# Построение ДКА

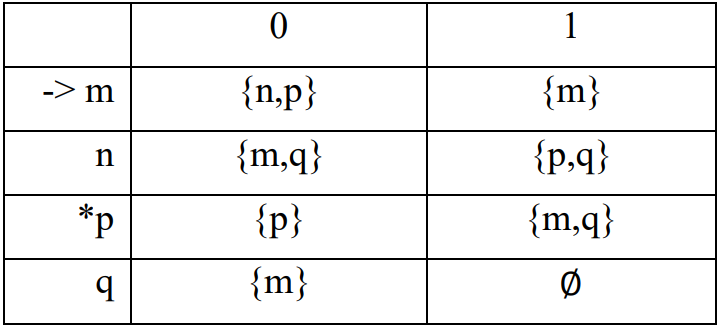
# НКА ДКА

“Ленивый алгоритм”

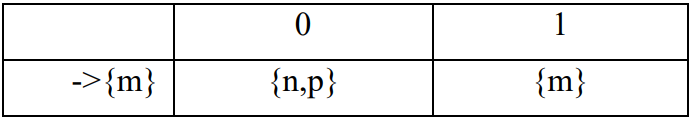
Пример

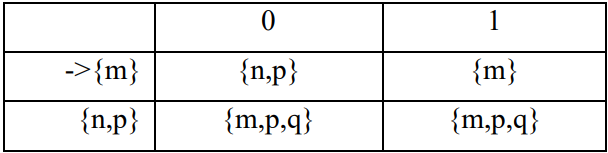
НКА:

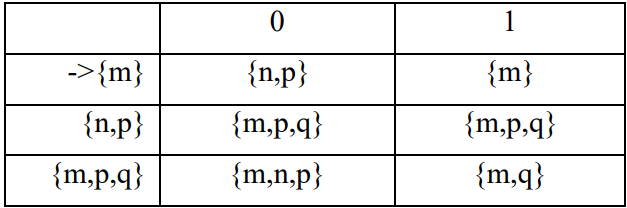


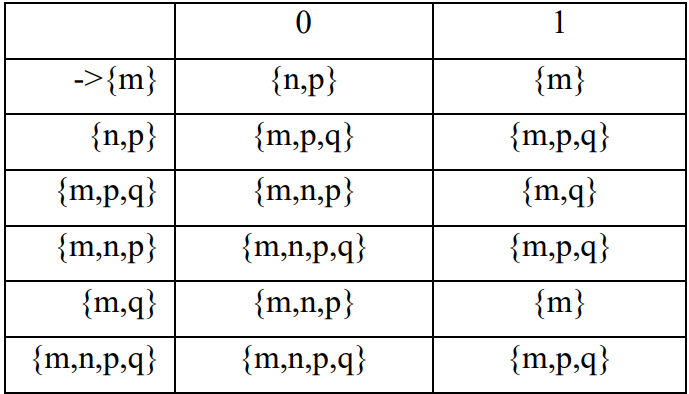


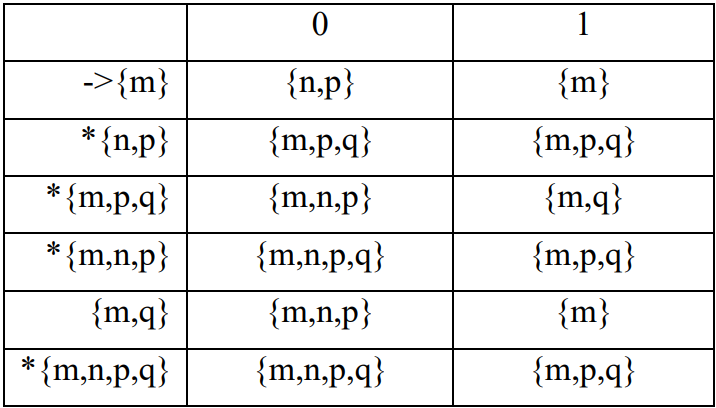
ДКА:







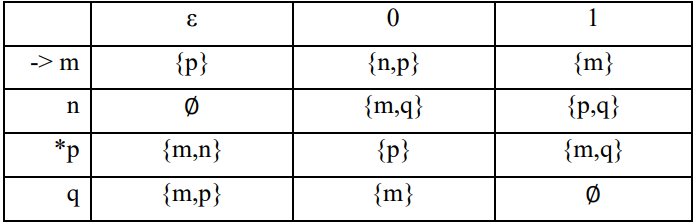




# ε-НКА ДКА

ε-НКА:





ДКА

Расписываем ECLOSE() от каждого состояния.

ECLOSE(m) = {p, m, n}

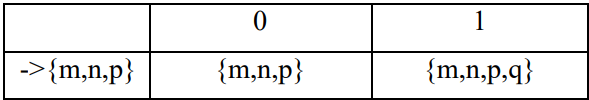
ECLOSE(n) = {n} (!)

