АВТОМАТНЫЕ РАСПОЗНАВАТЕЛИ И ЯЗЫКИ

Автоматные распознаватели и грамматики Хомского

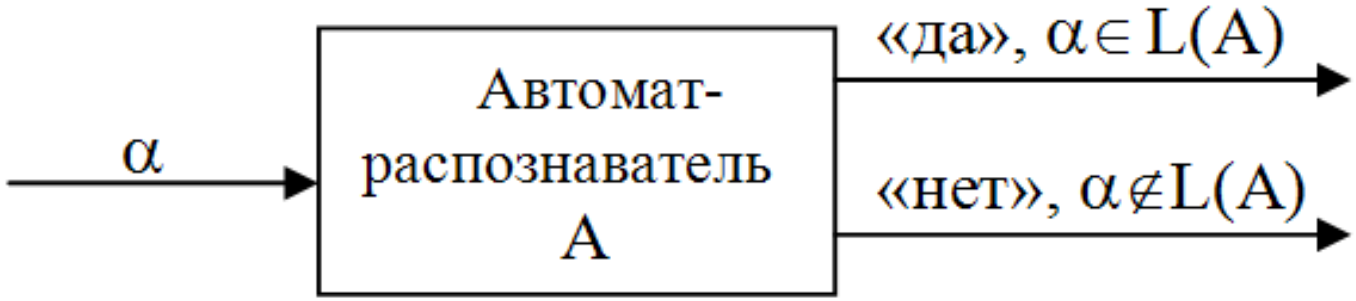
Для языков **L(G[Z])**, порождаемых грамматиками, распознаватель в виде диаграммы состояний или графа Г являлся по существу определяемым также порождаемой грамматикой **G[Z]**, то есть правилами вывода автоматной грамматики вида $N \to t\ M$, причём $N, M \in \mathbf{V_N}$, $t \in \mathbf{V_T}$.

Автоматные распознаватели и грамматики Хомского

Если граф Г задан другим способом, то он уже не является порождающим от редукций **P** порождающей грамматики.

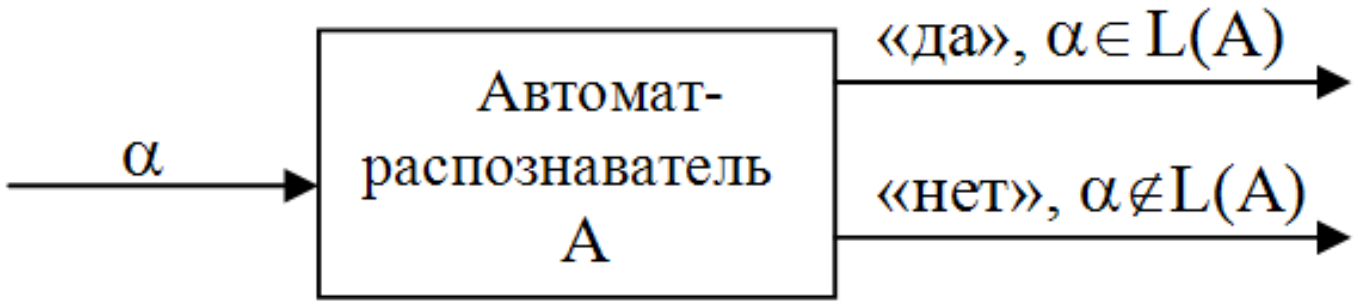
Такой граф называют автоматом-распознавателем **A**, а язык **L**(**A**) определяется *конфигурацией* автомата.

Если входная цепочка α, поданная на вход автоматного распознавателя, переводит **A** последовательно из некоторого начального состояния S_0 в одно из заключительных **F**, то $\alpha \in$ **L(A**).



Автоматные распознаватели и грамматики Хомского

Определение. Распознаватель (recognizer) – это алгоритм, представленный в виде конечного автомата, который, обрабатывая входные цепочки, принимает их («ДА») или отвергает («НЕТ»).



Автоматные распознаватели и грамматики Хомского

По классификации Хомского порождаемые грамматикой языки являются того же типа, что и грамматика **G**[Z], которая их порождает.

Связь порождающих грамматик из классификации Хомского и автоматов-распознавателей устанавливается через соответствующий язык.

С точки зрения конечных автоматов справедливы утверждения:

- Язык является автоматным, если он определяется (задаётся или распознаётся) конечным автоматом (возможно и недетерминированным);
- Язык является контекстно-свободным, если он определяется (задаётся или распознаётся) автоматом с магазинной памятью.

Конечные автоматы

α # - входная цепочка (# - конечный символ)

A - считывающий механизм

S - состояния автомата ($s_0 \in$ **S** - начальное состояние)

δ - функция переходов

Конфигурация конечного автомата определяется в виде

(s, ω, n), где s – текущее состояние **A**, s $\in$ **S**; ω - цепочка входных символов, $\omega \in \Sigma^+$; n – положение указателя в цепочке ω, n $\in$ {0, 1, 2, …}, n $\leq |\omega|$.

Конечные автоматы

Определение. Конечным автоматом-распознавателем **A** называется пятёрка объектов:

A = (**S**, Σ, s_0, δ, **F**), где

- **S** – конечное непустое множество (состояний);
- Σ – входной алфавит автомата (конечное непустое множество входных символов);
- $s_0 \in$ **S** – начальное состояние **A;**
- δ : **S** x $\Sigma \rightarrow$ **S** – функция переходов;
- **F** $\subseteq$ **S** – множество заключительных (конечных) состояний.

Конечные автоматы

Определим итерацию функции переходов δ как отображение на множество состояний декартова произведения множеств состояний автомата **S** на итерацию словаря, то есть δ^***: S x** $\Sigma^* \rightarrow$ **S.**

Определение. Конечный автомат **A = (S,** Σ**,** s_0, δ, **F)** допускает входную цепочку $\alpha \in \Sigma^*$, если символы α переводят **A** в одно из заключительных состояний **F** так, что $\delta^*(s_0, \alpha) \in$ **F**.

Конечные автоматы

Определение. Языком **L(A)** над словарём Σ^* называется множество всех цепочек α, которые допускает (принимает) конечный автомат, то есть

$$\mathbf{L(A)} = \{\alpha \mid \alpha \in \mathbf{\Sigma}^*, \delta^* (s_0, \alpha) \in \mathbf{F}\}.$$

Определение. Автоматным языком называют язык **L**(**A**). Иначе говоря, язык относится к классу автоматных, если существует конечный автомат **A,** принимающий этот язык.

Приведённое определение не противоречит ранее введенному определению **L(G**[Z]) для порождающих грамматик.

Конечные автоматы

Граф автоматной грамматики Г(**G**[Z]) легко превращается в КА, если положить:

$s_0 = Z$; $\Sigma = \mathbf{V_T}$, $\mathbf{F} = \{K\}$, $\mathbf{S} = \mathbf{V_N}$.

