



* ***Словарь*** - конечное множество объектов.
* ***Символы*** - элементы словаря.
* ***Цепочка над словарем***- конечная последовательность символов словаря.
* **Операция конкатенации**: для любых α, β принадлежащих V\*, результат конкатенации (записывается αβ) есть цепочка, получающаяся приписыванием в конец цепочки α символов цепочки β.
* **Операция подстановки**: выполняет замену подцепочки заданной цепочки на другую цепочку: γαδ из V\*, α →β, результат γβδ.
* **Язык L над словарём V** - произвольное множество цепочек над этим словарём.
* **Грамматика** - конечный механизм задания языка.
* Два типа грамматик:

1) **Порождающая грамматика** - за конечное число шагов должна построить правильные цепочки языка.

2) **Распознающая грамматика** - за конечное число шагов определяет, принадлежит ли данная цепочка данному языку.

* **Порождающая грамматика Хомского** - четверка объектов G = (T, N, S, R)
* **Т** – терминальный словарь, состоящий из терминальных символов
* **N** – нетерминальный словарь, состоящий из нетерминальных символов
* **S** из N называется начальным символом
* **R** – конечное непустое множество правил вывода (продукций).

## **Классификация грамматик Хомского:**

## Тип 0 – не накладывается никаких ограничений (α ≠ ε): α → β

## Для распознавания необходимо абстрактное устройство «машина Тьюринга»

## Тип 1(контекстно-зависимые) – ограничения: γAδ → γδ; А – нетерминал;

## Для распознавания необходимо абстрактное устройство «линейно ограниченный автомат»

## Тип 2(контекстно-свободные) – правила имеют ограничения: A → β, где А - нетерминал, а β цепочка.

## Для распознавания необходимо абстрактное устройство «конечный автомат с магазином (стеком)»

## Тип 3(регулярные) - A → tN | t , где t терминал; A,N нетерминалы;

## Для распознавания необходимо абстрактное устройство «конечный автомат»

и

* **Детерминированный конечный автомат(ДКА)** - это пятёрка объектов

Q - конечное множество состояний

- конечное множество входных символов(алфавит)

- функция перехода:

- начальное состояние из Q

- множество допускающих состояний

## **ДКА: Расширенная функция переходов**

Расширенная функция переходов ставит в соответствие состоянию q и цепочке w состояние p, в которое попадает автомат из состояния q, обработав цепочку w.

Базис

=q

Индукция

Пусть **w=x**a

Если , то

* **Недетерминированный конечный автомат(НКА)** - это пятёрка объектов

Q - конечное множество состояний

- алфавит

- функция, аргументами которой является состояние из Q и входной символ из , а значение - некоторое подмножество множества Q

- один из элементов Q, начальное состояния

- множество допускающих состояний

## **НКА: Расширенная функция переходов**

Расширенная функция переходов ставит в соответствие состоянию q и цепочке w множество состояний p, в которое попадает автомат из состояния q, обработав цепочку w.

Базис:

={q}

Индукция:

Пусть w=xa

Если , то

* **Язык НКА**:
* ***Формальное определение -НКА***

**-замыкание**

Базис:

ECLOSE(q) содержит q

Индукция:

Если ECLOSE(q) содержит состояние p и существует переход, отмеченный из p в r, то ECLOSE(q) содержит r.

## **-НКА: Расширенная функция переходов**

Базис:

ECLOSE(q)

Индукция:

Пусть w=x, из

# 

* **Регулярный язык** — множество слов, которое распознает некоторый конечный автомат за конечное число шагов.

## Построение регулярного выражения

Базис:

– константы Ø и суть РВ, определяющие языки Ø и {}

– если a символ алфавита, то a – РВ, определяющее язык {a}

Индукция:

– E и F суть РВ => E+F тоже РВ, определяющее объединение языков L(E) и L(F): L(E) U L(F)

– E и F суть РВ => EF тоже РВ, определяющее конкатенацию языков L(E) и L(F): L(E)L(F)

– E есть РВ => E\* тоже РВ, определяющее итерацию языка L(E): L(E\*)=(L(E))\*

– E есть РВ => (E) тоже РВ, определяющее тот же язык L(E), что и выражение E: L((E))=L(E)

* Операции над регулярными языками:

1. **Объединение языков** (L U M) - множество цепочек, содержащихся либо в L, либо в M, либо в обоих языках.

2. **Конкатенация языков** (LM) - множество цепочек, которые можно образовать путем дописывания к любой цепочке из L в ее конец любой цепочки из M.

3. **Итерация** (замыкание Клини) (L\*) представляет множество цепочек, образованных путем конкатенации любого (и нулевого) количества цепочек из L.

* Алгебраические законы для РВ

• L+M=M+L

• (L+M)+N=L+(M+N)

• (LM)N=L(MN)

* Единичные и нулевые элементы

• Ø+L=L+Ø=L

• L=L=L

• ØL=LØ=Ø

• L(M+N)=LM+LN – левосторонняя дистрибутивная конкатенация относительно объединения

• (M+N)L=ML+NL – правосторонняя дистрибутивная конкатенация относительно объединения

* Операция называется **идемпотентной**, если результат ее применения к двум одинаковым значениям равен этому значению.(L+L=L)
* Состояния p и q **эквивалентны**, если: для всех входных цепочек состояние ) является допускающим тогда и только тогда, когда состояние ) является допускающим.
* Если два состояние p и q не эквивалентны, то говорят, что они **различимы**.
* **Контекстно-свободные грамматики** - формальная запись рекурсивных определений языков.

