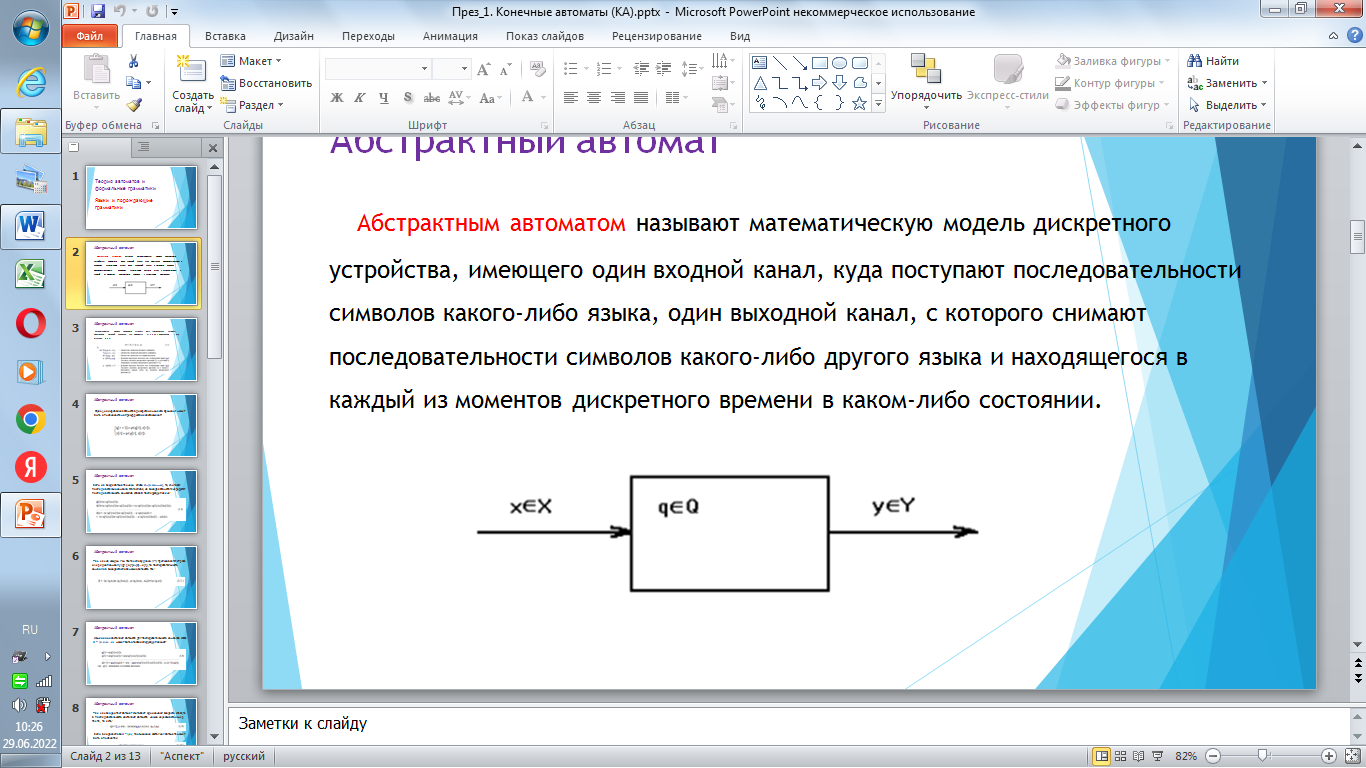
**Абстрактный автомат**



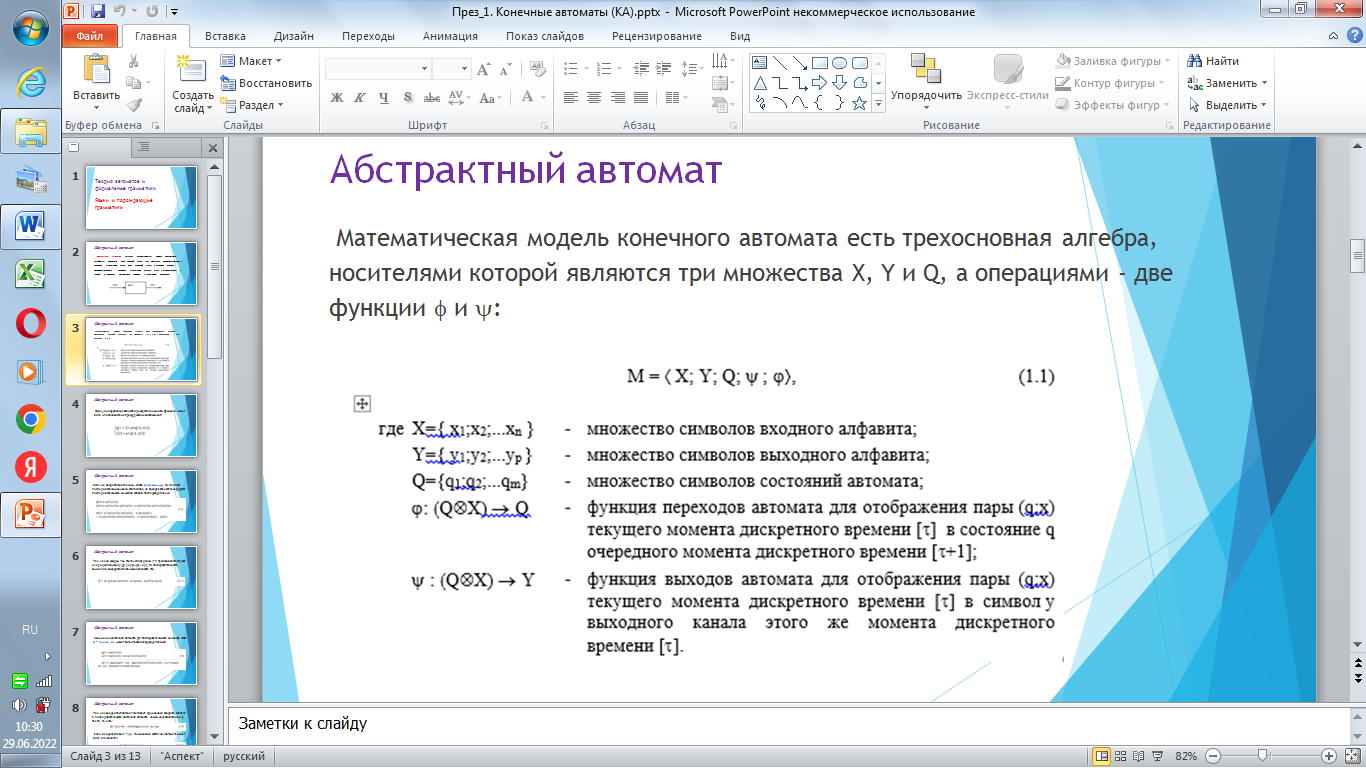
Абстрактный автомат – математическая модель дискретного устройства.

Х – канал входных символов (кнопочки, которые нажимают)

Q – состояния

Y – канал выходных символов (реакция на нажатые кнопочки)

Любой автомат задается 5кой



Где X – множество символов входного алфавита

Y – множество символов выходного алфавита

Q – множество состояний

ψ – функция выходов автомата (если мы находимся в таком-то состоянии и нажимаем такую кнопочку, то на выходе появится такой-то сигнал)