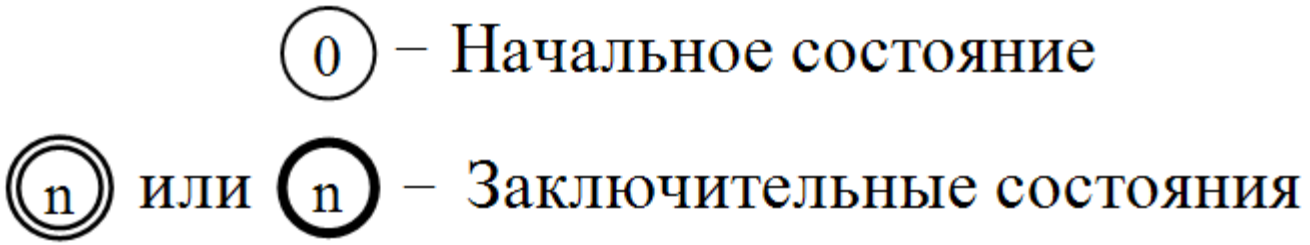
Тогда легко доказать, что множество редукций **P** эквивалентны $\{\delta\}$, т.к. правила вида $N \to t\ M$, где $N, M \in \mathbf{V_N}$, $t \in \mathbf{V_T}$, можно представить как отображение декартова произведения множества состояний и нетерминального словаря на множество состояний.

$\mathbf{P: V_N \times N_T \to V_N}$

Конечные автоматы

Конечные автоматы представляются в виде графов или диаграмм состояний.

Состояние автомата принято нумеровать или обозначать буквой. Начальное состояние нумеруется как 0, а заключительные состояния нумеруют n (n - целое).



Синтаксические диаграммы и конечные автоматы

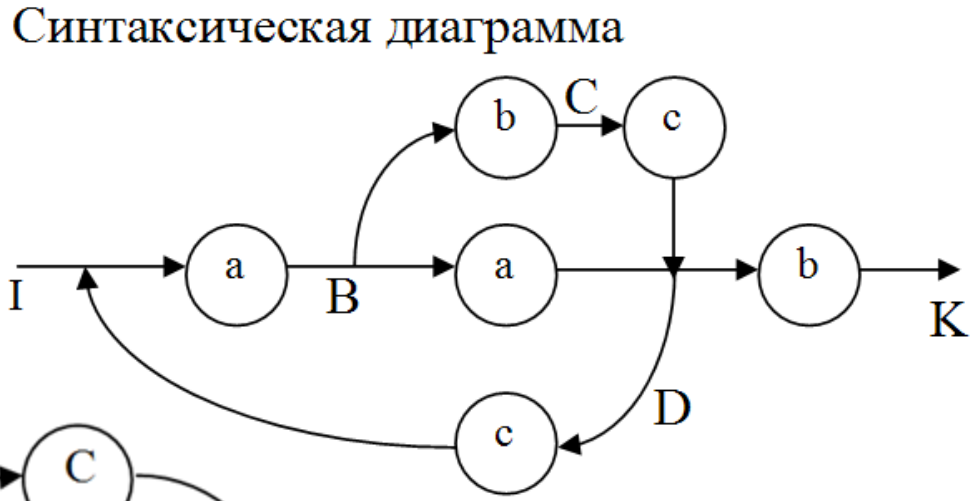
G[I]:

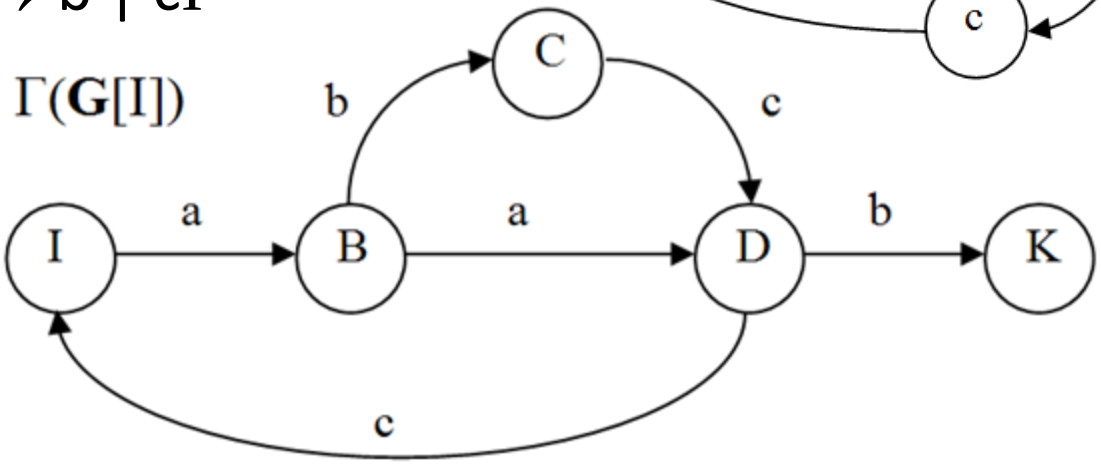
P: 1) $I \rightarrow aB$

2) $B \rightarrow bC \mid aD$

3) $C \rightarrow cD$

4) $D \rightarrow b \mid cI$





Язык **L(G**[I]**)** порождает множество цепочек {aab, aabaab,..., aab(aab)*, abcb, abccab,...}.

Синтаксические диаграммы и конечные автоматы

Если в Г(**G**[I]) нетерминалы принять множеством состояний **S** = {I, B, C, D, K}, то такой граф становится эквивалентным автоматным преобразователем (конечным автоматом), т.е. функция переходов $\delta(s,a)$, $\forall\ s \in \mathbf{S}$, $\forall\ a \in \mathbf{\Sigma}$ определена продукциями множества правил вывода **P**, причём $\mathbf{\Sigma = V_T}$, $\mathbf{S = V_N}$.

Легко убедиться в эквивалентности графа автоматной грамматики и конечного автомата.

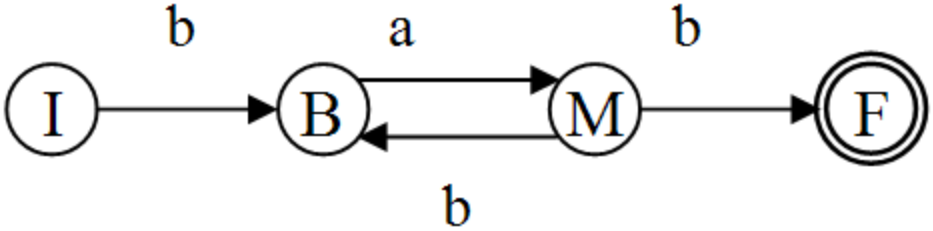
Эквивалентный конечный автомат в общем случае является недетерминированным ввиду наличия в автоматной грамматике правила вывода $A \to \Lambda$.

Синтаксические диаграммы и конечные автоматы

Определение. КА называется полностью определённым, если в каждом его состоянии существует функция перехода для всего словаря Σ, то есть $\forall$ a $\in \Sigma$, $\forall$ s $\in$ **S** $\exists$ δ (a, s) = **R** | **R** $\subseteq$ **S**.

Пример. Пусть $\mathbf{A}_1$ = ({I, M, B, F}, {a, b}, δ, I, {k})

Функция переходов δ: δ(I, b) = B; δ (B, a) = M; δ (M, b) = {B, F}.