## 

## КС грамматика G состоит из четырех компонентов G=(V,T,P,S):

## V – конечное множество переменных (нетерминальные символы, нетерминалы, синтаксические категории)

## T – конечное множество символов (терминальные символы, терминалы), из которых состоят цепочки языка

## Одна из переменных S представляет определяемый язык (стартовый символ)

## P – конечное множество продукций (правил вывода), состоящих из следующих частей:

## Переменная, определяемая продукцией (голова)

## Символ продукции

## Конечная цепочка терминалов и нетерминалов, возможно, пустая (тело продукции)

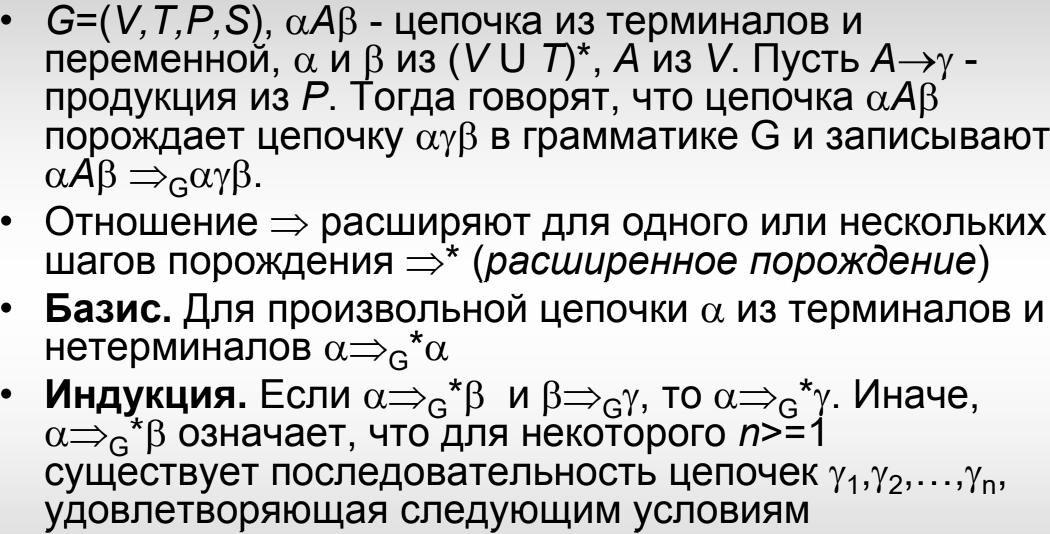
## **Продукции КС грамматики используются для проверки принадлежности цепочки языку**

## – «от тела к голове»: рассматриваем цепочки и убеждаемся, что полученная цепочка принадлежит языку переменной в голове (**рекурсивный вывод**)

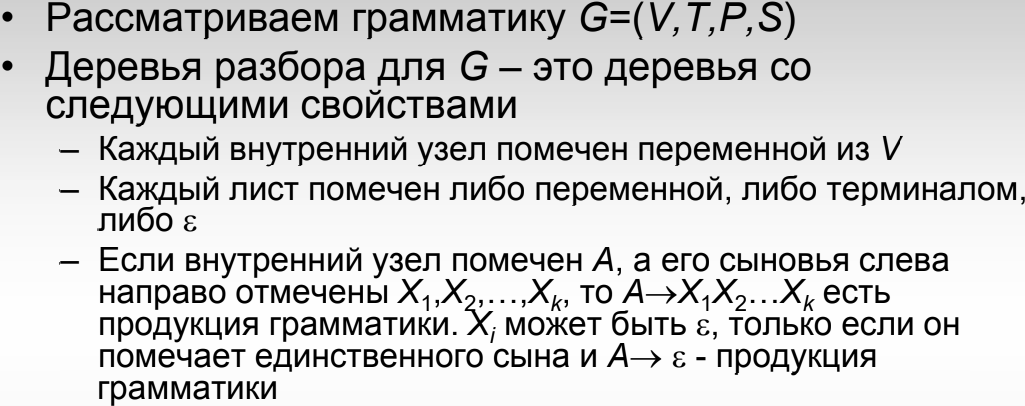
## 

## – «от головы к телу»: разворачиваем стартовый символ, используя одну из его продукций, затем разворачиваем полученную цепочку, заменяя переменные телами их продукций, пока не получится терминальная цепочка (**порождение**)

## **Отношение (порождение)(для цепочек)(цепочка цепочка)**



* **Левое порождение** - на каждом шаге заменяем крайнюю слева переменную одним из тела ее продукции
* **Правое порождение** - на каждом шаге заменяем крайнюю справа переменную одним из тел ее продукции
* **Язык, задаваемый грамматикой** G=(V,T,P,S)(L(G)), представляет собой множество терминальных цепочек, порождаемых из стартового символа:
* ***Сентенциальная форма***(**выводимая цепочка**) – это цепочка из терминалов и нетерминалов, выводимая из стартового символа



**Крона дерева**- отметки листьев дерева разбора, рассматривая их слева направо

* Говорят, что КС грамматика G=(V,T,P,S) является ***неоднозначной***, если найдется хотя бы одна терминальная цепочка w из T\*, для которой существуют два разных дерева разбора.
* Если каждая терминальная цепочка имеет одно дерево разбора в грамматике, то грамматика ***однозначна****.*
* КС язык называется ***существенно неоднозначным***, если все его грамматики неоднозначны.
* ***Сомножитель*** – выражение, которое не может быть разделено на части никаким символом примыкающей операции (\*,+)
* Идентификаторы
* Выражения в скобках
* ***Терм*** – последовательность из одного или нескольких сомножителей, связанных символом \*
* ***Выражение*** – сумма одного или нескольких термов
* ***МП-автомат*** – это семерка объектов