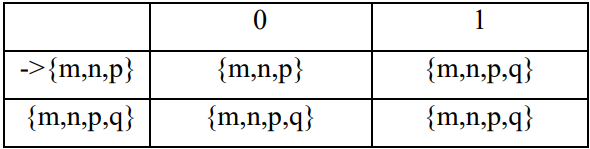
ECLOSE(p) = {m, n, p}

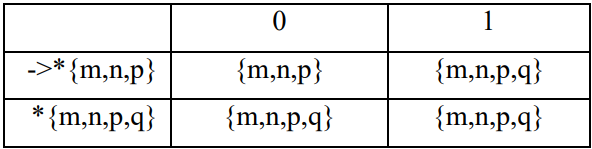
ECLOSE(q) = {m, p, n,q}

Пример:

1. Вычисляем ECLOSE от текущего состояние
2. Находим множество состояний, в которое переходим по входящему символу и множеству состояний, полученному на 1-ом шаге.
3. Находим ECLOSE от множества, полученного на 2-ом шаге.







# РВ

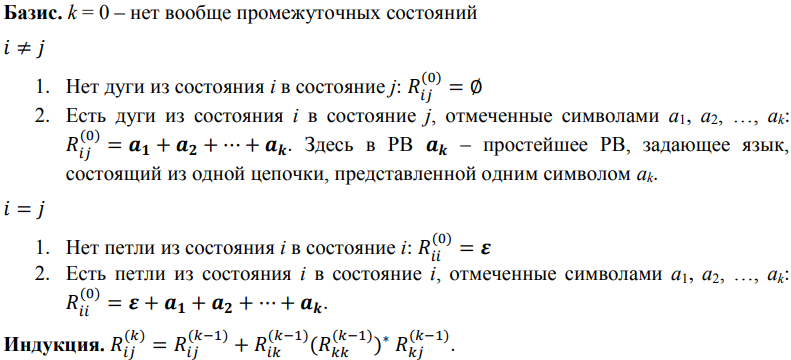
Построить РВ для множества цепочек над алфавитом {a,b,c}, содержащих хотя бы один символ a и хотя бы один символ b.

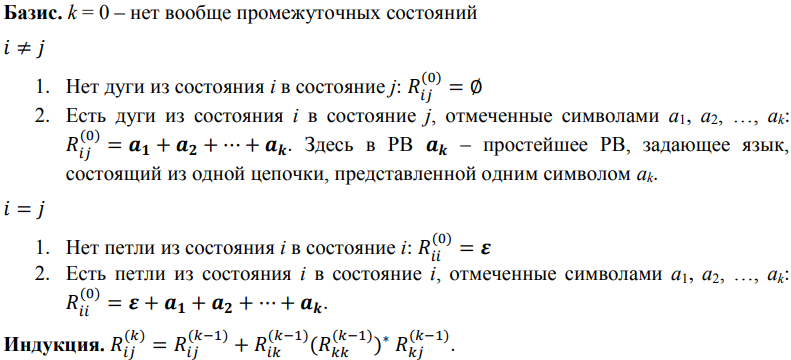


# РВ ДКА

С помощью регулярных выражений , языком которых являются цепочки, получаемые при работе ДКА при его переходе из состояния i в состояние j и не проходящих через промежуточные состояния с номерами больше k.

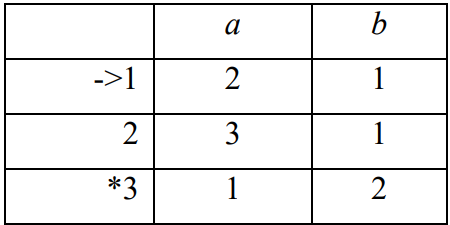
Задается с помощью индукции.



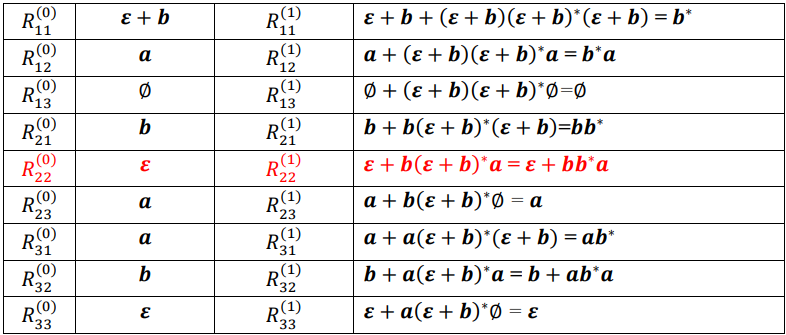
 ВАЖНО!!!

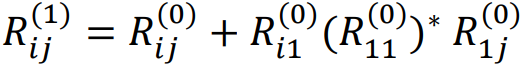
**Пример**:

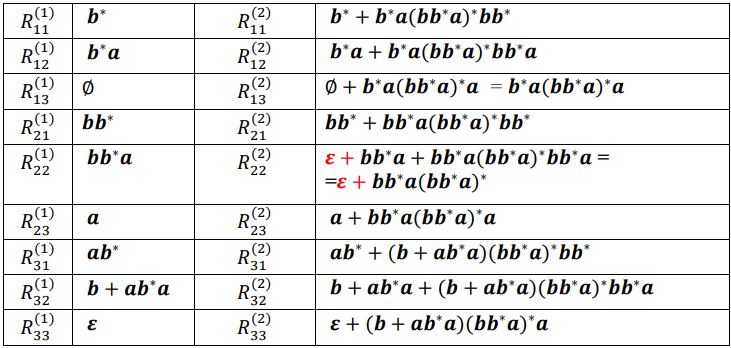
ДКА:

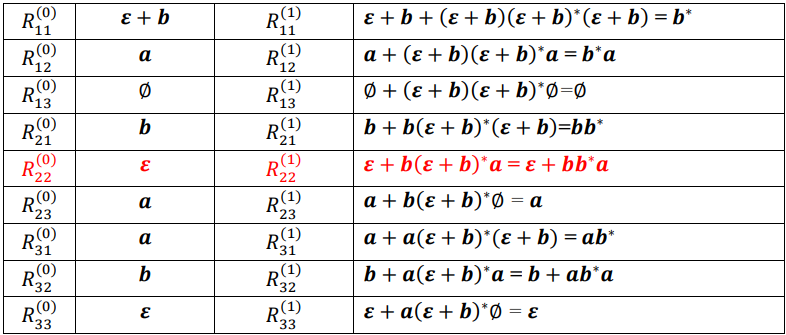


РВ:

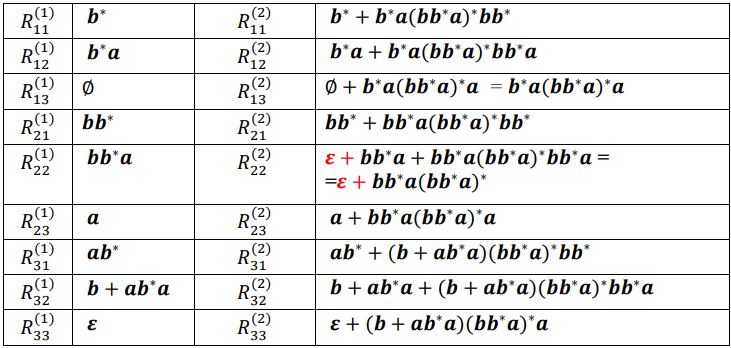


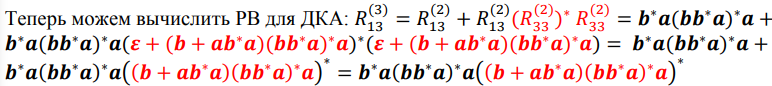






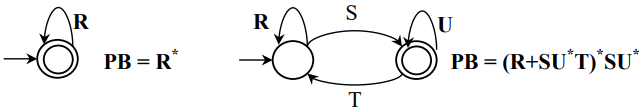




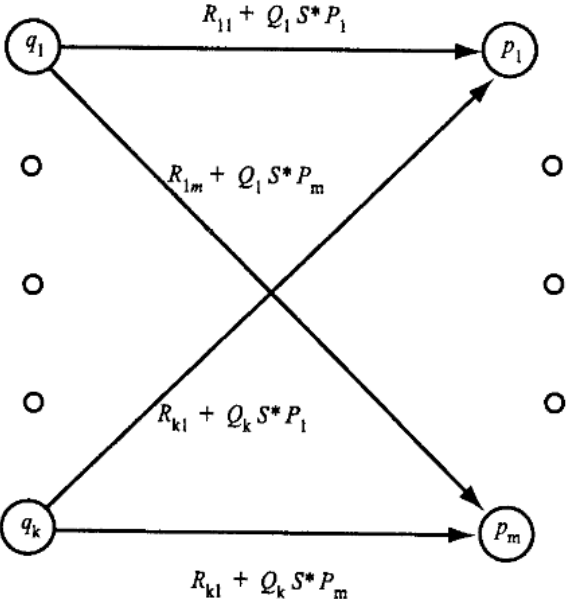
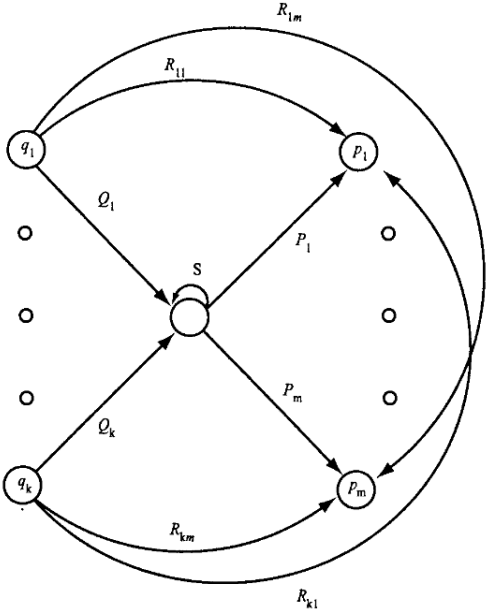


# ДКА РВ (исключение состояний)

Метод исключения состояний приводит автомат к одному из двух видов, для которых можно сразу же написать РВ:

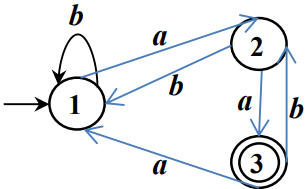
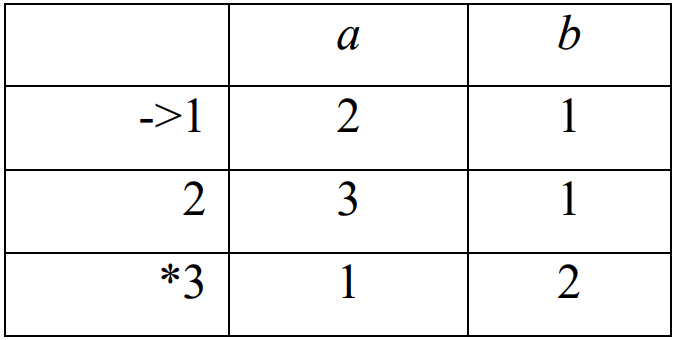
 ВАЖНО!!!

