число его внутренних состояний конечно. А, наверное, самым простейшим из конечных автоматов является детерминированный: для каждого входного сигнала существует лишь одно состояние, в которое автомат может перейти из текущего.

Таким образом, автомат определяется следующим:  
начальным состоянием;

- входным алфавитом (набором возможных входных сигналов);
- множеством состояний;
- и таблицей переходов.

Собственно вся суть автомата определяется последним пунктом. Таблица переходов (также изображается как диаграмма переходов) состоит из 3-х столбцов: входной сигнал (символ), текущее состояние, следующее состояние.



## Базовый класс

Реализуем базовый класс. Мы уже определились, что нам нужно начальное состояние, текущее состояние и таблица переходов. Алфавит входных символов определяется задачей, поэтому также потребуется нормализация (упрощение) ввода.

Роль выходных сигналов будут выполнять исполняемые методы класса потомка.

Для упрощения, искусственно добавим в алфавит символ "\*" — любой другой символ.

# Приложение. Программная реализация FSA

---

```
package FSM;

use strict;
use Carp;
use vars qw($AUTOLOAD);

# Create new object
sub new {
    my $self = {};
    my ($proto, $initial) = @_ ;
    my $class = ref($proto) || $proto;

    # Init ourselves
    $self->{INITIAL} = $initial;
    $self->{CURRENT} = $initial;
    $self->{STATES} = {};

    bless ($self, $class);
    return $self;
}
```

# Приложение. Программная реализация FSA

---

```
sub setInitialState {  
    my ($self, $initial) = @_;  
    $self->{INITIAL} = $initial;  
    return $self;  
}  
  
sub setCurrentState {  
    my ($self, $current) = @_;  
    $self->{CURRENT} = $current;  
    return $self;  
}  
  
sub getCurrentState {  
    my ($self, $current) = @_;  
    return $self->{CURRENT};  
}
```

# Приложение. Программная реализация FSA

---

```
sub reset {  
    my $self = shift;  
    $self->{CURRENT} = $self->{INITIAL};  
    return $self;  
}  
sub addState {  
    my $self = shift;  
    my %args = @_;  
    $self->{STATES}->{$args{STATE}}->{$args{SYMBOL}} = {NEXT => $args{NEXT}, ACTION => $arg  
    return $self;  
}
```

# Приложение. Программная реализация FSA

---

```
sub removeState {
    my $self = shift;
    my %args = @_;
    if (exists $args{SYMBOL}) {
        delete $self->{STATES}->{$args{STATE}}->{$args{SYMBOL}};
    } else {
        delete $self->{STATES}->{$args{STATE}};
    }
    return $self;
}

# Be sure to override in child
sub normalize {
    my ($self, $symbol) = @_;
    my $ret = {};
    $ret->{SYMBOL} = $symbol;
    return $ret;
}
```

# Приложение. Программная реализация FSA

---

```
sub process {
    my ($self, $rawSymbol) = @_;
    my $state = $self->{STATES}->{$self->{CURRENT}};
    $rawSymbol = $self->normalize($rawSymbol);
    my $symbol = $rawSymbol->{SYMBOL};

    print STDERR "Current state " . $self->{CURRENT} . ", got symbol " . $symbol . "\n";
    if (!exists $state->{$symbol} && exists $state->{'*'}) {
        print STDERR "Unrecognized symbol " . $symbol . ", using *\n";
        $symbol = "*";
    }
}
```

# Приложение. Программная реализация FSA

---

```
# Do some action!
$state->{$symbol}->{ACTION}($self, $rawSymbol)
    if ref $state->{$symbol}->{ACTION};

# Switch state
if (exists $state->{$symbol}->{NEXT}) {
    $self->{CURRENT} = $state->{$symbol}->{NEXT};
} else {
    die "Don't know how to handle symbol " . $rawSymbol->{SYMBOL};
}

return $self;
}
```

Названия всех методов говорят сами за себя.

Остановимся на методах **normalize** и **process**.

## **Normalize**

преобразует входную строку в хэш, содержащий поле SYMBOL с упрощенным до алфавита автомата входным символом.

## **Process**

собственно осуществляет «тактирование» процессов перехода между состояниями, обрабатывая очередной сигнал.



**Пример.** Реализуем подобие чат-бота. Положим в его основу следующие правила: Перед выполнением команд надо сначала представиться (**login**)

- По команде **memorize** будем запоминать все что вводит пользователь, окончание по команде **exit**
- По команде **say N** выводить запомненную фразу номер N
- Завершение сеанса будет происходить по команде **exit**

# Приложение. Программная реализация FCA

---

Символ	Состояние	Следующее состояние	Действие
LOGIN	INIT	SESSION	Открываем сессию
*	INIT	INIT	-
*	SESSION	SESSION	-
SAY	SESSION	SESSION	Выводим строку номер N
EXIT	SESSION	INIT	-
MEMORIZE	SESSION	STORE	-
*	STORE	STORE	Сохраняем строку в буфер
EXIT	STORE	SESSION	-

## Приложение. Программная реализация FCA

---

Итак, алфавит автомата состоит из символов (LOGIN, MEMORIZE, SAY, EXIT, \*),

автомат имеет 3 состояния (INIT, SESSION и STORE).