L($\mathbf{A}_1$) = {bab, babab, bababab…} или **L($\mathbf{A}_1$)** = {b (ab)* ab}

Синтаксические диаграммы и конечные автоматы

Определение. Слово $\omega = a_1 \ldots a_k$ над алфавитом Σ допускается конечным автоматом **A = (S, Σ,** δ, s_0, **F),** если существует последовательность состояний $s_1, \ldots, s_n$ такая, что

$\forall$ i, j: $1 \leq i < n$, $1 \leq j < k$, $\exists\, \delta(s_i, a_j) = s_{i+1}$, $s_1 = s_0$, $s_n \in$ **F**.

Это определение следует понимать, как существование функции переходов δ, которая под управлением входных символов a_i (i = 1, …, n) последовательно переводит автомат из начального состояния s_0 в конечное - $s_n \in$ **F**.

Синтаксические диаграммы и конечные автоматы

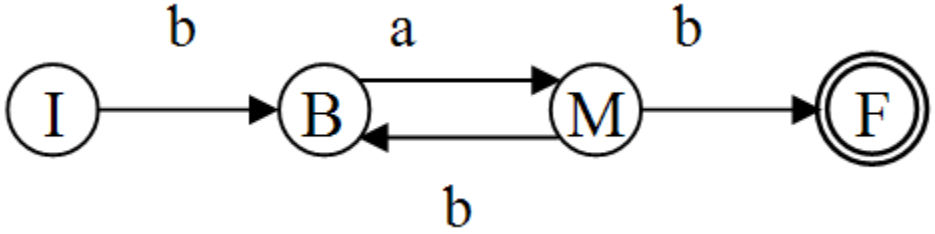
Предполагая что формальный язык представляется множеством слов (цепочек) ω, т.е. **L(A)** = $\{\omega\}$, $\omega \in \Sigma^*$, можно дать следующее определение языка:

Определение. Язык **L**(A) = $\{\omega\}$, $\omega \in \Sigma^*$, распознаётся конечным автомат тогда и только тогда, когда каждое слово языка над алфавитом Σ допускается этим конечным автоматом.

Если $\mathbf{A_1}$ принимает $\alpha \in \mathbf{L(A_1)}$ = {b $(ab)^*$ab}, то все другие цепочки $\beta \neq \alpha$, $\beta \in \Sigma^*$, Σ = {a, b} должны переводить конечный автомат в состояние ошибки (ERROR или E).

Синтаксические диаграммы и конечные автоматы

Пример.

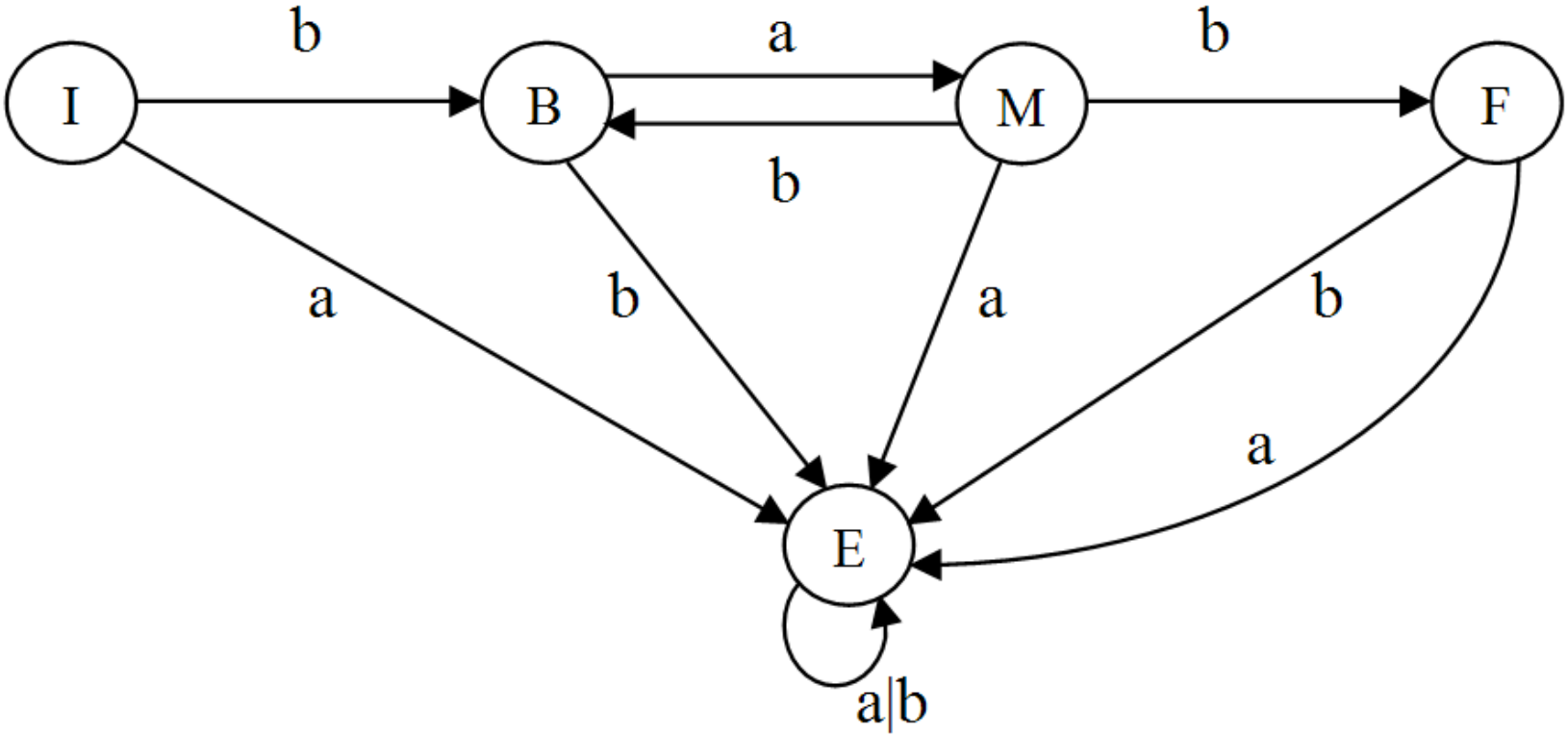


На состояние E замыкаются все неопределённые переходы из множества состояний **S** = {I, M, B, F}. Для полностью определённого КА необходимо E замкнуть само на себя переходами a | b и кроме того, на E замкнуть переход из F по a | b.

