1. Понятие алгоритма. Теория алгоритмов и ее необходимость.

Алгоритм - точное предписание о выполнении в определенном порядке некоторой системы операций для решения задач некоторого данного типа. Также, алгоритм - функциональное выражение, содержащее входные и выходные данные. Математический конгресс в Париже 1900 год 20 алгоритмически неразрешенных проблем. Пример: проблема Гильберта. $\exists P_n = 0$, при решении целых числах, такой алгоритм который дает ответ при $\forall n$? **Необходимость теории алгоритмов** 1) доказательства невозможности алгоритмического решения различных математических проблем 2) с точки зрения практической вычислительной математики или кибернетики, алгоритм - это программа *критерием алгоритмичности* является возможность запрограммировать эту задачу 3) кроме того с точки зрения практической, нужно уметь сравнивать разные алгоритмы для решения одной и той же задачи (по качеству решения, по характеристикам самих алгоритмов, точность)

2. Основные подходы к построению алгоритмов (Уточнения понятия алгоритмов). Алгоритмическая система.

1) теория рекурсивных функций - основана на понятие числовой функции и ее вычислении
2) Теория интерпретирующих систем (автоматы и машины) - алгоритм реализуется некоторым абстрактным, детерминированным устройством (машин Тьюринга)
3) Алгоритм алфавитных преобразований - основа операторы постановки и вывода на множестве слов *(формальные системы на основе формальных грамматик)* Все выше перечисленные уточнения/подходы реализации “Алгоритм” задаются с помощью средств алгоритмической системы Алгоритмическая система - всякий общий способ формального математического задания алгоритма. Основные свойства:
4) Детерминированность - алгоритм всегда четко определен. Один набор входных данных -> единственный набор выходных данных.
5) Массовость - применение множества входных данных
6) Результативность - получение результата за конечное число шагов

3. Определение примитивно-рекурсивной функции.

Примитивно рекурсивная функция - функцию полученная из базисной элементарной функции, функции проекции, функции непосредственного следования, с помощью конечного числа применения оператора суперпозиции и примитивной рекурсии Базовые функции: 1) 0 - функция O^n n-арность. $O^n(x_1, \dots, x\}$ = 0 2) Функция проекции $I^n_m(x_1, \dots^{x_m}, x_n) = x_m$ 3) Функция непосредственно следования $S^1(x) = x + 1$ Предполагается, что все элементы функции являются вычислимыми.

4. Оператор примитивной рекурсии и его использование.

Оператор примитивной рекурсии позволяет строить n+1 местную функцию F по 2-м заданным функциям: 1) $g^n(x_1, \dots, x_n)$ 2) $h^{n+2}(x_1, \dots, x_n)$ $f^{n+1}(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$ $f^{n+1}(x_1, \dots, x_n, 1) = h(x_1, \dots, x_n, 0, f(x_1, \dots, x_n, 0))$ $f^{n+1}(x_1, \dots, x_n, 2) = h(x_1, \dots, x_n, 1, f(x_1, \dots, x_n, 1))$ ... $f^{n+1}(x_1, \dots, x_n, y) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$ Пример: $f^2(x, y) - ?$ $g(x) = 2x$ $h^3(x_1, x_2, x_3) = x_1 + x_2 + x_3$ f(x, 0) = g(x) = 2x $f(x, 1) = h(x, 0, f(x, 0)) = x + 0 + 2x = 3x$ $f(x, 2) = h(x, 1, f(x, 1)) = x + 1 + 3x = 4x + 1$ ... $f(x, y + 1) = h(x, y, f(x, y))$

5. Обоснование недостаточности примитивно-рекурсивных функций.

1) Неопределенность параметра g. Например, когда $\frac{x}{0}$
2) Когда определенно значение k ($g(x_1, \dots, x_n, k)$), но есть такие y > k, при которых выражение тождественно неверное или не определено.
3) Когда функция $g(x_1, \dots, x_n, k)$ - всюду определена, но тождество не выполняется ни на одном наборе аргументов. Недостаточность сводится к ограниченности рекурсии, невычислимости функции и более сложных структурах.