В первую очередь зададим в конструкторе таблицу переходов, где необходимо — добавим ссылки на вызываемые методы.

# Приложение. Программная реализация FSA

---

```
package ChatBot;
use FSM;
@ISA = ("FSM");

use strict;
use Carp;
use vars qw($AUTOLOAD);

sub new {
    my $proto = shift;
    my $class = ref($proto) || $proto;

    my $self = $class->SUPER::new("INIT");
```

# Приложение. Программная реализация FCA

---

```
$self->addState(STATE => "INIT",    SYMBOL => "*",    NEXT => "INIT",    ACTION => \&d
$self->addState(STATE => "INIT",    SYMBOL => "LOGIN", NEXT => "SESSION", ACTION => \&d
$self->addState(STATE => "INIT",    SYMBOL => "EXIT",  NEXT => "INIT",    ACTION => \&d
$self->addState(STATE => "SESSION", SYMBOL => "*",    NEXT => "SESSION");
$self->addState(STATE => "SESSION", SYMBOL => "EXIT",  NEXT => "INIT");
$self->addState(STATE => "SESSION", SYMBOL => "SAY",    NEXT => "SESSION", ACTION => \&d
$self->addState(STATE => "SESSION", SYMBOL => "MEMORIZE",NEXT => "STORE");
$self->addState(STATE => "STORE",   SYMBOL => "*",    NEXT => "STORE",   ACTION => \&d
$self->addState(STATE => "STORE",   SYMBOL => "EXIT",  NEXT => "SESSION");

$self->{SESSION} = {};
$self->{LOGIN}   = "";

return $self;
}
```

# Приложение. Программная реализация FSA

---

```
sub normalize {  
  my ($self, $symbol) = @_;  
  my $ret = {};  
  
  if ($symbol =~ /^(\S+)(.*)$/) {  
    $ret->{SYMBOL} = uc $1;  
    $ret->{DATA}    = $2;  
    $ret->{RAW}     = $symbol;  
  } else {  
    $ret->{SYMBOL} = "*";  
    $ret->{DATA}   = $symbol;  
    $ret->{RAW}    = $symbol;  
  }  
  
  return $ret;  
}
```

# Приложение. Программная реализация FSA

---

```
sub doIntroduce {
    my $self = shift;
    print "Please introduce yourself first!\n";
    return $self;
}

sub doLogin {
    my ($self, $symbol) = @_;
    print "Welcome," . $symbol->{DATA} . "\n";
    $self->{LOGIN} = $symbol->{DATA};
    $self->{SESSION}->{$self->{LOGIN}} = () unless exists $self->{SESSION}->{$self->{LOGIN}}
    return $self;
}
```

# Приложение. Программная реализация FCA

---

```
sub doSay {  
    my ($self, $symbol) = @_;  
    if (defined $self->{SESSION}->{$self->{LOGIN}}->[$symbol->{DATA}]) {  
        print $self->{SESSION}->{$self->{LOGIN}}->[$symbol->{DATA}];  
    } else {  
        print "No record\n";  
    }  
    return $self;  
}
```