**Удаление состояния:**



**Пример:**

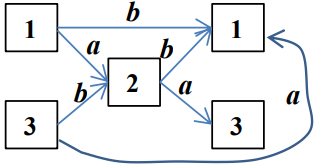
ДКА:



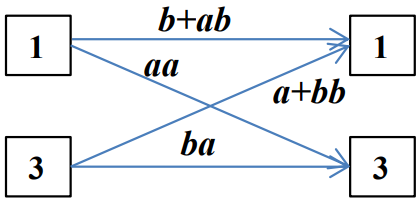
Удаляем второе состояние.

Рисуем схему: из каких состояний попадаем в это состояние и в какие состояния попадаем из него.

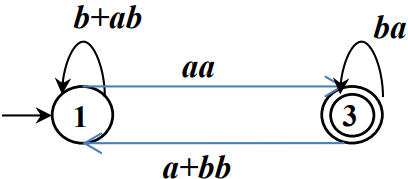
1 Дублируем состояния, так что один дубликат соответствует выходящим дугам, а другой входящим



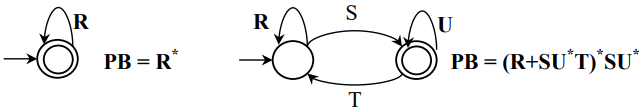
2 Удаляем состояние 2, учитывая все возможные переходы в это состояние и из этого состояния.



3 Переходим к ДКА



4 Строим РВ по ДКА учитывая маркировки(2 формулы ниже)

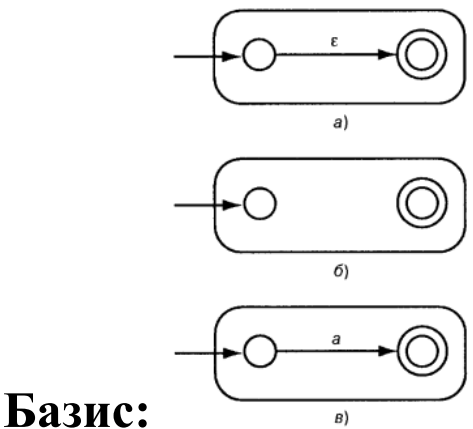




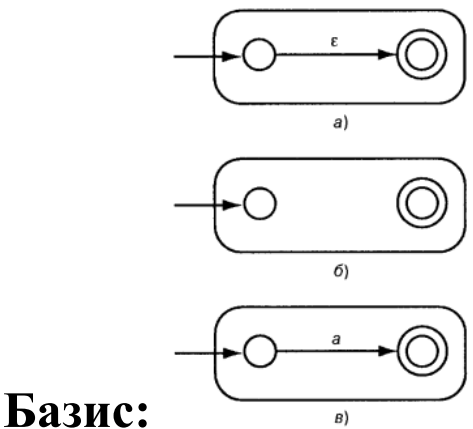
# РВ в ε-НКА

Базис:

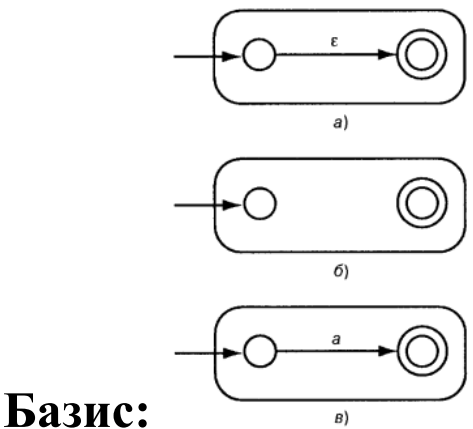
а) -переход



б) нет перехода

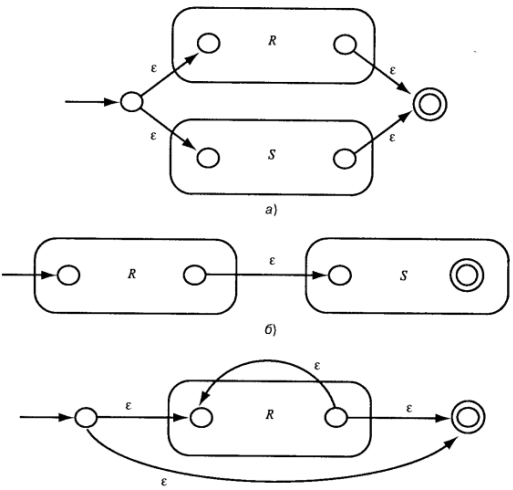


в) Символ для РВ (т.е. в ε-НКА мы строим конструкцию, которая ниже)