Q - конечное множество состояний

- конечное множество входных символов(алфавит)

Г - конечный магазинный алфавит

- функция переходов , аргументами которой является:

* q - состояние из Q,
* входной символ из или пустая цепочка
* X - магазинный символ из Г

Выход образует пары , где

* p - новое состояние
* - цепочка магазинных символов, замещающая X на вершине магазина

- один из элементов Q, начальное состояния

- начальный магазинный символ(“маркер дна”)

- множество допускающих состояний

* Используем символ ⊢\* для представления нуль или нескольких переходов МП автомата (аналог расширенной функции перехода КА):

Базис

I ⊢\* I для любой конфигурации I

Индукция

I ⊢\* J , если существует некоторая конфигурация K, удовлетворяющее условиям I ⊢ K и K ⊢\*J.

Таким образом, I ⊢\* J , если существует последовательность конфигураций , ,…, , у которой I=, J= и ⊢ для всех i=1,2,…,n-1.

* ***Конфигурация*** (мгновенное описание, МО) в каждый момент времени содержит состояние q, непрочитанную часть ввода w и содержимое магазина : тройка (q, w, )

## **Допустимость по заключительному состоянию**

Пусть - МП автомат.

Тогда языком L(P), допускаемым автоматом P по заключительному состоянию, является множество цепочек: , где и произвольной магазинной цепочки

## **Допустимость по пустому магазину**

Пусть - МП автомат.

Тогда языком L(P), допускаемым автоматом P по пустому магазину, является множество цепочек: , где

## ДМП (детерминированный МП-автомат)

МП автомат определяется как детерминированный (ДМП автомат), если выполнены условия:

(q,a,X) имеет не более одного элемента для каждого q Q, a или a= и X

* Говорят, что язык L имеет **префиксное свойство**, если в L НЕТ двух различных цепочек x и y таких, что x является префиксом y.

**Беспрефиксный**: не существует такой цепочки, являющейся префиксом другой цепочки.

* **Нормальные формы КС грамматик**

Продукции вида:

Нормальная форма Хомского

1. Удалить -продукции (A)

2. Удалить цепные продукции (AB)

3. Удалить бесполезные символы (не встречаются в порождениях терминальных цепочек из стартового символа)

* Символ X называется ***полезным*** в грамматике G=(V,T,P,S), если существует порождение вида
* Символ X называется ***порождающим***, если для некоторой терминальной цепочки .
* Символ X называется ***достижимым***, если существует порождение для некоторых α, 𝛽 ∈ (𝑉 ∪ 𝑇)
* Переменная A называется ***-порождающей***, если A\*.
* ***Цепная продукция*** – это продукция вида AB, где A и B являются переменными.
* ***Нормальная форма Хомского***
* Нормальная форма Хомского (НФХ) грамматики для КС языка без -продукций и бесполезных символов:
* Все правила неукорачивающие.
* Вывод строки длины n занимает 2n-1 шагов.
* Дерево разбора представляет бинарное дерево, высота которого ограничена длиной строки.
* ***Нормальная форма Грейбах***
* продукции вида: , k0, - переменные
* язык не содержит пустую цепочку
* цепочка длины n порождается ровно за n шагов(каждое использование продукции вводит ровно один терминал)
* Пусть для каждого терминала 𝑎 из алфавита Σ поставлен в соответствие некоторый язык (над произвольным алфавитом).

Тогда говорят, что задана ***подстановка***𝑠(𝑤), заменяющая в терминальной цепочке 𝑤 из символ 𝑎 любой цепочкой из языка :

𝑤 = , 𝑠(𝑤) = , – цепочка из языка : 𝑠(𝑤) = 𝑠()𝑠() … 𝑠()

𝑠(𝐿) это объединение всех 𝑠(𝑤) из языка 𝐿.

Операции над языками:

* ***Объединение***

Пусть L и М языки в алфавите Σ.

Тогда язык L U M содержит все цепочки, которые принадлежат хотя бы одному из языков L или M.

* ***Пересечение***

Пусть L и М языки в алфавите Σ. Тогда язык L M содержит все цепочки, которые принадлежат обоим языкам L и M.

* ***Дополнение***

Пусть L язык в алфавите Σ. Тогда язык дополнение L содержит те цепочки из Σ\*, которые не принадлежат языку L.

* ***Разность***

Пусть L и М языки в алфавите Σ. Тогда язык L-M содержит все цепочки, которые принадлежат языку L и не принадлежат M.

* ***Обращение***

Обращением цепочки называется цепочка, записанная в обратном порядке.

Обращение языка , обозначаемое , состоит из всех цепочек, обратных цепочкам языка .

* ***Гомоморфизм***

Гомоморфизм – это такая функция на множестве цепочек, которая подставляет определенную цепочку вместо каждого символа данной цепочки.

Гомоморфизм языка определяется с помощью его применения к каждой цепочке языка, то есть если – язык в алфавите , а – гомоморфизм на , то

--------------------------------------------------------------Теоремы-----------------------------------------------------------------------

**Теорема**

НКА в ДКА(конструкция подмножеств)

* - множество всех подмножеств .
* состоит из всех множеств состояний N, содержащих хотя бы одно допускающее состояние N.

Если ДКА построен по НКА посредством конструкции подмножеств, то .

Док-во: | дано | расширенные функции перехода для НКА | конструкцию подмножества |

## “Ленивый” алгоритм

Базис:

Знаем, что одноэлементное множество, состоящее из начального состояния N, является достижимым.

Индукция:

Множество состояний S является достижимым.