6. Оператор наименьшего корня и его использование

Оператор наименьшего корня (оператор минимизации) μ - позволяет определить новую арифметическую функцию $f^n(x_1, \dots, x_n)$ с помощью ранее известной или построенной функции $g^n(x_1, \dots, x_{n+1})$ Для $\forall$ заданного набора значений переменных, в качестве $f(x_1, \dots, x_n)$ принимается значение *наименьшего* целого неотрицательного корня $y = a$ в уравнении $g^{n+1}(a_1, \dots, a_n, y) = 0$ $[g^{n+1}(x_1, \dots, x_{n+1}) \to^M = f^n(x_1, \dots, x_n)$ $[f^n(x_1, \dots, x_n) = \mu(g^{n+1}(x_1, \dots, x_n, y) = 0)$

$f(x_1, \dots, x_n)$—$? \downarrow g(x_1, \dots, x_n, 0) = ?$ если $\neq 0$, то $\downarrow g(x_1, \dots, x_n, 1) = ?$ ... $g(x_1, \dots, x_n, y) = 0$ пока не найдем корень уравнения Такой процесс вычислений может *не* привести к результату

7. Частчино-рекурсивные функции. Общерекурсивные функции. Частично-рекурсивные функции. Классификация рекурсивных функций.

ПРФ $\subset$ ОРФ ОРФ $\subset$ ЧРФ 1) ПРФ (примитивно-рекурсивная функция) - функция, которая может быть определена с помощью базовых функций и операторов. 2) ОРФ (общерекурсивная функция функция) - всюду определенная ЧРФ 3) ЧРФ (частично-рекурсивная функция) - функция, построенная из элементарных функций, оператора суперпозиции, оператора примитивной рекурсии и оператора наименьшего корня.

8. Значение рекурсивных функций. Тезис Черча

Значение: 1) Рекурсия позволяет разбивать задачу на более простые подзадачи 2) Используется для определения последовательностей, функций и операторов (например: *, /, x^n) 3) Представление множества алгоритмов (например: обход дерева) 4) Основа функционального программирования (без изменения состояний и использования циклов). **Тезис Черча**: Класс алгоритмично- или машинно-вычислимых частично числовых функций, совпадает с классом всех ЧРФ (это *не* теорема).

9. Определение и принципы функционирования машины Тьюринга

Машина Тьюринга - абстрактный автомат, задаваемый картежом из 4 параметров: Q - внутренний алфавит состояний, A - внешний алфавит символов, K_0 - стандартное состояние и P - программа. | q_{i1} | q_{i2} | q_{i3} | ... | ... | ... | ... | q_i | ... | ... | ... | ... | ... | q_{iu} | ....q_i $\uparrow \updownarrow$ q_j $\leftarrow_L \uparrow^E \rightarrow_R$ ....A Q ....YY.......$\downarrow \longrightarrow$ $\longrightarrow \uparrow$

Принцип функционирования: 1) В каждом такте МТ гаходится в одном из своих состояний (Q). А головка обозревает одну ячейку. 2) В следующий такт МТ переходит в другое состояние или остается в том же. 3) В обозреваемую ячейку записывается символ алфавита А или остается тот же. 4) Головка передвигается влево (L), вправо (R) или остается на месте (E).

10. Способы задания МТ

M = $< \quad Q, A, U_0, P \quad >$ Q = $\{q_0, q_1, q_2, q_3\}$ A = {a, b, c, λ} $K_0 = q_0 abc\lambda$ Способы: 1) Совокупность команд: P = $\{q_0 a \rightarrow q_1 \lambda R;\ q_1 b \rightarrow q_2 \lambda R;\ q_2 c \rightarrow q_2 \lambda E\}$ 2) Таблица переходов

Q / A	a	b	c	λ
q_0	$q_1 \lambda R$			
q_1		$q_2 \lambda R$		
q_2			$q_2 \lambda E$	

3) Диаграмма переходов (блок-схема/граф)

11. Вычисления на МТ

A = $A \cup A \cup A$ В соответствии с определенной конфигурацией, ко всякой не заключительной конфигурации применима некоторая команда. После такого применения, МТ переходит в новую конфигурацию $K_i \rightarrow_M K_{i+1}$ Пусть f - функция, приводящая A к A МТ правильно вычисляет, если: 1) $f(V) = W; q_0 V \Rightarrow_M q_z W$ 2) $\exists V : f(V)$ неопределенно, то МТ работает бесконечно Две МТ являются эквивалентными, если они правильно вычисляют одну и ту же функцию.