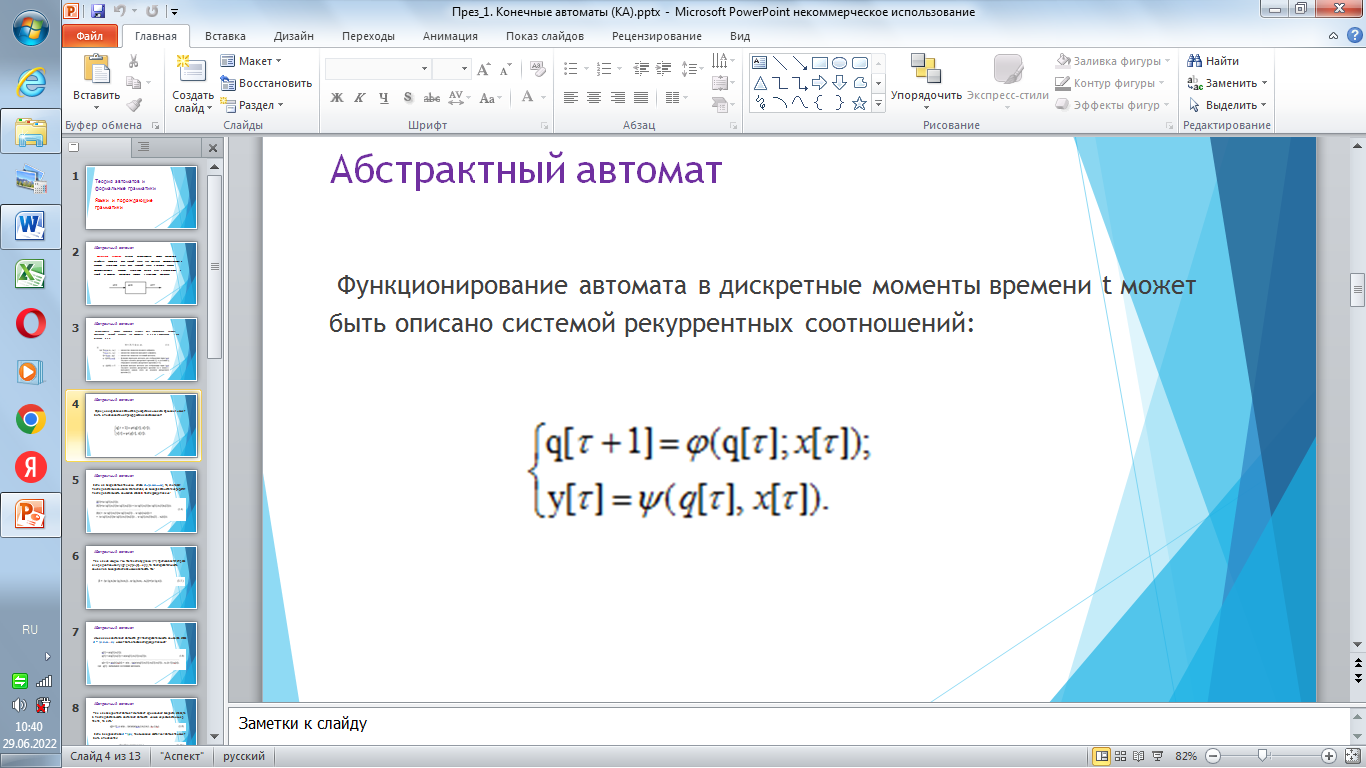
ϕ – функция переходов (если мы находимся в таком-то состоянии и нажимаем такую кнопочку, то переходим в такое то состяние)