Состояния	**Функция δ**	
	a	**b**
I	E	B
B	M	E
M	E	F
F	E	E
E	E	E

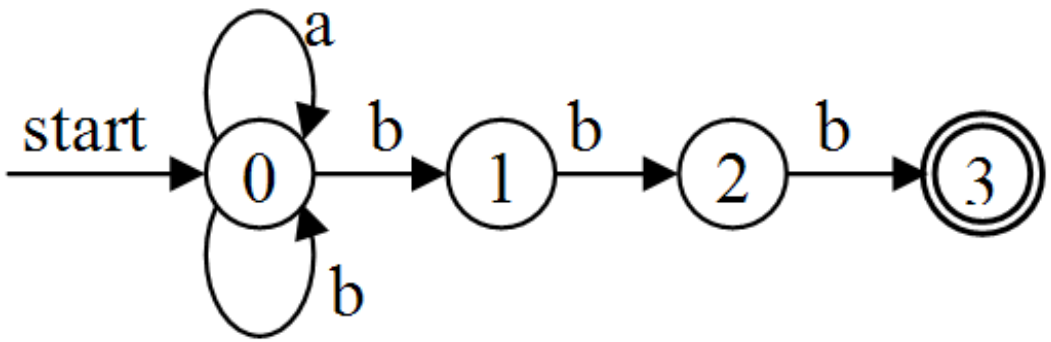
Синтаксические диаграммы и конечные автоматы

Полностью определенный КА:



Синтаксические диаграммы и конечные автоматы

Пример. Пусть КА задан таблицей:



Состояния	**Функция δ**	
	a	**b**
0	{0}	{0,1}
1	–	{2}
2	–	{3}

$L(A_2)$ = {(a|b)*bbb}.

Все допустимые цепочки можно показать с помощью так называемых перемещений (move):

b b b

$0 \to 1 \to 2 \to 3$

a b b

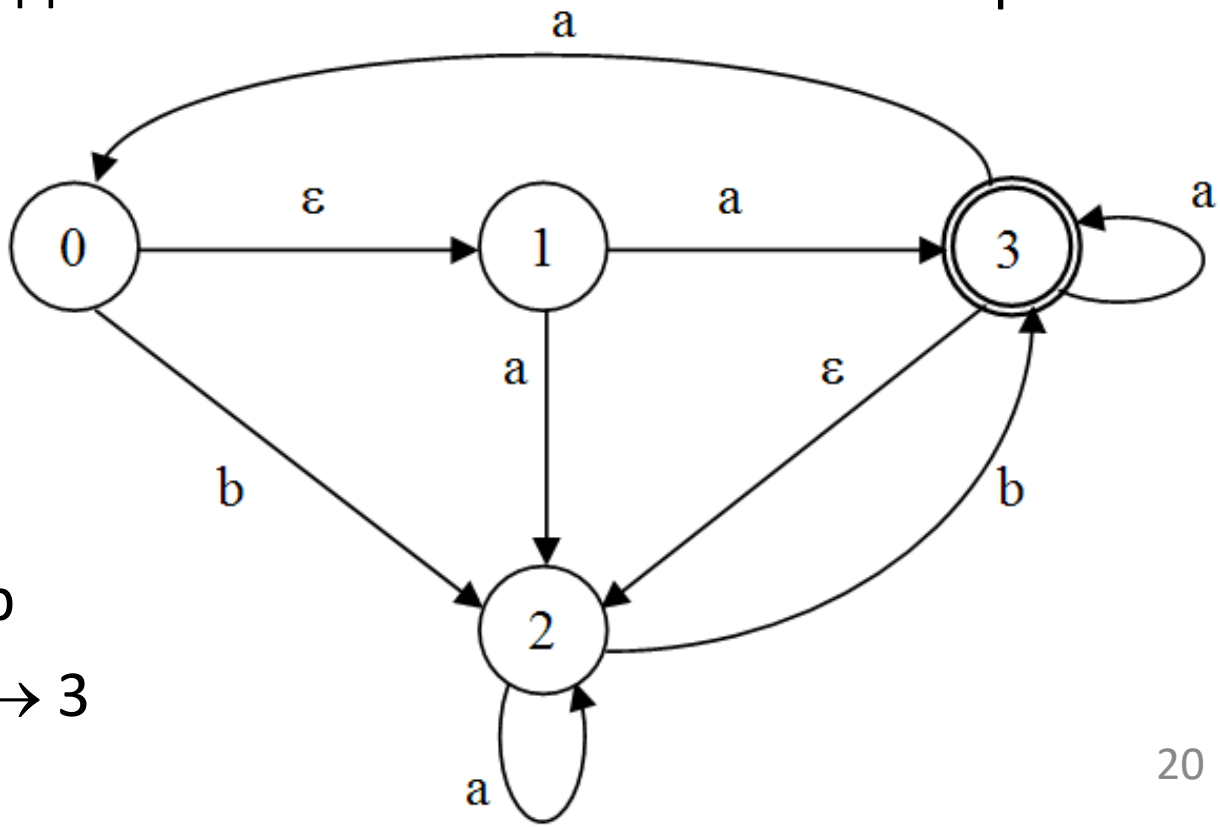
$0 \to 0 \to 1 \to 2$

Недетерминированные конечные автоматы (НКА)

Особенности: либо существует переход по $\varepsilon \in \Sigma$, либо из одного состояния выходят несколько переходов, помеченных одним и тем же символом из словаря Σ.

Пример.

НКА $\mathbf{N}_\varepsilon$



ω = abb

ε a b b

$0 \to 1 \to 2 \to 2 \to 3$

Недетерминированные конечные автоматы (НКА)

Определение. Недетерминированным Конечным Автоматом **N** (Nondeterministic finite automation) называется пятёрка **N = (S, Σ, I**, δ, **F),** где

S – конечное непустое множество (состояний);

Σ – конечное непустое множество входных символов (входной словарь);

I $\subseteq$ **S** – множество начальных состояний;

δ : **S x (Σ** $\cup$ ε) $\rightarrow$ 2^{S} – функция переходов. Здесь 2^{X} – обозначен булеан X, т.е. множество всех подмножеств множества X;

F $\subseteq$ **S** –множество заключительных состояний.

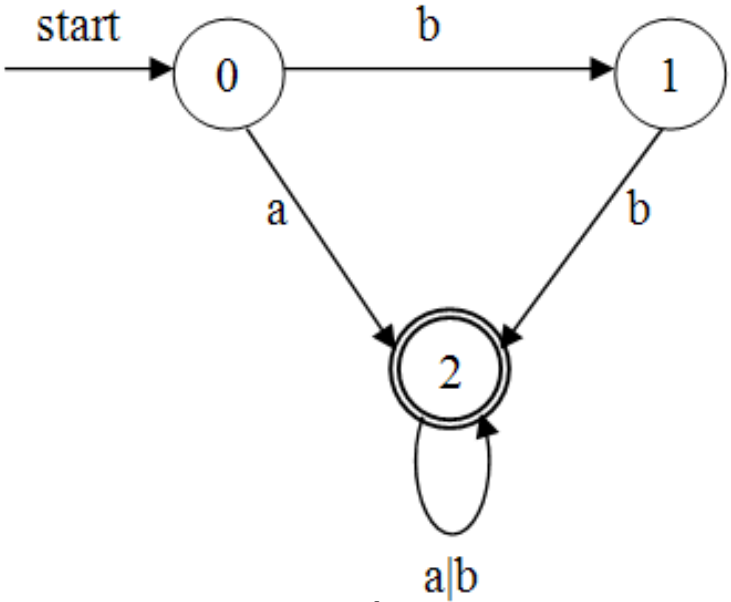
Недетерминированные конечные автоматы (НКА)

Определение. Недетерминированный конечный автомат **N** распознаёт входную цепочку ω, если существует путь, помеченный символами цепочки ω из начального в одно из заключительных состояний автомата возможно, с учётом ε-переходов. Недетерминированный КА распознаёт язык **L(N)**, если он распознаёт все цепочки этого языка.