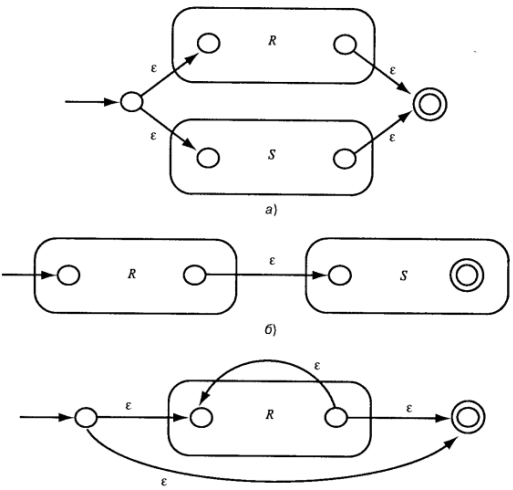


Индукция:

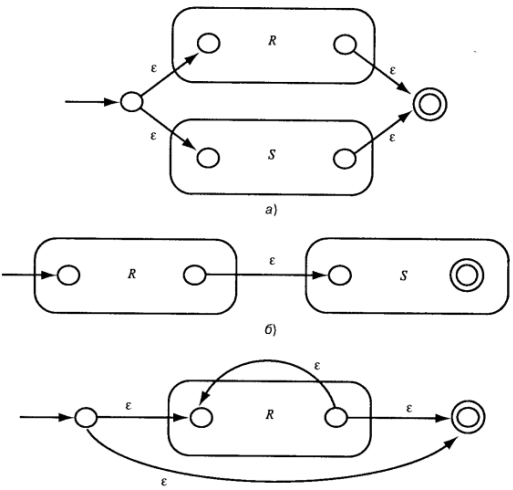
а) Оператор +

ВАЖНО!!!

б) Конкатенация

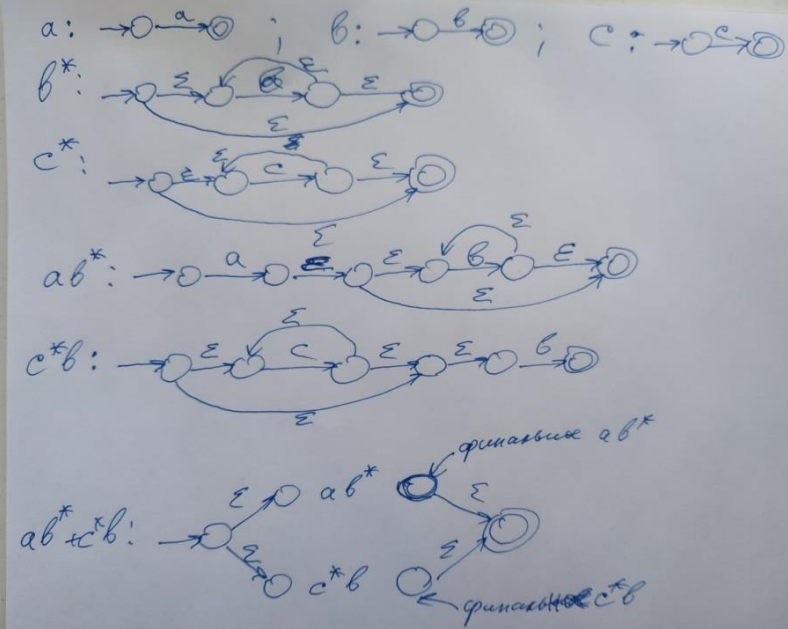
ВАЖНО!!!

в) Оператор \*

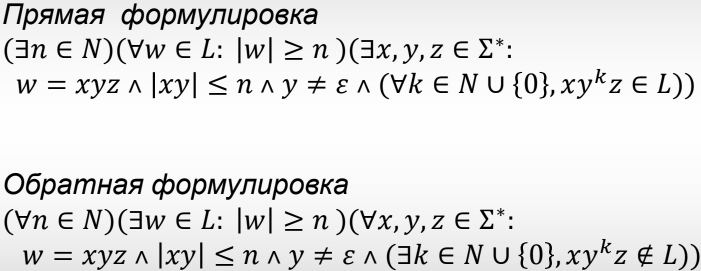
ВАЖНО!!!

Суть метода заключается в построении отдельных частей автомата по РВ и постепенная “сборка” автомата из этих частей.

**Пример**: ab\*+c\*b



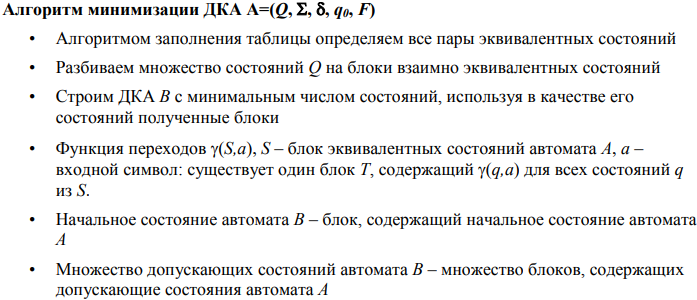
# Определение нерегулярности языка (лемма о накачке для регулярного языка) (разобрать практики)

ВАЖНО!!!