Тогда для каждого входного символа *a* нужно найти множество состояний .

Найденные таким образом множества состояний также будут достижимы.

**Теорема** (ДКА ⇔ НКА)

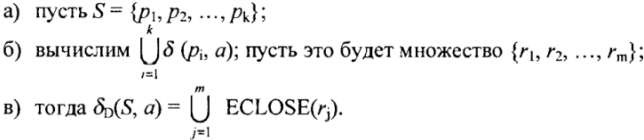
Язык L допустим некоторым ДКА тогда и только тогда, когда он допускается некоторым НКА.

Док-во:Д: Из конструкции подмножеств и предыдущей теоремы.Н: Строим ДКА экв. НКА “От противного”

**Теорема** (-НКА ДКА)

Пусть – -НКА. Эквивалентный ДКА .

1. – множество -замкнутых подмножеств
2. ECLOSE()
3. = {S | S принадлежит и }



Язык L допускается некоторым -НКА тогда и только тогда, когда L допускается некоторым ДКА.

Док-во:

*Д:* ДКА преобразовать в -НКА | | →

*Н:* | , |

**Теорема** (ДКА ⇒ РВ)

Если для некоторого ДКА , то существует регулярное выражение , причем .

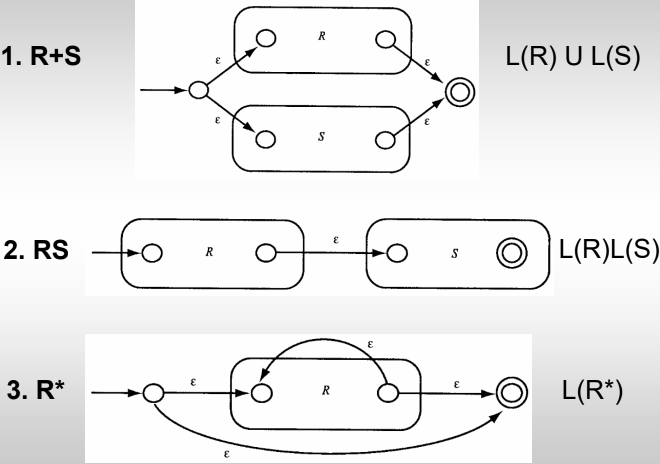
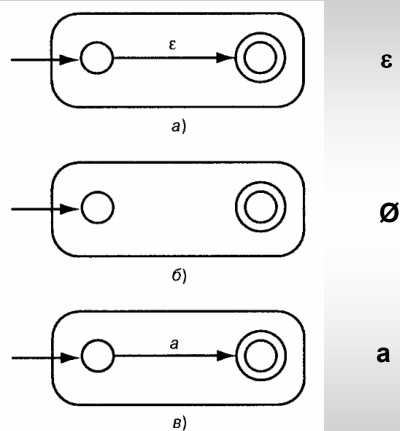
Док-во: строим РВ, язык которого состоит из множества цепочек w, ведущих от состояния i к состоянию j ДКА и не имеющих промежуточных состояний с номерами > k () |индуктивное определение от k=0 до k=n | 

**Теорема** (РВ ⇒ -НКА)

Любой язык, определяемый РВ, можно задать некоторым конечным автоматом(-НКА).

Конструктивное доказательство:

Базис Индукция



**Теорема.**

Если L, M и N – произвольные языки, то L(M U N)=LM U LN

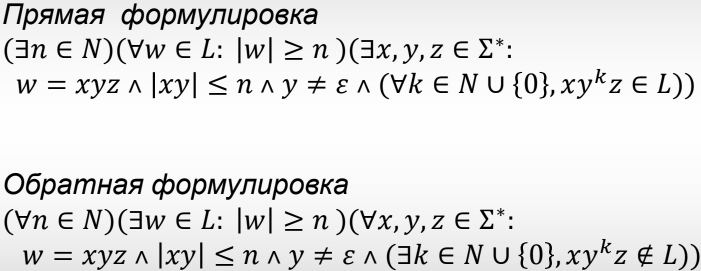
Док-во: w = xy | где x принадлежит …, y принадлежит ….

**Теорема (лемма о накачке для регулярных языков)**

Пусть - регулярный язык.

Существует константа n (зависящая от ), для которой каждую цепочку из языка , удовлетворяющую неравенству , можно разбить на три цепочки так, что выполняются следующие условие.

1. .
2. .
3. Для любого цепочка также принадлежит .



Док-во: ДКА имеет состояний | | опред сост. как ) | рассмотрим n+1 состояний

**Теорема**

Алгоритм заполнения таблицы

Базис

если p допускающее, а q не допускающее, то пара состояний {p, q} различима

Индукция

пусть для p и q существует входной символ a, приводящий их в различимые состояния. Тогда пара {p, q} различима.

Если два состояния не различаются с помощью алгоритма заполнения таблицы, то они эквивалентны.

Док-во: плохая пара | кратчайшая цепочка, различ. плохую пару | предшественники для плохой пары

**Теорема**

Эквивалентность состояний транзитивна.

Док-во: от противного| 2 экв. пары и 1 различ.| существует цепочка w, для которой только одно из состояний ) и ) допускающее

**Теорема**

Если для каждого состояния q некоторого ДКА создать блок, состоящий из q и эквивалентных ему состояний, то различные блоки образуют разбиение множества состояний.