Детерминированный конечный автомат (ДКА)

Пример. Автомат D, допускающий тот же язык **L(D)**, что и недетерминированный автомат $\mathbf{N}_{\varepsilon}$.

D:



Особенности **D**:

- отсутствие ε-переходов;
- для каждого состояния s и входного символа t существует не более одной дуги, исходящей из s, помеченной как t.

Детерминированный конечный автомат (ДКА)

Определение. Два автомата $\mathbf{A_1} = (\mathbf{S_1}, \mathbf{\Sigma_1}, \delta_1, s_1, \mathbf{F_1})$ и $\mathbf{A_2} = (\mathbf{S_2}, \mathbf{\Sigma_2}, \delta_2, s_2, \mathbf{F_2})$ называются эквивалентными, если они распознают один и тот же язык.

Определение. Конечный автомат $\mathbf{D} = (\mathbf{S}, \mathbf{\Sigma}, \delta, s_0, \mathbf{F})$ называется детерминированным конечным автоматом (Deterministic finit automation), если в каждом его состоянии $s \in \mathbf{S}$ функции переходов $\delta(s, a)$, для любого входного символа $\forall\ a \in \mathbf{\Sigma}$ содержит не более одного состояния, т.е.

$\forall\ a \in \mathbf{\Sigma}, \forall\ s \in \mathbf{S}: \delta(a, s) = \{r\}, r \in \mathbf{S}$

или $\forall\ a \in \mathbf{\Sigma}, \forall\ s \in \mathbf{S}: \delta(a, s) = 0.$

Детерминированный конечный автомат (ДКА)

Пример.

Входные данные: входная строка string, завершаемая символом конца файла EOF (End Of File);

D: стартовое состояние $s_0 \in S$, множество заключительных состояний F.

Выходные данные: true – если D допускает (принимает) x,

false – в противном случае.

Используемые функции:

move(s, c) – возвращает состояние, в которое переходит **D** при входном символе c;

getchar( ) – возвращает очередной сканируемый символ строки string.

Детерминированный конечный автомат (ДКА)

```
void()
{
 s = s0; /*стартовое состояние (начало «путешествия» по графу)*/
 c = getchar(string); /*сканирование очередного символа*/
 while(c != EOF)
 {
  s  = move(s, c); /*переход в новое состояние под управлением «c»*/
  c = getchar(string);
 }
 if (s ∈ F)
  return (true);  /*переход в заключительное состояние*/
 else
  return (false);
}
```

Детерминированный конечный автомат (ДКА)

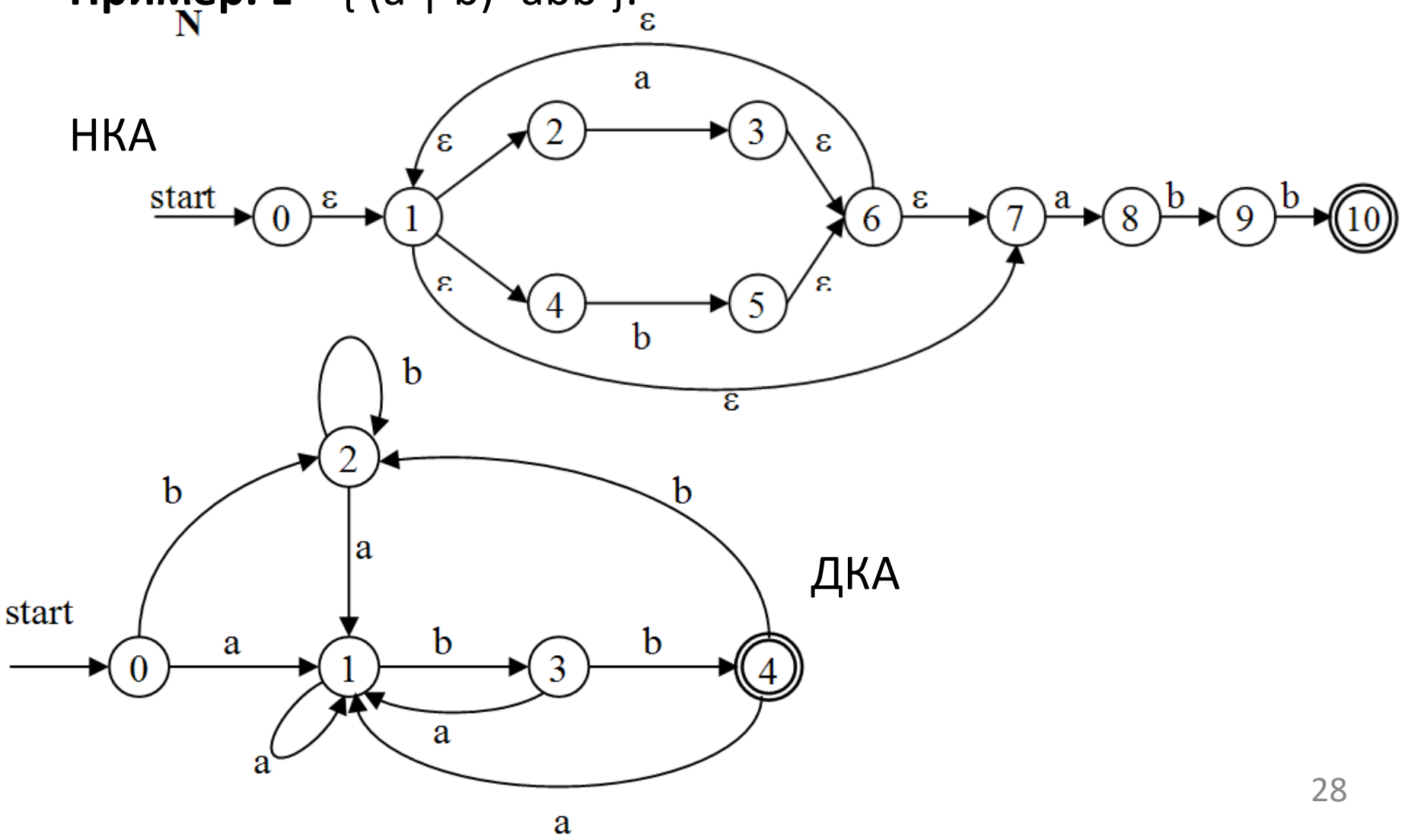
Теорема. Для любого недетерминированного конечного автомата существует эквивалентный ему детерминированный конечный автомат.

Определение. ε-замыкание состояния s (ε-closure) $\Xi(s)$ называется множество состояний недетерминированного конечного автомата **N**, достижимых из состояния $s \in \mathbf{S}$ только по ε-переходам.