12. Тезис Тьюринга и его связь с тезисом Черча

Тезис Тьюринга: Всякий алгоритм может быть реализован на МТ ИЛИ Если невозможно создать МТ для какой-либо функции, то *не* существует алгоритма для этой функции. **ДОКАЗАТЕЛЬСТВА НЕТ**

Связь: 1) Эквивалентность(если функция вычислима в одной модели, то она вычислима и в другой) 2) Оба описывает вычислимость и ее пределы

13. Определение формальной системы

Формальными системами называется кортеж FS = < A, A_1, R >, где A - конечный алфавит формальной системы; A_1 - множество аксиом, то есть множество правильно построенных выражений в алфавите A; R - набор правил

14. Определение системы (полусистемы) Туэ. Примеры систем.

Полусистема Туэ - определяется как формальная система через: 1) алфавит A; 2) конечное множество подстановок вида $\alpha_i \rightarrow \beta_i; \alpha_i, \beta_i \in A$ Система Туэ - система подстановок. Полусистемы - подстановка только правых частей (работает в одну сторону).

Пример: A = { a, b, c } R = {a -> ccba bc -> aa ca -> ∅ acb -> c cc -> b} Из "a" выводится любая строка, а из "b" и "c" не выводится ничего **Свойства системы Туэ:** 1) Рефлексивность $\forall(a \in A\star)(\alpha \leftrightarrow a)$ 2) Транзитивность $\forall(\alpha, \beta, \gamma \in A\star)(\alpha \leftrightarrow \beta\ \beta \leftrightarrow \gamma) \Rightarrow (\alpha \leftrightarrow \gamma)$ 3) Симметричность $\forall(\alpha, \beta \in A\star)(\alpha \leftrightarrow \beta) \Rightarrow (\beta \leftrightarrow \alpha)$ **Для полусистемы Туэ таких свойств *НЕТ***

15. Определение канонической системы Поста

Каноническая система Поста - абстрактный автомат, задаваемый кортежем из 4 параметров: A - собственный алфавит X - алфавит переменных A_1 - множество аксиом ($A_1 \in A, X$) R - множество правил

FS_p = < A, X, A_1, R > A = { 1 } X = { x } A_1 = { 1 } R = { x $\rightarrow$ 11x } Для нечетных 1, 111, 11111 Если для четных, то меняем только аксиому A_1 : $A_1 = 11$

11111 x = 1111

16. Неклассические алгоритмические системы. Виды и применение.

Виды: 1) Операторные алгоритмы Ван Хао i: | w | α | β | i - номер приказа; w - операция над объектом; α, β - номера дальнейших приказов 2) Операторные алгоритмы А. А. Ляпунова p (x < y), если истинно, то выполняется, если ложно, то не выполняется 3) Блок-схемный метод алгоритмизации Распределение решения задачи на отдельные этапы (блоки) 4) Логические схемы алгоритмов Ю. И. Янов Логические выражения - основа алгоритма Янова. Потоки данных проходят через блоки.

17. Разрешимые и перечислимые множества. Их применимость.

Множество М *разрешимо*, если существует некоторый алгоритм A_M, который по любому объекту a дает ответ: принадлежит ли объект a множеству М. Множество М *разрешимо*, если оно обладает **общерекурсивной** (всюду определелнной) характеристической функцией. Множество М *перечислимо*, если это множество значений некоторой **общеркурсивной** функции.

18. Алгоритмическая проблема неразрешимости. Источники возникновения, сущность и способы устранения.

Относится к задачам, для которых не существует алгоритма **Источники возникновения**: 1) Теоретические ограничения 2) Временные и ресурсные ограничения 3) Недостаток информации **Сущность:** Для некоторых задач не существует общего алгоритма, который мог бы решить все случаи задачи **Способы устранения:** 1) Ограничения области 2) Приближенные алгоритмы 3) Анализ задачи

19. Определение конечного автомата. Примеры.