Функционирование автомата

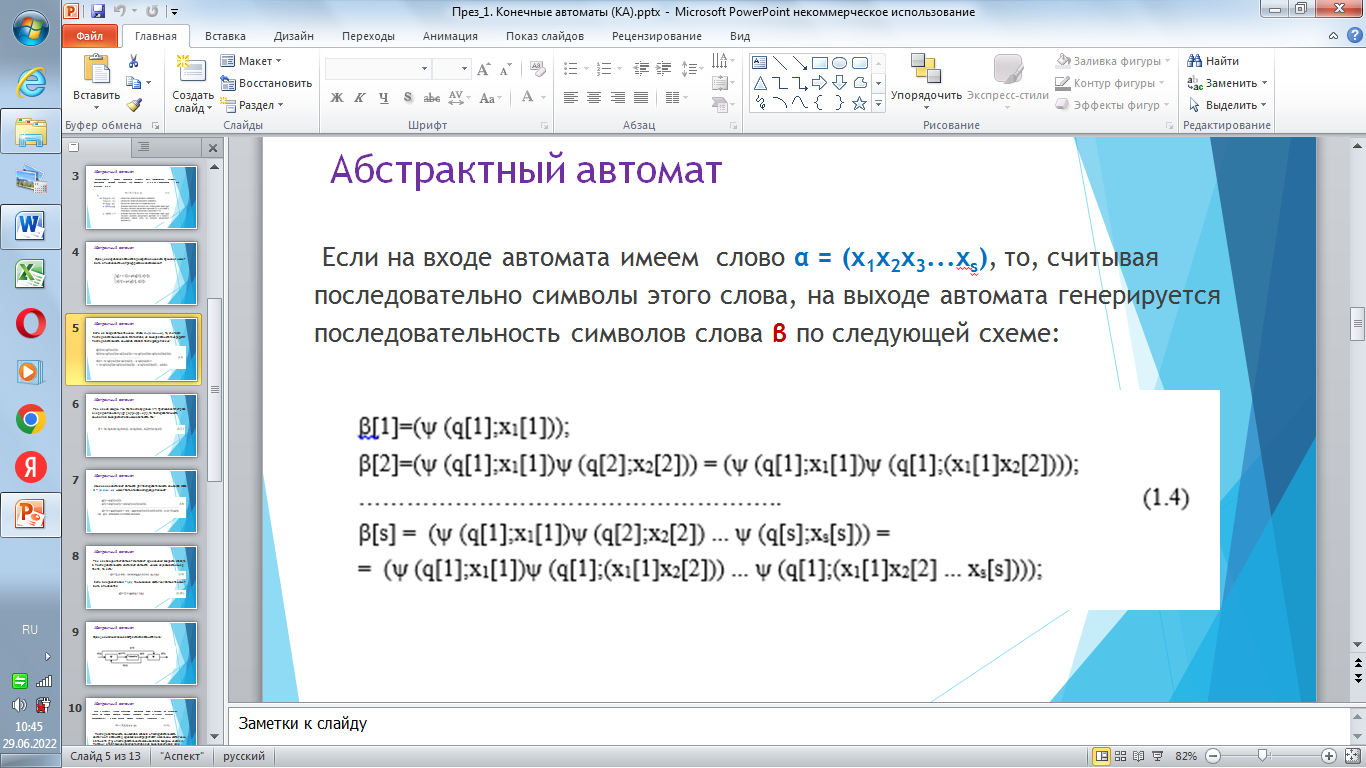
Чтобы узнать, куда мы перейдем в последующий момент времени мы используем функцию переходов, в качестве параметров передаем состояние автомата и входной символ в текущий момент времени

Чтобы узнать, какой сигнал будет на выходе, мы используем функцию выходов, в качестве параметров передаем состояние автомата и входной символ в текущий момент времени.

Мы работаем с автоматом Мили, поэтому считаем, что выходной сигнал появляется сразу. То есть функцией выходов мы определяем сигнал в текущий момент времени



Если на входе слово, то генерация последовательности на выходе будет формироваться по правилу



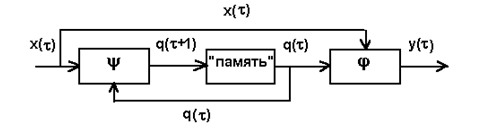
Так как от текущего состояния и входного символа происходит изменение состояния, то нам достаточно в функции указать начальное состояние и все входные символы



То есть так, где альфа – множество входных символов

Изменение состояний строится по этому же принципу, за одним лишь исключением, что текущее состояние и входной символ определяют последующее состояние.