Док-во: отношение по эквивалентным состояниям - это отношение эквивалентности(рефлексивно, транзитивно, симметрично) | Всякое отношение эквивалентности определяет разбиение множества X на классы эквивалентности по этому отношению(на множества, в которых каждый элемент эквивалентен каждому) ([x]={y|xy}

**Теорема.**

Алгоритм минимизации ДКА

## Алгоритмом заполнения таблицы определяем все пары эквивалентных состояний

## Разбиваем множество состояний Q на блоки взаимно эквивалентных состояний (Q)

## Строим ДКА B с минимальным числом состояний, используя в качестве его состояний полученные блоки

## Функция переходов (S,a), S – блок эквивалентных состояний автомата A, a – входной символ: существует один блок T, содержащий (q,a) для всех состояний q из S. ()

## Начальное состояние автомата B – блок, содержащий начальное состояние автомата A ()

## Множество допускающих состояний автомата B – множество блоков, содержащих допускающие состояния автомата A (F)

## Если из некоторого ДКА A с помощью алгоритма минимизации построен ДКА M, то M имеет наименьшее число состояний из всех ДКА, эквивалентных A.

Док-во:ДКА N(меньше сост., чем в M) | алгоритм заполнения таблицы для M и N | каждое состояние автомата M неотличимо хотя бы от одного состояния автомата N.

**Теорема** (От рекурсивного вывода к дереву)

Пусть G=(V,T,P,S) – КС грамматика.

Если рекурсивный вывод утверждает, что терминальная цепочка w принадлежит языку переменной A, то существует дерево разбора с корнем A и кроной w.

Док-во: Индукцией по числу шагов в рекурсивном выводе |продукция A ,рассмотрим последний шаг рекурсивного вывода.| вывод из

**Теорема(**От деревьев к левым порождениям**)**

Пусть G=(V,T,P,S) – КС грамматика.

Существует дерево разбора с корнем A и кроной w из терминальных символов => в грамматике G существует левое порождение

Aw

Док-во: Индукция по высоте дерева | Б: дерево→продукция→порождение| И: Корень A и сыновья |Начнем с шага | | вывод |

**Теорема(**От деревьев к правым порождениям**)**

Пусть G = (V,T,P,S) - КС грамматика.

Предположим, что существует дерево разбора с корнем А и кроной w из терминальных символов. Тогда в грамматике G существует правое порождение

**Теорема(**От порождений к рекурсивному выводу**)**

Пусть G=(V,T,P,S) – КС грамматика.

Существует терминальное порождение => существует рекурсивный вывод .

Док-во:Индукцией по длине порождения |Б: порождение→продукция→рек.вывод|И: |

**Теорема**

Для каждой грамматики G=(V,T,P,S) и w из T\* цепочка w имеет два разных дерева разбора тогда и только тогда, когда w имеет два разных левых порождения.

Док-во: узел различных продукций → левые порождения различны (и обратно)

—----------------------------------------------------------------------------------------------------------------------------------------------

**Принципы конфигураций МП автомата**

1. Если последовательность конфигураций является допустимой, то вычисление, образованное дописыванием одной и той же цепочки к концам входных цепочек всех конфигураций, также допустимо.

2. Если вычисление допустимо, то вычисление, образованное дописыванием одних и тех же магазинных символов внизу магазина в каждой конфигурации, также допустимо.

3. Если вычисление допустимо и в результате некоторый остаток входной цепочки не прочитан, то вычисление, полученное удалением этого остатка из входных цепочек каждой конфигурации, также допустимо.

**Теорема** (из 1 и 2 принципов)

Если – МП автомат и (q,x,) ⊢\*(p,y, ), то для любой цепочки w из \* и из Г\* верно следующее утверждение: (q,xw,) ⊢\*(p,yw,)

Док-во: обосновывается переходами МП автомата P без какого-либо использования w или

**Теорема**(из 3-его принципа)

Если – МП автомат и (q,xw,) ⊢\*(p,yw,) , то верно также (q,x,) ⊢\*(p,y,)

**Теорема** (от МП-автомата по пустому магазину к МП-автомату по заключительному состоянию)

Если N() допускаемый по пустому магазину язык некоторого МП автомата , то существует МП автомат , допускающий язык автомата по заключительному состоянию.

Док-во: | | = | = ( ,) q.

Д( wN() wL() ) | | (2 пр. кон.) |

\*

Н( wN() wL() ) | все вычисления в

**Теорема**. (от МП-автомата по заключительному состоянию к МП-автомату по пустому магазину)

Если L( ) допускаемый по заключительному состоянию язык некоторого МП автомата , то существует МП автомат , допускающий язык автомата по пустому магазину, то есть N( ) = L()

Док-во: | | =

(q,,Y) = (p,) q | (p,,Y)={(p,)}

**Теорема** (От КС грамматик к МП-автоматам)

G=(V,T,Q,S) тогда P=({q},T, V U T, ,q,S)

1. (q,,A)={(q, ) | A} для каждой переменной A

2. (q,a,a)={(q, )} для каждого терминала a

Если МП автомат P построен по грамматике G в соответствии с описанной выше конструкцией, то N(P)=L(G)

Док-во:

**Д:**  L(G) =>S==w | индукцией по | =, - цепочка, которую мы прочитали | ↴ | включает замену переменной в остатке одним из тел ее продукций | правило 1→ правило 2 МП-автомата

**Н:**  тогда |1-ый переход: | В последующих n-1 переходах автомат читает цепочку x и выталкивает из магазина по очереди | порция входа, прочитанная до выталкивания |

**Теорема**. (От МП-автомата по пустому магазину к КС грамматике)

Пусть – МП автомат. Тогда существует КС грамматика G, для которой L(G)=N(P).