Определение. ε-замыканием множества состояний $\mathbf{T} \subseteq \mathbf{S}$ называется множество $\Xi(\mathbf{T})$ состояний недетерминированного конечного автомата **N**, достижимых из каждого s из **T** ($\forall s \in \mathbf{T}$).

Переход от НКА к ДКА

Пример. L = { (a | b)*abb }.



Моделирование НКА

НКА: **N** = (**S**, **Σ**, s_0, δ, **F**)

```
N(…) /* Процедура моделирования НКА*/
{
 S = Ξ(s0); /* Определение ε-замыканий состояния s0 */
 c = getchar();  /* Сканирование первого символа из x*/
 while (c ≠ eof)
 {
  S = Ξ(M(S));
  c = getchar();
 }
 if (S ∩ F ≠ ∅)
  return (true);  /* N допускает цепочку x ∈ L(N) */
 else return (false); /* x ∉ L(N) */
}
```

Затраты на реализацию алгоритма составляют O(|x| * n), где |x| - длина входной цепочки; n - количество состояний.

Минимизация ДКА

Определение. Два состояния p и q из множества **PQ** множества состояний **S**, называются различными, если $\delta(p, c) \neq \delta(q, c)$, $c \in \Sigma$, $p, q \in \mathbf{PQ}$, $\mathbf{PQ} \subseteq \mathbf{S}$.

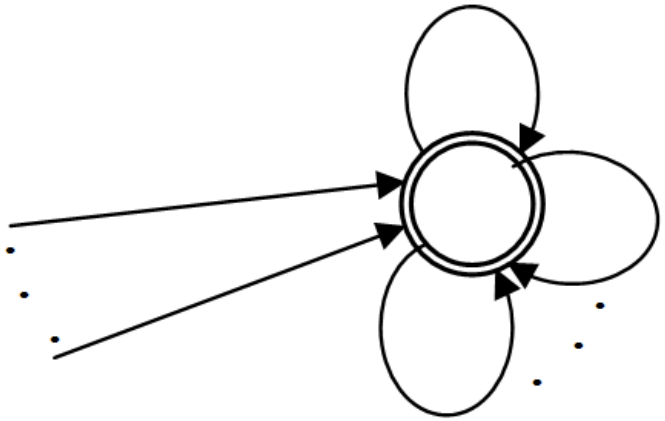
В противном случае состояния p и q являются неразличимыми.

Другими словами, если входную ленту с цепочкой ω и указателем n($c = (\omega, n)$, $c \in \Sigma$) подать на вход D, то проходя через подмножество различных состояний $\mathbf{PQ} \subseteq \mathbf{S}$, автомат всегда будет иметь разные конфигурации (S, C), $\mathbf{S} \in \mathbf{PQ}$.

Минимизация ДКА

Определение. Состояние называется «мертвым», если не являясь заключительным, оно для всех входных символов может иметь только переходы в себя, т. е.

$$\delta(q, c) = \mathbf{P}, \forall\ c \in \mathbf{\Sigma}, \mathbf{P} = \{p = q, \varnothing\}.$$



Определение. Недостижимыми называются все те состояния $\{q\}$, которые не достижимы из s_0

$$\delta(s_0, \omega) \neq \{q\}, \omega \in \mathbf{\Sigma}^*.$$

Минимизация ДКА

Алгоритм минимизации

Исходные данные: детерминированный автомат

$D = \{\mathbf{S}, \delta, s_0, \mathbf{F}, \Sigma\}$.

Необходимо получить: $D` = \{\mathbf{Q}, \delta`, q_0, \mathbf{E}, \Sigma\}$.

1. Удалить все “мёртвые” и недостижимые состояния и получить $\overline{S}$
2. Выполнить разбиение $\overline{S}$ на два подмножества $\overline{S} = \overline{S} - \mathbf{F}$ и $\mathbf{F}$.
3. Провести разбиение множеств $\overline{S}$ и $\mathbf{F}$ на подмножества $\mathbf{S_1 \ldots S_k}$, $\mathbf{F_1 \ldots F_m}$ ($k=1, 2,\ldots$; $m=1, 2\ldots$) причем все подмножества $\mathbf{S_i}$ ($i=\overline{1,k}$) таковые, что выключают только различимые состояния. В худшем случае может оказаться так, что $\mathbf{S_1}=\{s_1\}$, … $\mathbf{S_k}=\{s_k\}$, т.е. каждое подмножество разбиений множества $\mathbf{S}$ состоит из единственного состояния $s \in \mathbf{S}$. В этом случае оптимальным является исходный автомат и перейти к п. 5.