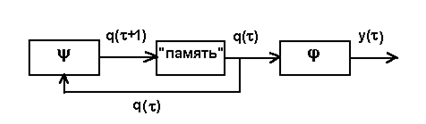
Функциональная схема автомата Мили



|  |  |
| --- | --- |
| Поступает входной символ, из памяти забираем текущее состояние автомата.  ↓  Отправляем в блок с функцией выходов  ↓  Получаем выходной символ | Поступает входной символ, из памяти забираем текущее состояние автомата.  ↓  Отправляем в блок с функцией переходов  ↓  Получаем последующее состояние и отправляем его в память |

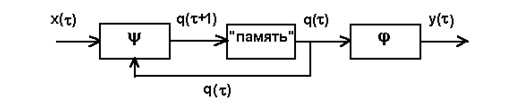
**Типы автоматов**

Порождающий автомат



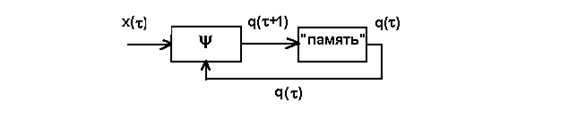
*Порождает* последовательность сообщений на выходе при переходе из одного состояния в другое

Преобразующий автомат



*Преобразовывает* последовательность входных символов в последовательность выходных, при этом автомат меняет свое состояние.

Распознающий автомат



Отличает конечные последовательности входных данных с помощью своих состояний.

**ДКА**

Детерминированный – жестко определенный.

В определенный момент времени автомат находится в определенном состоянии.

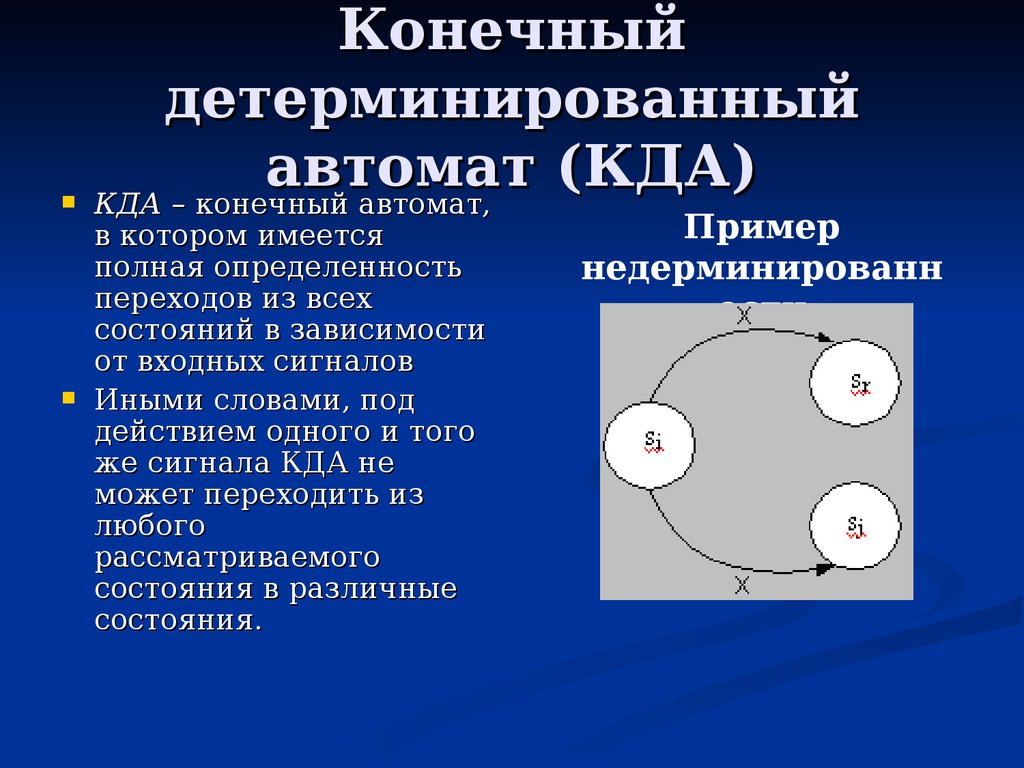


*Красным отмечены конечные состояния – но у нас они отмечаются двойным кружочком*