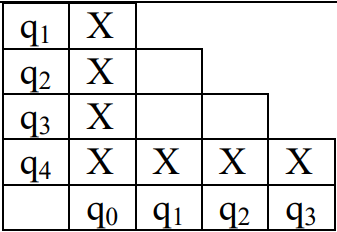
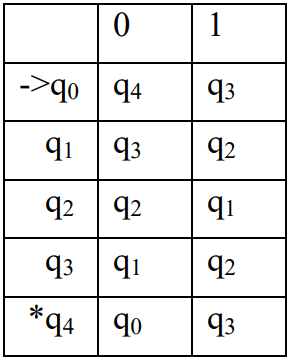
Пример:

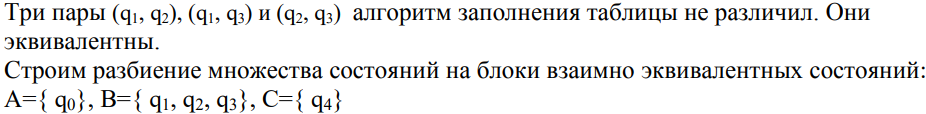
1. Показать, что язык {: 𝑛 ≥ 1} нерегулярен.
2. Задаем произвольный n.
3. Выберем цепочку 𝑤 =, |𝑤| ≥ 𝑛.
4. Разобьем эту цепочку на три подцепочки 𝑤 = 𝑥𝑦𝑧 такие, что |𝑥𝑦| ≤ 𝑛.
5. Тогда 𝑥𝑦 = , 𝑙 ≤ 𝑛, а 𝑧 = .
6. Так как 𝑦 ≠ 𝜀, то 𝑦 = , 1 ≤ 𝑚 ≤ 𝑙, а 𝑥 =.
7. Накачаем произвольное разбиение нашей цепочки при 𝑘 = 0.
8. Накаченная при любом разбиении цепочка будет иметь вид 𝑤 = 𝑥𝑧 == = .
9. Но 1 ≤ 𝑚, это означает, что 𝑛 − 𝑚 ≤ 𝑛. Следовательно, накачанная при любом разбиении, удовлетворяющем обратной формулировки леммы, цепочка не принадлежит языку L. (т.е. язык не является регулярным)

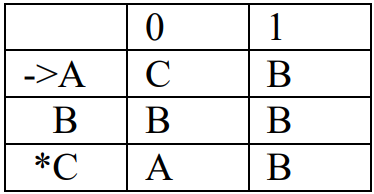
# Минимизация автомата



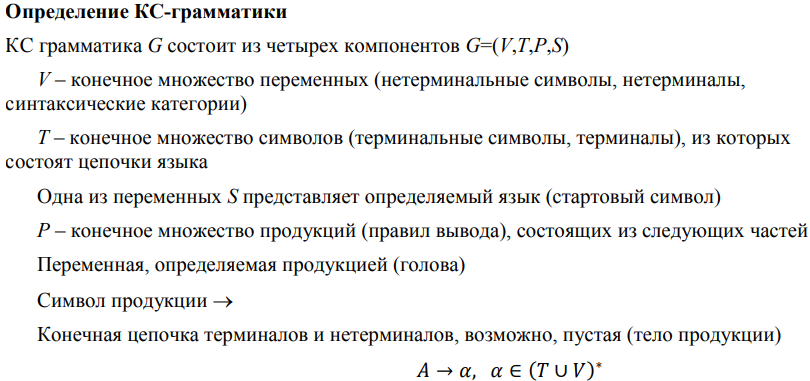
**Пример:**







# Построение КС-грамматик



Рекурсивный вывод: тело → голова

Порождение (вывод) цепочки в КС-грамматике: голова → тело

**Пример:**

Построить грамматику для языка { | 𝑛 ≥ 1}

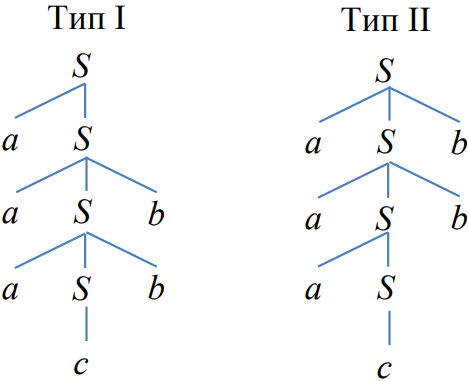
Формальное задание грамматики: 𝐺 = ({𝑆},{0,1},{𝑆 → 0𝑆1, 𝑆 → 01}, 𝑆)

# Устранение неоднозначности в КС-грамматиках

**Пример**

Рассмотрим грамматику: 𝐺 = ({𝑆},{𝑎, 𝑏, 𝑐},{𝑆 → 𝑎𝑆, 𝑆 → 𝑎𝑆𝑏, 𝑆 → 𝑐}, 𝑆). Показать, что она неоднозначна. Построить эквивалентную ей однозначную грамматику.