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 для всех возможных списков состояний в грамматике G есть продукции

Док-во:

Д**:**индукцией по числу переходов автомата | ->

∃ | по пост. ∃ | удаляются по очереди и автомат находится в состоянии после удаления . | ↝ []\*

Н**:**индукцией по числу шагов в порождении | [qXp]\*w |

, где | в МП по индукции:

**Теорема** (Из регулярной грамматики в ДМП автомат)

Если L – регулярный язык, то L=L(P) для некоторого ДМП автомата P

Док-во: строим ДМП эквивалентный ДКА

Д(из ДКА в ДМП)**:** (, xa) = 𝛿 ( (, x),a)= 𝑝 | ( (, x) ,𝑎,) (𝑝,𝜀, ) по постр. |по индукционному предположению (, x, )( (, x),𝜀, ) | по принципу конфигурации (, xa, )(, a, )

Н**:**  | 1 переход по индукции, 2 переход по построению

**Теорема**

Язык L=N(P) для некоторого ДМП автомата P тогда и только тогда, когда L имеет префиксное свойство и L = L(P’) для некоторого ДМП автомата P’.

Док-во:

Н**:** От противного: L не беспрефиксный | алгоритм построения автоматов по разному допуску

Д**:**Удалить переходы из допускающих состояний добавить переходы в состояние очистки стека

**Теорема**.

Классы языков, задаваемых МП-автоматами и ДМП-автоматами с допуском по допускающему состоянию не совпадают.

Док-во: 𝐿 = классы пересекаются | 𝐿 = не совпадают

**Теорема**

Если L=N(P) для некоторого ДМП автомата P по пустому магазину, то L имеет однозначную КС грамматику.

Док-во: ∃ только одна последовательность переходов => однозначно определяем выбор продукции в левом порождении.

**Теорема**.

Если L=L(P) для некоторого ДМП автомата P по заключительному состоянию, то L имеет однозначную КС грамматику.

Док-во: L’=L U {$} | L’ =N() некоторого ДМП автомата | существует однозначная грамматика | G получится добавлением $ → 𝜀

—----------------------------------------------------------------------------------------------------------------------------------------------

**Теорема**

Пусть G=(V,T,P,S) – КС грамматика с не пустым языком L(G).

Пусть – КС грамматика, полученная с помощью следующих шагов:

1. Вначале удаляются все непорождающие символы и все продукции, содержащие хотя бы один такой символ. Получим

2. Удаляются все недостижимые в символы.

Тогда не имеет бесполезных символов и L()= L(G)

Док-во: X оставшийся символ в (порождающий и достижимый) | достижим(поэтому не удалим из) → порождающий в | | . | L()L()

**Теорема**.

Вычисление порождающих символов

*Базис*

Каждый терминальный символ является порождающим, он порождает самого себя.

*Индукция*

Пусть существует продукция A, каждый символ в является порождающим.

Тогда символ A порождающий.

Приведенный алгоритм находит все порождающие символы.

Док-во:

* Добавляемый алгоритмом символ – порождающий | индукция по числу шагов алгоритма | послед. шаг: Xα | В цепочке α все символы порождающие
* Любой порождающий символ определяется алгоритмом | индукция по длине порождения |

X – переменная и Xα \*w.| Каждый символ цепочки α порождает терминальную цепочку менее чем за n шагов

**Теорема**

Вычисление достижимых символов

*Базис*

Символ S достижим.

*Индукция*

Пусть некоторая переменная A достижима.

Тогда для всех продукций с головой A все символы тел достижимы.

Приведенный алгоритм находит все достижимые символы.

Док-во:

* Добавляемый алгоритмом символ – достижимый | Последний шаг: AαXβ | К переменной A дошли за n-1 шаг
* Любой достижимый символ определяется алгоритмом | Последний шаг: AαXβ | Для порождения сентенциальной формы с A потребовалось n-1 шагов

**Теорема**

**Базис**. Если A - продукция, то A - -порождающая.

**Индукция**. Если в грамматике есть продукция , где каждый символ является -порождающим, то B - -порождающая переменная.

В любой грамматике -порождающими являются те и только те переменные, которые найдены вышеприведенным алгоритмом

Док-во:

* Добавляемый алгоритмом символ A – -порождающий | Последний шаг: Aα => Aα |В α все символы -порождающие
* Любой -порождающий символ определяется алгоритмом | Aα\* | каждый символ α -порождающий

**Теорема**

Если грамматика построена по грамматике G с помощью описанной выше конструкции удаления -продукций, то L()=L(G)-{}

Док-во: <=> и w ≠

Н**:** A | 1-ая продукция построена по A , тело которой : и символы, которые сменяются ε –порождающими переменными | = , где , i=1,...,k.

Д**:** Aw | w= , где | Обозначим через те , для которых ≠ ε | , но

**Теорема**

Базис

(A,A) цепная пара

Индукция

(A,B) цепная пара и существует BC. Тогда (A,C) цепная пара.

Приведенный выше алгоритм находит все цепные пары грамматики и только их.

Док-во:

* Алгоритм обнаружил цепную пару (A,B), то A \*B только с помощью цепных продукций | (A,B) – цепная пара за n шагов алгоритма из цепной пары (A,C) и цепной продукции CB | Пара (A,C) за n-1 шаг обнаружена
* Пусть A\*B обнаружено только с помощью цепных продукций. Алгоритм обнаружит пару (A,B) | A\*B за n шагов только цепными продукциями.| A\*CB | (A,C) обнаруживается по индукции

**Теорема**.

Построим по грамматике КС грамматику следующим образом:

1. Найдем все цепные пары грамматики G.

2. Удаляем все цепные пары.

3. Добавляем переходы, которые ранее обеспечивали цепные пары.