**НКА**

Недетерминированный – не строго определенный

При одном из проходов по цепочке он может находиться в одном состоянии, а при другом – в другом.

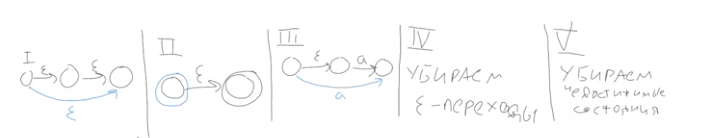


Наличие eps переходов дает недетерминированность?

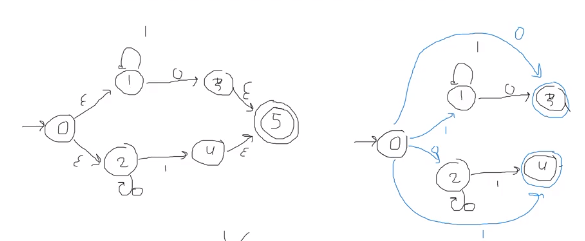
**Преобразование НКА в ДКА**

Преобразование НКА в ДКА делается в 2 этапа

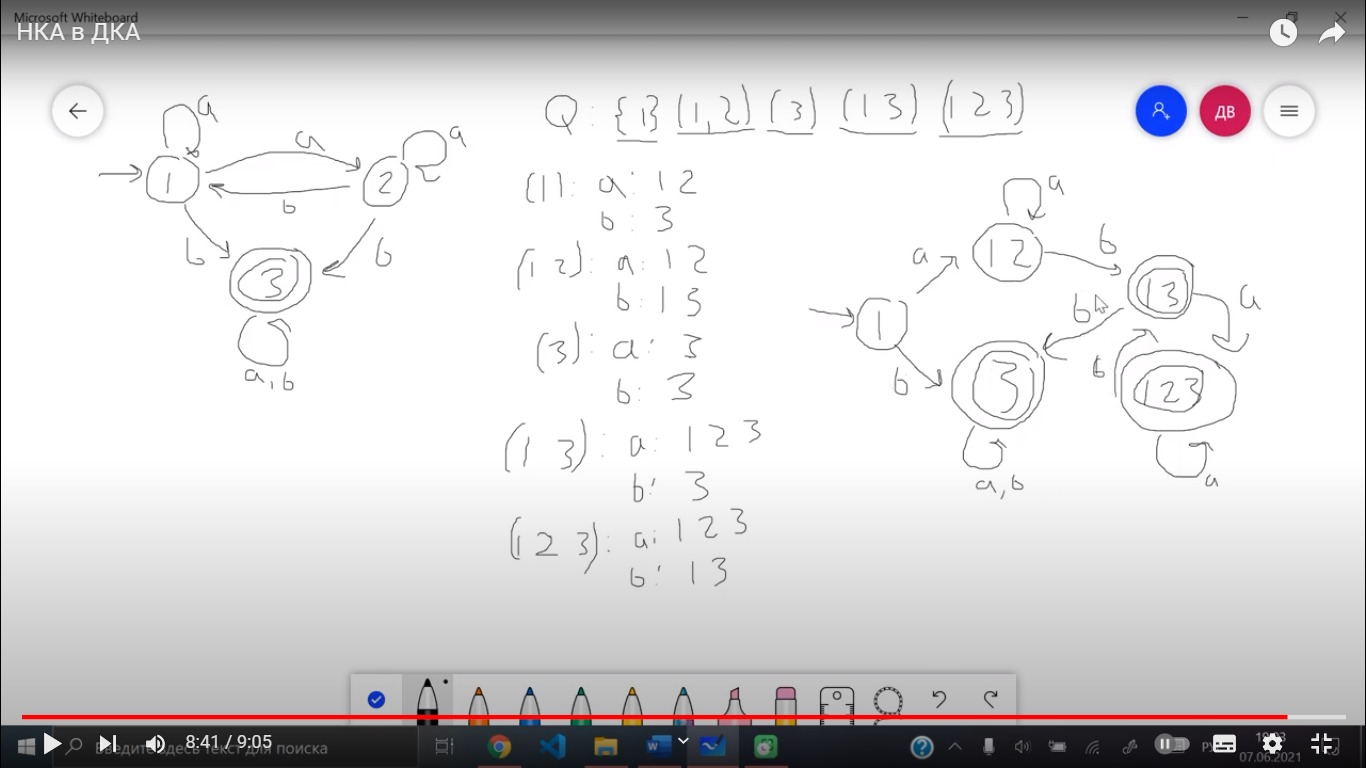
1 этап: убираем eps переходы



Для этого используем вот эти правила



2 этап: формирование нового множества Q



**Формальные языки и формальные грамматики.**

**Основные определения**

Алфавит (V) – конечное непустое множество элементов, называемыми символами(буквами).

Цепочкой символов (словом) – конечная последовательность символов

Пример:

Пусть задан алфавит V = {a,b,c}. Тогда ***α =*** baaaявляется словом в алфавите V.

Цепочка, которая не содержит ни одного символа, называется пустой цепочкой и обозначается e (eps)

Длина цепочки – число составляющих ее символов.

Но длина пустой цепочки равна 0

V\* - множество, содержащее все цепочки в алфавите, включая пустую

V+ - множество, содержащее все цепочки в алфавите без пустой

Говорят, что слово ***х*** *– префикс (начало)* слова ***у****,* если ***у*** *=* ***хи****.*

Говорят, что слово ***х*** *– суффикс (конец)* слова ***у****,* если ***у*** *=* ***их****.*

Говорят, что слово ***х*** *– подслово* слова ***у****,* если ***у*** *=* ***uxv***для некоторых слов ***и***и ***v***