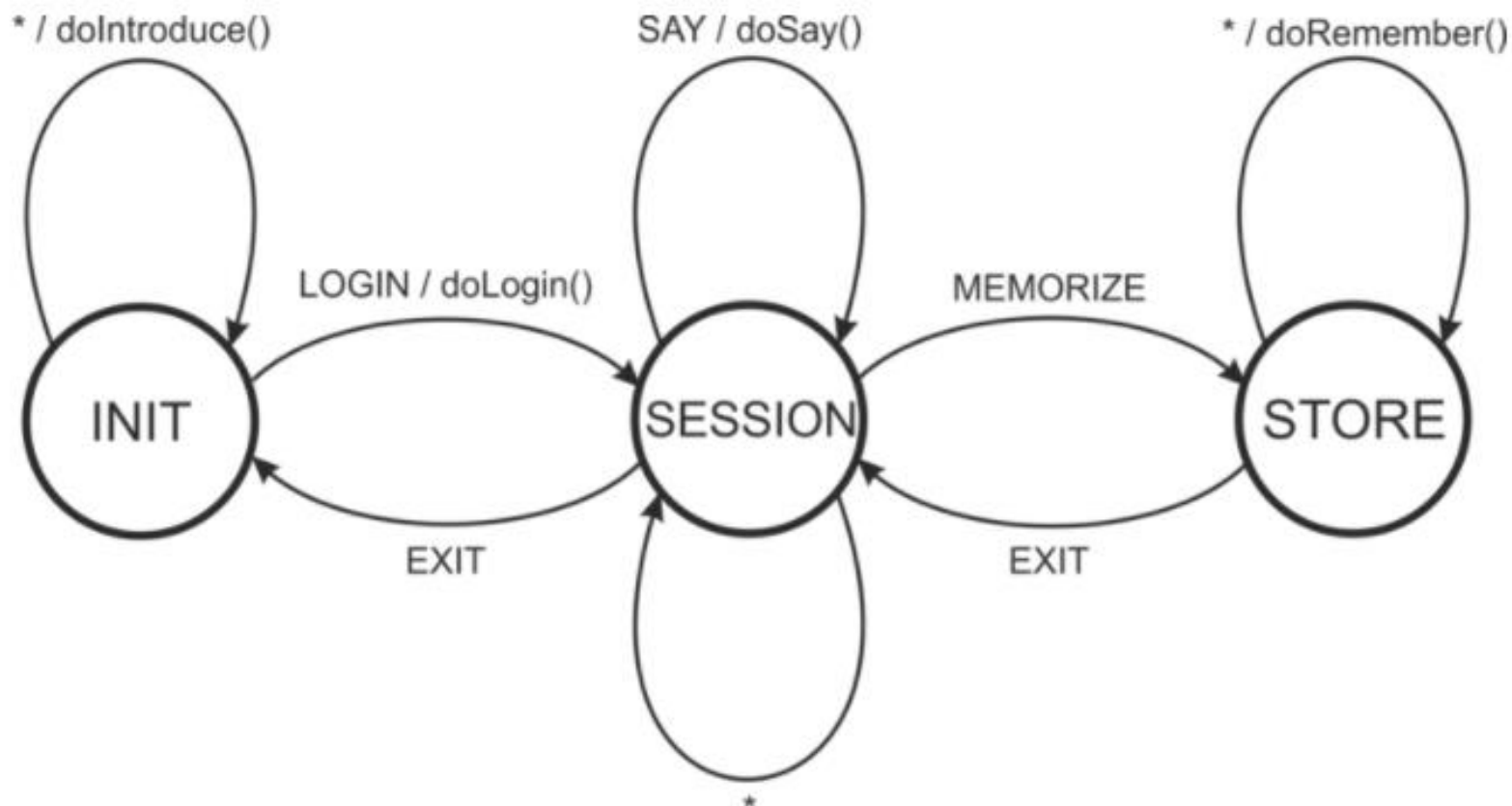
# Приложение. Программная реализация FSA

---

```
sub doRemember {  
    my ($self, $symbol) = @_;  
    push @{$self->{SESSION}->{$self->{LOGIN}} }, $symbol->{RAW};  
    return $self;  
}  
  
sub doQuit {  
    my ($self, $symbol) = @_;  
    print "Bye bye!\n";  
    exit;  
    return $self;  
}
```

# Приложение. Программная реализация FCA

---



# Приложение. Программная реализация FSA

---

Таблица переходов в данном случае будет иметь следующий вид

```
{
  'INIT' => {
    '*' => {
      'ACTION' => \&doIntroduce,
      'NEXT' => 'INIT'
    },
    'LOGIN' => {
      'ACTION' => \&doLogin,
      'NEXT' => 'SESSION'
    },
    'EXIT' => {
      'ACTION' => \&doQuit,
      'NEXT' => 'INIT'
    }
  },
}
```

# Приложение. Программная реализация FCA

---

```
'STORE' => {  
    '*' => {  
        'ACTION' => \&doRemember,  
        'NEXT' => 'STORE'  
    },  
    'EXIT' => {  
        'NEXT' => 'SESSION'  
    }  
},
```

# Приложение. Программная реализация FSA

---

```
'SESSION' => {  
    'SAY' => {  
        'ACTION' => \&doSay,  
        'NEXT' => 'SESSION'  
    },  
    '*' => {  
        'NEXT' => 'SESSION'  
    },  
    'MEMORIZE' => {  
        'NEXT' => 'STORE'  
    },  
    'EXIT' => {  
        'NEXT' => 'INIT'  
    }  
}  
}
```

# Приложение. Программная реализация FSA

---

## Тест.

На ввод дадим следующую последовательность.

```
hello world!  
login %username%  
hello world!  
say 3  
memorize  
hey, do you really remember everything i would say?  
let's check  
exit  
say 0  
exit  
hello  
login %username%  
say 1  
exit
```

# Приложение. Программная реализация FSA

---

## Результат.

```
Please introduce yourself first!  
Welcome, %username%  
No record  
hey, do you really remember everything i would say?  
Please introduce yourself first!  
Welcome, %username%  
let's check
```

**Спасибо за внимание!!!**