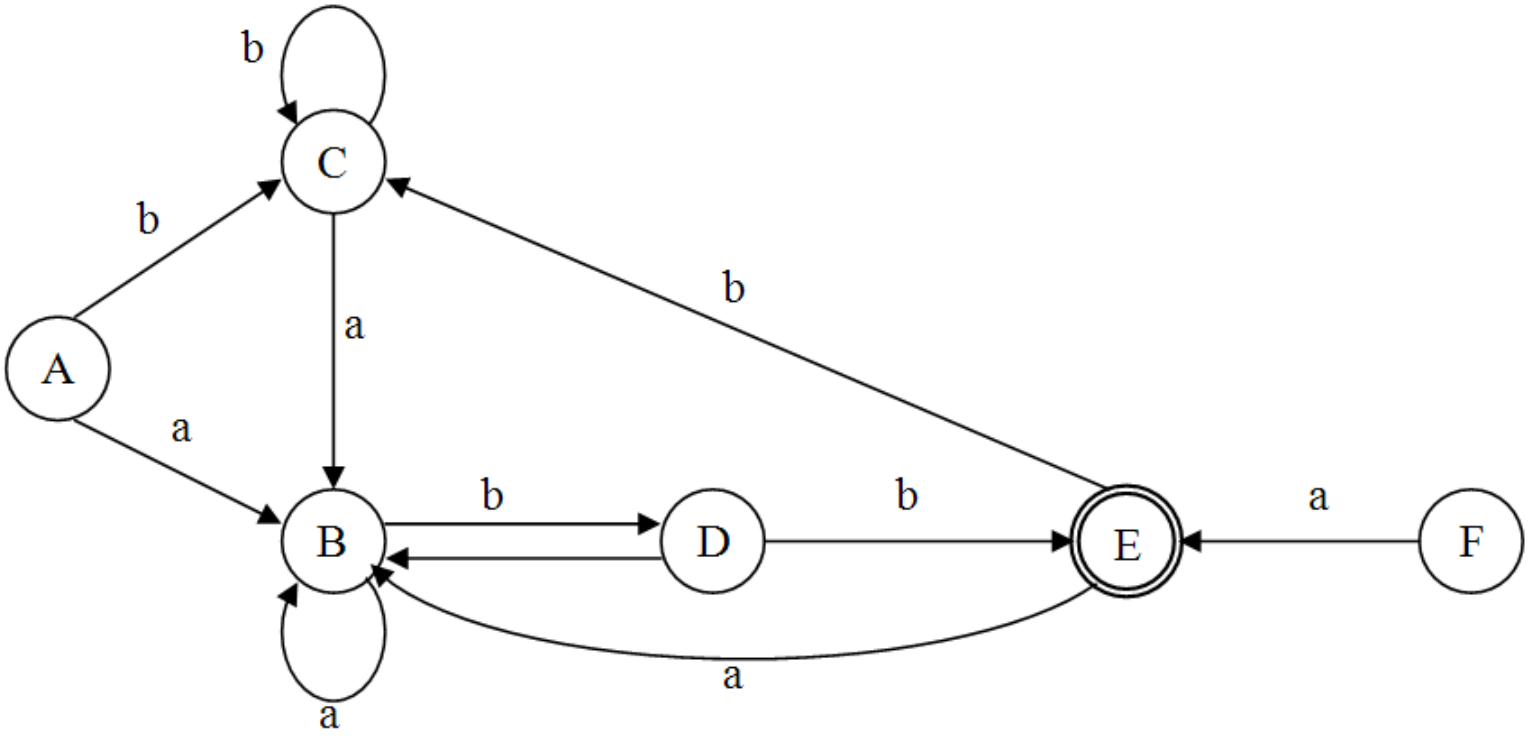
Минимизация ДКА

Алгоритм минимизации

4. Выбрать по одному состоянию из всех подмножеств $\mathbf{S_i}(i=\overline{1,k})$, в качестве представителя нового состояния в D`. Пусть $s \in \mathbf{S_i}$ $(i=\overline{1,k})$ – является представителем нового состояния в подмножестве $\mathbf{S_i}$. Предположим, что для входного символа $c \in \Sigma$ существует переход $\delta(s, c) = t, \forall c$, причём $t \in \mathbf{S_j}$ $(i \neq j, 1 \leq j \leq k)$. Тогда t сделать представителем нового состояния для подмножества различимых состояний $\mathbf{S_j}$.
5. Стартовым q_0 сделать то подмножество состояний $\mathbf{S_r}$, куда входит $\mathbf{S_0}$ $(q_0=S_0)$. Заключительным q_F сделать любое состояние из подмножества заключительных **F**.

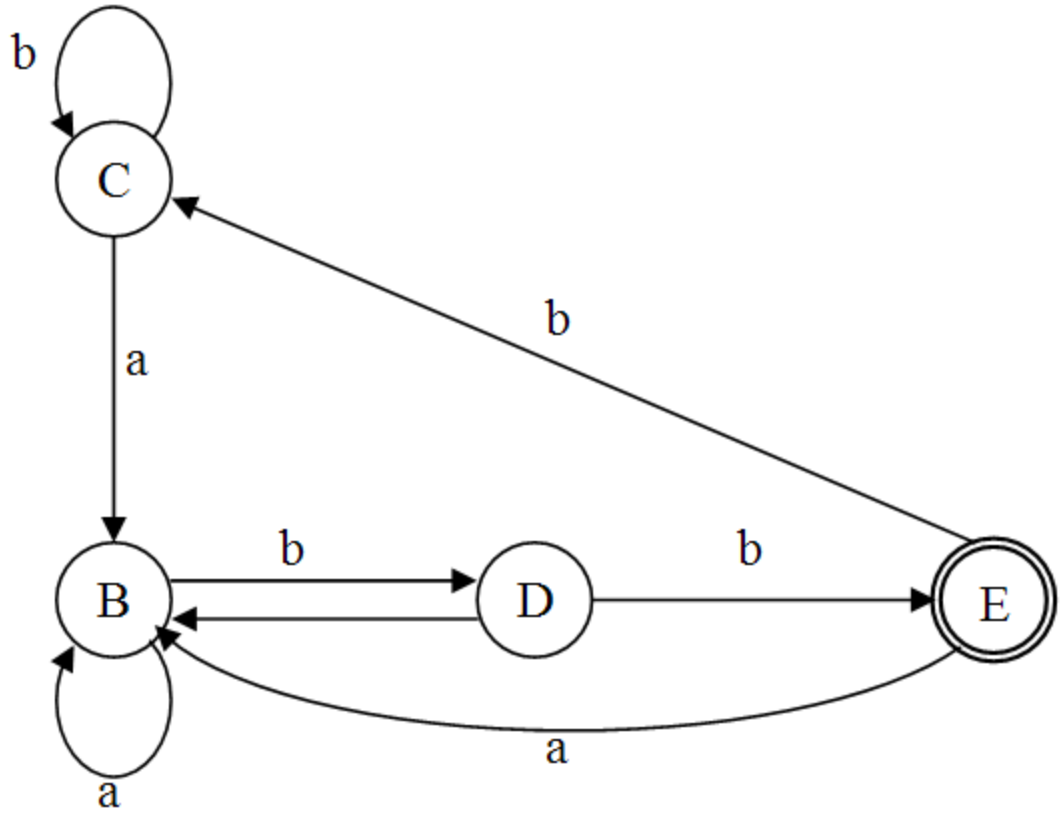
Минимизация ДКА

Пример. Исходный неоптимальный ДКА



Минимизация ДКА

Эквивалентный оптимальный (минимальный) ДКА



МП-автоматы

Пример. Пусть цепочка некоторого языка **L** имеет вид:

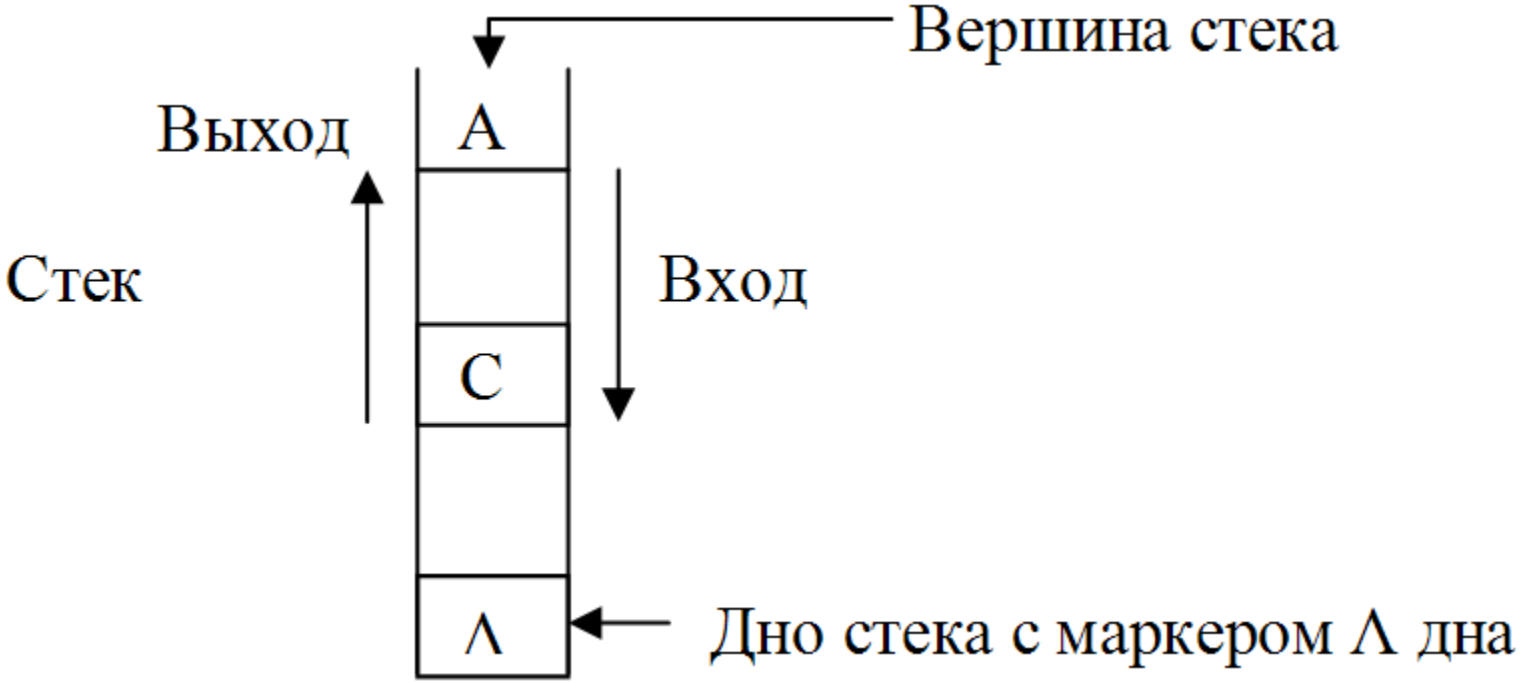
$$\mathbf{L} = \{w^n \alpha w_1^n \mid w \in \mathbf{W}, \alpha \in \mathbf{V_t}^*\}, \text{ где}$$