Формальный язык – множество конечных слов (строк и цепочек) над конечным алфавитом V.

Грамматика – система правил, предназначенная для задания слов из символов данного алфавита.

**Классификация формальных грамматик по Хомскому**

Тип 3 – Регулярные

A 🡪 abc

A 🡪 Bab

Ограничения:

В левой части всегда стоит один нетерминальный символ

В правой части либо терминальные символы, либо нетерминальный, но только с одной стороны

Примеры: регулярные выражения.

Могут быть разобраны простым конечным автоматом

Тип 2 – контекстно-свободные

A🡪aAbcBx

Ограничения:

В левой части один нетерминальный символ

В правой части все, что угодно

Могут быть разобраны автоматами с магазинной памятью

*Автоматы с магазинной памятью – обычный конечный автомат, к которому прикрути стек. То есть он умеет писать в стек и читать от туда.*

*A🡪 abc, тут действует и в обратном порядке. То есть идя по строке и видя, abc, мы можем заменить на нетерминальный символ А*

Тип 1 – контекстно – зависимые

xyz A ctx 🡪 xyz abc ctx

Ограничения:

В левой части стоит нетерминальный символ, окруженный терминальными

*Мы можем заменить последовательность терминальных символов abc на нетерминальный символ А лишь в случае, если abc находится между xyz и ctx. Поэтому данная грамматика называется контекстно-зависимой.*

Тип 0

Никаких ограничений нет

**Регулярные грамматики**

Регулярные выражения – один из способов описания языков, используя алгебраические конструкции.