Следовательно, существует и два дерева разбора:



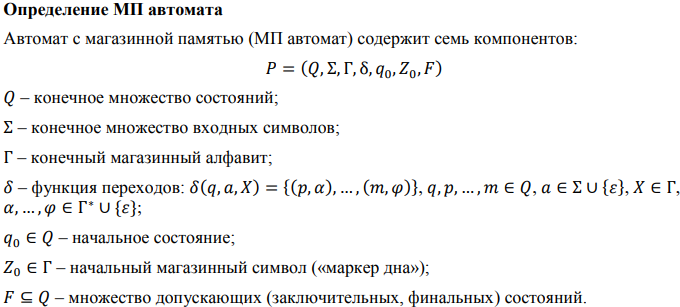
* Убираем неоднозначность (первый вариант): сначала набираем «дополнительные» символы a, а затем одинаковое количество символов a и b.

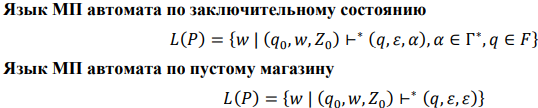
𝐺 = ({𝑆},{𝑎, 𝑏, 𝑐},{𝑆 → 𝑎𝑆|𝑇, 𝑇 → 𝑎𝑇𝑏, 𝑇 → 𝑐}, 𝑆)

* Убираем неоднозначность (второй вариант): сначала набираем одинаковое количество символов a и b, а затем «дополнительные» символы a.

𝐺 = ({𝑆},{𝑎, 𝑏, 𝑐},{𝑆 → 𝑎𝑆𝑏|𝑇, 𝑇 → 𝑎𝑇, 𝑇 → 𝑐}, 𝑆)

# МП-автоматы



ВАЖНО!!!

**Пример**

Построить МП автомат по заключительному состоянию для языка { | 𝑛 ≥ 1} и найти все конфигурации, в которых окажется автомат по прочтении цепочки 0011.

Построим автомат 𝑃 = ({𝑞, 𝑝,𝑡},{0,1},{𝑍, }, δ, 𝑞, ,{𝑡})

Функция перехода:

1. 𝛿(𝑞, 0, ) = {(𝑞, 𝑍)}

2. 𝛿(𝑞, 0, Z) = {(𝑞, 𝑍𝑍)}

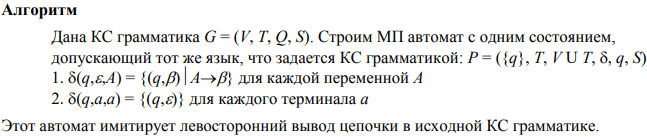
3. 𝛿(𝑞, 1, Z) = {(𝑝, 𝜀)}

4. 𝛿(𝑝, 1, Z) = {(𝑝, 𝜀)}

5. 𝛿(𝑝, 𝜀, ) = {(𝑡, )}

# КС-грамматики → МП автомат

Алгоритм

ВАЖНО!!!

**Пример**

Дана КС грамматика G = (V, T, Q, S): V = {F}, T = {a,b}, S = F , Q = {F → ab, F → aFb, F → FF}.

Построить МП автомат, допускающий этот язык по пустому стеку и породить какую-нибудь цепочку языка и найти переходы автомата (конфигурации) при обработке этой цепочки автоматом.

P = ({q}, {a,b}, {a,b,F}, , q, F)

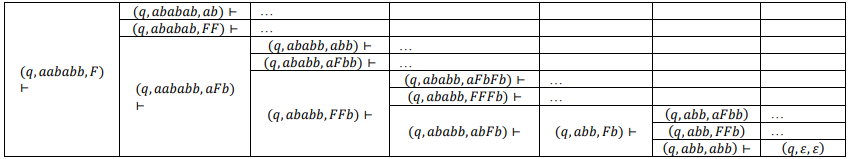
1. (q,,F) = {(q,ab), (q,aFb), (q,FF)}

2. (q,a,a) = {(q,)}

(q,b,b) = {(q,)}

Вывод цепочки aababb: 𝐹𝑎𝐹𝑏𝑎𝐹𝐹𝑏𝑎𝑎𝑏𝐹𝑏𝑎𝑎𝑏𝑎𝑏𝑏

Конфигурации автомата:



Шаг: -переход, удаление терминальных символов

# МП автомат → КС грамматика

Строим КС грамматику G=(V,Σ,R,S), для которой L(G)=N(P).

* V состоит из переменных

– стартовый символ S

– [pXq], где p,q Q, X Г(стек)

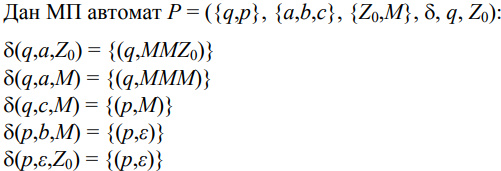
* R состоит из продукций:

– S[] для всех p Q.