- w – открывающая круглая скобка.
- w_1 – закрывающая круглая скобка.
- α – выражение без скобок.

Условие баланса скобок сразу ограничивает применение схем ДКА и НКА к моделированию таких цепочек.

В МП-автоматах для возможности реализации подобных проблем вводится дополнительная память (магазин или стек) для хранения предыстории.

МП-автоматы



МП-автоматы

Определение. МП-автомат – это семёрка

$$P=(\mathbf{S}, \mathbf{\Sigma}, \mathbf{M}, \delta, s_0, Z_0, \mathbf{F}),$$ где

- **S** – конечное множество состояний;
- **Σ** – конечный входной алфавит;
- **M** – конечный алфавит магазинных символов;
- δ – функция перехода, отображение множества $\mathbf{S}\times(\mathbf{\Sigma} \cup \{\varepsilon\})\times\mathbf{M}$ на множество конечных подмножеств множества $\mathbf{S}\times\mathbf{M}^*$;
- $s_0 \in \mathbf{S}$ – начальное состояние управляющего устройства;
- $Z_0 \in \mathbf{M}$ – символ, находящийся в магазине в начальный момент времени (начальный символ);
- $\mathbf{F} \subseteq \mathbf{S}$ – множество заключительных состояний.

МП-автоматы

Определение. Конфигурацией МП-автомата P называется тройка $(s,w,\alpha) \in \mathbf{S}\times\mathbf{\Sigma}^*\times\mathbf{\Gamma}^*$, где

- s – текущее состояние управляющего устройства,
- w – неиспользованная часть входной цепочки (если $w=\varepsilon$, то считается, что вся входная цепочка прочитана),
- α – содержимое магазина (самый левый символ цепочки α считается верхним символом магазина; если $\alpha=\varepsilon$, то магазин считается пустым).

МП-автоматы

Пример. $\mathbf{L(G_1)}=\{0^n1^n \mid n\geq 0\}$.

Пусть **P**=($\{s_0, s_1, s_2\}$, {0,1,e}, {1,Λ}, δ, s_0, Λ=ε, $\{s_0\}$), где:

- $\delta(s_0, 0, \varepsilon) = \{(s_1, 0\varepsilon)\}$
- $\delta(s_1, 0, 0) = \{(s_1, 00\varepsilon)\}$
- $\delta(s_1, 1, 0) = \{(s_2, 0\varepsilon)\}$
- $\delta(s_2, 1, 0) = \{(s_2, \varepsilon)\}$
- $\delta(s_3, e, \varepsilon) = \{(s_0, \varepsilon)\}$

Работа автомата заключается в копировании в магазин начальных нулей из входной цепочки и последующем устранении по одному нулю из магазина на каждую прочитанную единицу. Таким образом, дополнительная память (стек) должна обеспечить запоминание количества записанных нулей и затем отследить такое же количество единиц во входной цепочке.

Детерминированные МП-автоматы

Механизм управления МП-автоматом через δ(s, α, c) удобно иллюстрировать управляющими таблицами.

S

s_0 :	add s_1 n++	error	true	ε
s_1 :	add s_1 n++	del s_2 n++	false	ε
s_2 :	add s_1 n++	del s_2 n++	false	ε
	false	false	true s_0	ε

M

Детерминированные МП-автоматы

Пример. Состояние входной ленты (цепочки), стек и состояния P при обработке цепочки 0011e от начального до конечного символа.

0.	ε	$[s_0]$	0011e	
1.	ε0	$[s_1]$	011e	
2.	ε00	$[s_1]$	11e	
3.	ε0	$[s_2]$	1e	
4.	ε	$[s_2]$	e	
5.	ε	$[s_0]$	e	true

Детерминированные МП-автоматы

Определим итерацию функции переходов δ^* для

P=(**S**, **Σ**, **M**, δ, s_0, Z_0, **F**) как отображение декартова произведения множества состояний (**S**) на итерации множества словаря (ε^*) и множества алфавита магазинных символов ($\mathbf{M}^*$), в декартово произведение **S** и $\mathbf{M}^*$, то есть

$$\delta^*: \mathbf{S}\times\mathbf{\Sigma}^*\times\mathbf{M}^* \rightarrow \mathbf{S}\times\mathbf{M}^*$$

Детерминированные МП-автоматы

Определение. Конечный МП-автомат $P=(\mathbf{S}, \mathbf{\Sigma}, \mathbf{M}, \delta, s_0, Z_0, \mathbf{F})$ допускает входную цепочку $\alpha \in \mathbf{\Sigma}^*$, если α переводит P из начального (s_0) состояния с начальным магазинным символом в памяти (стеке) Z_0 в одно из заключительных состояний **F**, притом, что магазинная память в результате обработки α вновь стала Z_0.

$$\mathbf{L}(\mathbf{P}) = \{\alpha \mid \alpha\, \mathbf{\Sigma}^*, \delta^*(s_0, \alpha, Z_0)\ (\mathbf{F}\times Z_0)\}$$

Детерминированные МП-автоматы

Определение. МП-автомат **P**=(**S**, **Σ**, **M**, δ, s_0, Z_0, **F**) называется детерминированным, если для каждых $s \in$ **S** и $Z \in$ **Г** верно одно из следующих утверждений:

- δ(s, α, Z) содержит не более одного элемента для каждого $\alpha \in$ **Σ** и δ(s, ε, Z)=0;
- δ(s, α, Z)=0 для всех $\alpha \in$ **Σ** и δ(s, ε, Z) содержит не более одного элемента.