Эти конструкции включают в себя строки символов в некотором фиксированном алфавите, скобки и символы операция \* , + , ∙

Пример:

регулярное выражение t\*rk\* описывает тот же язык, что и грамматика G = (VT, VN, P, S), заданная: VN = {S, A}, VT = {t, r, k}, правила Р: S → tS, S → rA, A → ε, A → kA.

Язык L={t} обозначается регулярным выражением t

Язык L={t, r, k} обозначается как t+r+k («+» - символ обозначающий *объединение множеств*).

Дано: 2 языка – L{001, 10, 111} и M{e, 001}

1. Объединение

Множество цепочек, которые содержатся либо в L, либо в M, либо в обоих языках

L M ={001, 10, 111, е}

2. Конкатенация

Множество цепочек, полученных путем дописывания к любой цепочке L цепочки M

LM={001, 10, 111, 001001, 10001, 111001}

Синим отмечены цепочки, где дописывался e

3. Итерация

Множества цепочек, полученные путем конкатенации любого количество цепочек из L

L={0,11}

L0={e}

L1={0,11}

L2={00, 011, 110, 1111}

L3={000, 0011, 0110, 01111, 1100, 11011, 111111}

Чтобы получить следующий язык, нужно к последовательностям начального языка дописать последовательности предыдущего (правило фонтанчика)

Иерархия:

1 - \*

2 - ∙

3 - +

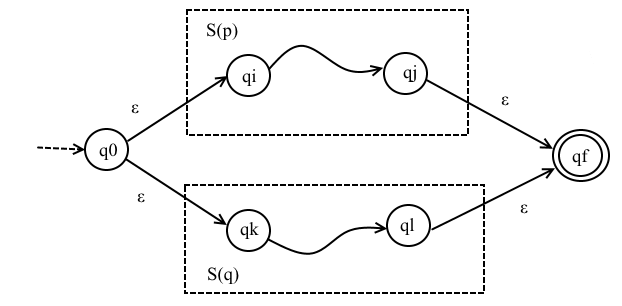
Выражение

p=(r+t∙k)\*

L=({r}+{t}{k})\*={e, r, tk, rr, rtk, …}

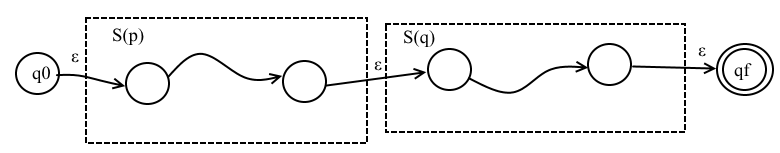
**Регулярные грамматики и КА**

Объединение

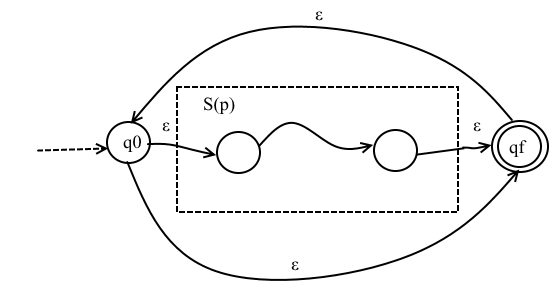


1. Строим граф переходов для каждого языка
2. Добавляем 2 дополнительных вершины – начальная и конечная
3. Соединяем с вершинами языка с е переходом

Конкатенация



Итерация



**Контекстно-свободные грамматики**

МП – автомат

S = (Q, X, M, δ, q0, Z0, F),

Q − конечное множество состояний,

X − конечный входной алфавит,

M − конечный алфавит магазинных символов,

δ: Q х (M ∪ {ε}) х X → 2QxХ\* - функция переходов,

q0 ∈Q − начальное состояние,

Z0 − начальный магазинный символ, так называемый маркер дна,

F ⊆ Q − множество заключительных состояний.