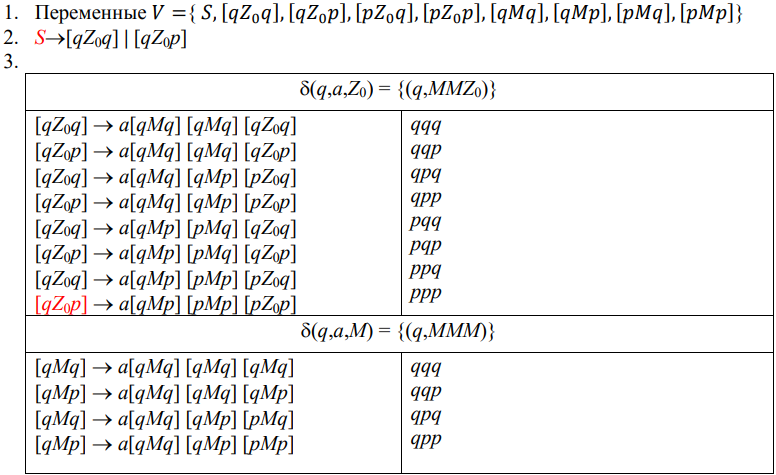
Символ вида [] предназначен для порождения всех цепочек w, которые приводят автомат к выталкиванию из магазина в процессе перехода из состояния в состояние p: (q,, ) (p,,)

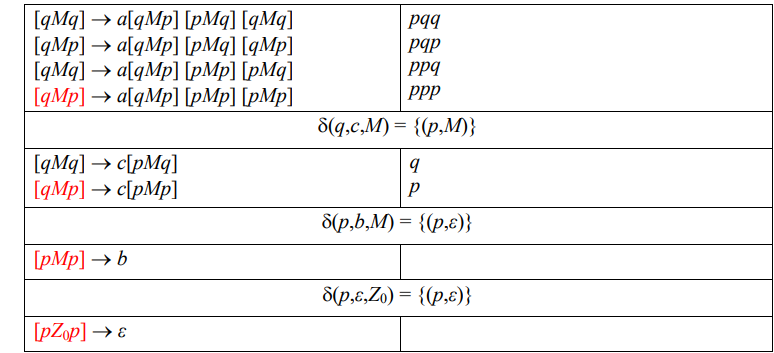
– если (q,a,X),то

**Пример**



Строим КС грамматику:



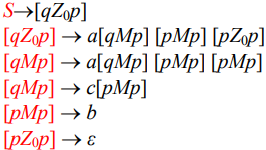


* Удаляем бесполезные символы.

Удаляем все правила с не порождающими символами.

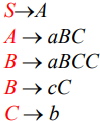
(Символ X называется *порождающим*, если для некоторой терминальной цепочки .)

Остаются:

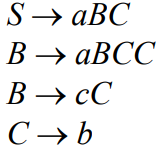


* Удалим ε-порождающий символ [pp].

Сделаем замену: [qp]=A, [qMp]=B, [pMp]=C. Получим грамматику:



* Удаляем цепное правило SA и получаем грамматику в нормальной форме Грейбах(вводим только один терминал)



# Нормальная форма Хомского КС грамматики

**Определение**

Нормальная форма Хомского (НФХ) грамматики для КС языка без -продукций и бесполезных символов:

ABC

Aa

Порядок применения преобразований:

1. Удалить -продукции

2. Удалить цепные продукции

3. Удалить бесполезные символы

**Определение**

Переменная A называется -порождающей, если A\*.

**Удаление -порождающих продукций**

Для продукции B, содержащей -порождающие переменные, строятся всевозможные их комбинации (по одному, по два и т.д. до n, где n - количество -порождающих переменных в продукции) и строятся продукции с исключенными комбинациями переменных. Нельзя только убирать все -порождающие переменные, если n = k.

**Удаление цепных продукций**

1. Находим все цепные пары грамматики G

2. Для каждой цепной пары (A, B) добавим в P все продукции A, где – тело не цепной продукции B

**Удалить бесполезные символы**

Определение

Символ X называется полезным в грамматике G=(V,T,P,S), если существует порождение вида .

1. Символ 𝑋 называется порождающим, если для некоторой терминальной цепочки 𝑤.

2. Символ 𝑋 называется достижимым, если существует порождение для некоторых 𝛼, 𝛽 ∈ (𝑉 ∪ 𝑇)\*.

**Вычисление порождающих символов**

**Базис**. Каждый символ из 𝑇 является порождающим, он порождает самого себя.

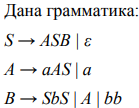
**Индукция**. Пусть существует продукция 𝐴, каждый символ в является порождающим. Тогда символ 𝐴 порождающий.

**Вычисление достижимых символов**

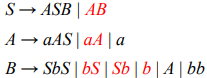
**Базис**. Символ 𝑆 достижим.

**Индукция**. Пусть некоторая переменная 𝐴 достижима. Тогда для всех продукций с головой 𝐴 все символы тел достижимы.

**Пример**.

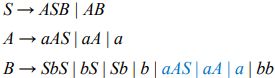


1. Удаляем ε-продукции: базис - S → ε, индукция не дает более ε-порождающих переменных. Грамматика без ε-продукций:



1. Удаляем цепные продукции: базис – (A, A), (B, B), (S, S)

индукция – B →A => (B, A) – цепная пара



1. Находим бесполезные символы.

* Сначала ищем порождающие символы:

**Базис**:

a и b

**Индукция**:

A → a и B → b, следовательно A и B – порождающие символы

S → AB, следовательно S – порождающий символ

Все символы порождающие.

* Ищем достижимые символы:

Базис:

S

Индукция:

S → AB, следовательно A и B – достижимые символы

B → aAS, следовательно a – достижимый символ

B → SbS, следовательно b – достижимый символ

Все символы достижимые.