Если грамматика построена по грамматике с помощью представленного выше алгоритма удаления цепных продукций, то L()=L(G).

Док-во:

Н: каждые цепные продукции с последующей нецепной продукцией выполняются одной продукцией.

Д:Любая продукция грамматики эквивалентна последовательности из нуль или более цепных продукций G с завершающей нецепной продукцией.

**Теорема**

1. Удалить -продукции (A)

2. Удалить цепные продукции (AB)

3. Удалить бесполезные символы (не встречаются в порождениях терминальных цепочек из стартового символа)

Если G - КС грамматика, порождающая язык, в котором есть хотя бы одна непустая цепочка, то существует другая грамматика , не имеющая бесполезных символов, -продукций и цепных продукций, у которой L()=L(G)-{}.

Док-во: по пред. теоремам

**Теорема**

Если G – КС грамматика хотя бы с одной непустой цепочкой, то существует НФХ-грамматика , язык которой L()=L(G)-{}

Док-во:

Н: можно заменить последовательностью продукций.

Д: Куст соответствует продукции

**Теорема**.

Пусть дано дерево разбора, соответствующее НФХ-грамматике 𝐺 = (𝑉, 𝑇, 𝑃, 𝑆) с кроной 𝑤.

Если 𝑛 наибольшая длина пути (от корня к листьям), то |𝑤|

Док-во: Корень дерева использует продукцию ABC. Два поддерева, начинающиеся с 𝐵 и 𝐶 имеют длину не больше, чем 𝑛 − 1, их кроны длиной не более . Крона всего дерева имеет длину не более + =

**Лемма о накачке(КС-языки)**

Пусть 𝐿 – КС язык. Тогда существует такая константа 𝑛 > 0, что если 𝑧 – произвольная цепочка из 𝐿, длина которой не меньше n, то можно представить 𝑧 = 𝑢𝑣𝑤𝑥𝑦, причем выполняются следующие условия:

1. |𝑣𝑤𝑥|𝑛

2. 𝑣𝑥

3. для всех 𝑖0.

Док-во:найдем НФХ-грамматику с m переменными | 𝑛 = |дерево с кроной 𝑧 (|𝑧|𝑛=) имеет путь длиной не менее 𝑚 + 1| Рассмотрим самый длинный путь | хотя бы две переменных должны совпасть | = , где | выбирали как можно ближе к кроне дерева | 𝑘 − 𝑖 𝑚 |самый длинный путь в поддереве с корнем имеет длину не более 𝑚 + 1 | крона имеет длину не более = 𝑛

**Следствие леммы о накачке**

В любом бесконечном КС языке существует последовательность цепочек, длины которых образуют возрастающую арифметическую прогрессию

Док-во: последовательность =

**Лемма Огдена для КС языков**

Для каждой КС-грамматики 𝐺 = (𝑉, 𝑇, 𝑃, 𝑆) существует такое целое число 𝑘1, что если 𝑧𝐿(𝐺), |𝑧|𝑘 и в цепочке 𝑧 выделены 𝑘 или более различных позиций, то 𝑧 можно записать в виде 𝑢𝑣𝑤𝑥𝑦, причем

(1) 𝑤 содержит хотя бы одну выделенную позицию;

(2) либо 𝑢 и 𝑣 обе содержат выделенные позиции, либо 𝑥 и 𝑦 обе содержат выделенные позиции;

(3) 𝑣𝑤𝑥 содержит не более 𝑘 выделенных позиций;

(4) ∀𝑖 ≥ 0

—----------------------------------------------------------------------------------------------------------------------------------------------

**Теорема**

Если 𝐿 – КС язык над алфавитом Σ, а 𝑠 – подстановка над Σ, при которой 𝑠(𝑎) является КС языком для любого 𝑎 из Σ, то 𝑠(𝐿) также является КС языком.

Док-во:

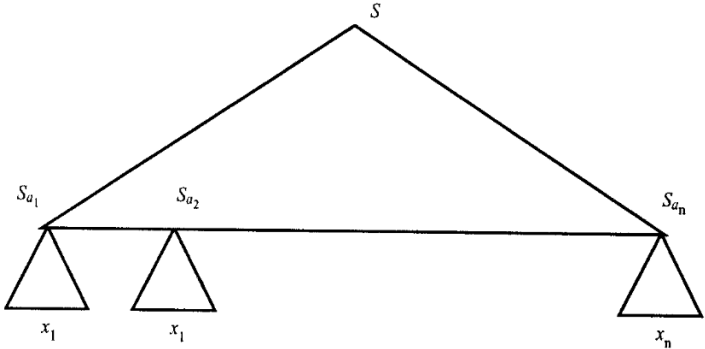
𝐺 = {𝑇, 𝑉, 𝑃, 𝑆} | –𝑠(𝑎) | 𝐺’ = {𝑇’, 𝑉’, 𝑃’, 𝑆} для 𝑠(𝐿):𝑉’ = 𝑉 ∪ ( ),𝑇’ = ,

𝑃’ состоит из:

* продукций
* продукций 𝑃, в которых каждый 𝑎 заменен на | ↴ 𝐿(𝐺’)=𝑠(𝐿) |

**Д: ∃** 𝑥 = в 𝐿 и цепочки из языков 𝑠(), что 𝑤 = . 𝑤 из 𝑠(𝐿)| Часть грамматики 𝐺’, состоящая из правил грамматики 𝐺 с замененными на переменные терминалами из Σ, выведет цепочку .| Правила грамматик являются и правилами грамматики 𝐺’, а они позволяют вывести из .

**Н:**Выводится цепочка , но каждое , выводимое из , принадлежит языку 𝑠(), а следовательно и 𝑤 принадлежит языку 𝑠(𝐿).



**Теорема**

КС языки замкнуты относительно операций:

1. объединение;

2. конкатенация;

3. итерация;

4. гомоморфизм.

Док-во:

1. ∪ является языком 𝑠(𝐿), 𝐿 = {1,2}, 𝑠(1) = , 𝑠(2) = .

2. является языком 𝑠(𝐿), 𝐿 = {12}, 𝑠(1) = , 𝑠(2) = .

3. является языком 𝑠(𝐿), 𝐿 = {1}, 𝑠(1) = .

4. 𝐿 является языком над алфавитом Σ, ℎ – гомоморфизм на Σ, 𝑠(𝑎) = {ℎ(𝑎)} для всех 𝑎 из Σ.

**Теорема**.

Если 𝐿 - КС язык, то и – КС язык.

Док-во: = {𝑇, 𝑉, , 𝑆}: 𝐴 → 𝛼 ⇒ 𝐴 → | ↴𝐿() =

**Н(**𝑤 ∈ 𝐿() ⇒ 𝑤 ∈ )**:**

𝑆 ⇒ 𝑤 | 𝑤 = , где =>

в грамматике 𝐺 : =>

𝑆 ⇒ ∈ 𝐿 => 𝑤 ∈ (→→)

**Д(**𝑤 ∈ ⇒ 𝑤 ∈ )**:**

𝑆 ⇒ 𝑤: 𝑤 = , где и ∈ ⇒

в грамматике : ⇒

𝑆 ⇒ => 𝑤 ∈ (→→)

**Теорема**.

Если 𝐿 - КС язык, а 𝑅 – регулярный язык, то 𝐿 ∩ 𝑅 – КС язык.

Док-во:

Пусть - МП-автомат, допускающий язык 𝐿 по заключительному состоянию - ДКА, допускающий язык 𝑅.

Строим МП-автомат ,

((𝑟, 𝑠) , 𝛾) (𝑞, 𝑝) , 𝑎, 𝑋), 𝑠, (𝑟, 𝛾) ∈ 𝑞, a, X)

где 𝑝 =

**Н: Б:** **И:** | по ИП | принцип конфигурации(1)

**Д: Б:**(( ,), 𝑤, ) ((𝑞, 𝑝), 𝜀, 𝛾) | по постр. **И:** (( ,), , ) ((), 𝜀, )=>

(, , ) ((, 𝜀, ) => (, , ) ((, , )

**Теорема**

Пусть , и 𝐿 – КС языки, 𝑅 – регулярный язык. Справедливы следующие утверждения:

1. 𝐿 − 𝑅 является КС языком

2. может не быть КС языком

3. – может не быть КС языком

Док-во:

1. 𝐿 − 𝑅 = 𝐿 ∩

2. Пусть КС-язык ⇒ ∩ = ⇒ полагаем, что отрицание от КС-языка – КС-язык, и знаем, что пересечение КС-языков не всегда КС-язык, но мы получили, что пересечение КС-языков есть КС-язык(противоречие)

3. - КС-язык (он регулярный). Если – - КС-язык, то − 𝐿 – КС-язык, если 𝐿 – КС-язык.

Но − 𝐿= ⇒ - КС-язык (противоречие с 2.)

**Теорема**

Если L и M - регулярные языки, то также регулярен

Док-во: Рег. языкам соответствует РВ | L=L(R) и M=L(S) | = L(R+S)

**Теорема**

Если L - регулярный язык в алфавите Σ, то язык также регулярен.

Док-во: L для ДКА A, для ДКA B | допускающие сост. A - недопуск. в B | цепочка принадлежит одному языку, если не принадлежит другому

**Теорема**

Если язык L регулярен, то язык также регулярен.

Док-во:L определяется РВ E | структурная индукция по длине E, что существует РВ : L()=.| E=+ => = + | E==> = | E= => =()\*

**Теорема**

Если L - регулярный язык в алфавите , а h - это гомоморфизм на , то язык h(L) также регулярен.

**Теорема(Алгоритм Кока-Янгера-Касами)**

Дана НФХ-грамматика 𝐺 = (𝑉, 𝑇, 𝑃, 𝑆) для языка 𝐿. На вход подается цепочка 𝑤 = .

За время 𝑂() алгоритм строит треугольную таблицу, которая говорит, принадлежит ли цепочка языку.

- множество, содержащее переменные, для которых 𝐴 ⇒ .

Если 𝑆 ∈ => S ⇒ 𝑤.

Базис

Для , следует найти в грамматике все правила 𝐴 → .

Индукция

Вычислить в очередной строке.

Любое порождение 𝐴 ⇒ должно начинаться шагом 𝐴 ⇒ 𝐵𝐶:

𝐵 порождает префикс 𝐵 ⇒ ,

𝐶 – суффикс 𝐶 ⇒

Чтобы переменная 𝐴 попала в множество следует найти переменные 𝐵, 𝐶 и целое 𝑘, чтобы

1. 𝑖 ≤ 𝑘 < 𝑗

2. 𝐵 ∈

3. 𝐶 ∈

4. ∃ 𝐴 → 𝐵𝐶 в грамматике 𝐺

Поиск таких переменных требует обработки не более 𝑛 пар множеств .

**Теорема**

Представленный алгоритм корректно вычисляет для всех 𝑖 и 𝑗. Время выполнения алгоритма . *Доказательство*

Элементов таблицы . Мощность множеств и не зависит от 𝑛 ⇒ их сравнение 𝑂(1).

Пар ( не более 𝑛 ⇒