КНА - это абстрактный автомат, задаваемый кортежем из 5 элементов: X - входной алфавит Q - внутренний алфавит (состояний) U - выходной алфавит δ - функция перехода λ - функция выхода

α β $\longrightarrow$S$\longrightarrow$ $\alpha \in x\star$ $\beta \in U\star$

КНА 1-го рода (Мили) q(i) = δ(x(i), q(i-1)); зависит от q и X U(i) = λ(x(i), q(i - 1)); КНА 2-го рода (Мура) q(i) = δ(x(i),q(i - 1)); зависит от q u(i) = λ(q(i)) = λ(δ(x(i), δ(i-1))) Пример: S = < X, Q, U, δ, λ > X = { a, b, c, d } Q = { q_1, q_2, q_3 } U = { 0, 1 }

$\frac{\triangle}{\triangle}$ =

Q / X	a	b	c	d
q_1	$q_2/0$	$q_3/0$	$q_3/1$	$q_1/1$
q_2	$q_1/0$	$q_2/1$	$q_1/1$	$q_2/0$
q_3	$q_3/1$	$q_1/1$	$q_2/0$	$q_1/0$

20. Способы задания конечно автомата. Примеры.

Способы: 1) Матрица переходов или матрица выходов

Q/X	x_1	x_2
q_1		$q_2/0$
q_2		
...		

2) Графический . X_k/U_m $(q_i) \longrightarrow (q_j)$
3) Автоматная матрица

Q / Q	q_1	q_2
q_1		a/0
q_2		
...		

21. Основные свойства конечных автоматов (инициативность, полнота, детерминированность).

1) Инициативность (начальное состояние) - КНА всегда имеет начальное состояние q_0
2) Полнота - КНА, который имеет всюду определенные функции переходов и выходов
3) Детерминированность - КНА, у которого для каждого состояния и каждого входного символа существует не более одного перехода.
4) Недетерменированность
5) Кортеж из 5 параметров: S = < X, Q, U, δ, λ >

22. Процедуры синтеза и анализа конечного автомата. Определение и взаимосвязь.

Синтез КНА - хто построение КНА по описанию множества слов во входном алфавите, которые допустимы КНА.

Анализ КНА - это процесс обратный к *синтезу* - получить множество входных слов, которые допустимы КНА. **Взаимосвязь:** С помощью одного из этих процессов можно проверить корректность другого.

23. Определение регулярных выражений. Примеры.

Регулярное выражение - это последовательность символов. R X = { x_1, ... , x_n} определяется рекурсивно следующим образом: 1) R = $\emptyset$ 2) $\alpha \in X \Rightarrow R = \alpha$ 3) R_1, R_2 и $\vee, \wedge$, < >, то 1) $R = (R_1 \cup R_2)$ 2) $R = (R_1 \cap R_2)$ 3) $R = (R_1)\star; R =< R_2 >$ 4) Другое *не* является регулярным выражением Примеры: R = (a $\vee$ b) c $\Leftrightarrow$ L(x) = {ac, bc} R = (a $\vee$ b)* c $\Rightarrow$ L(R) = {aabc, c, bbc}

24. Регулярные выражения и примеры их использования. Определяемые регулярными выражениями множества.

Примеры использования: 1) Ограничения на ввод символов в сферах (пароли, почта, имена, номера телефонов) 2) Состояния системы или ее подсистемы (космические аппараты, генераторы, диспетчеры задач) Определяемые регулярными выражения множества (23 вопрос)

25. Регулярные выражения и их разметка

Разметка: X = {x, y, z}; R = (z v x < y v z >); R = | (| z | v | x | < | y | v | z | > |) |; . 0 1 2 3 4 5 6 7 8 9 10

26. Правила подчинения мест в регулярных выражений

1) Начальные места всех термов или букв, или символов многочлена, помещенные в “()” или “<>”, подчинены месту, расположенному слева от открывающей скобки | (| a | ∨ | b |) | ↘ ↘

2) Место, располагающее справа от закрывающей скобки, подчинено конечным местам всех термов многочлена, заключенных в эти скобки. А в случае “< >” еще и месту слева от открывающей скобки | (| a | ∨ | b |) | | < | a | ∨ | b | > | . ↘ ↘ ↘ ↘ ↘

3) Начальные места всех термов многочлена заключены в “< >”, подчиненны месту расположенному справа от закрывающей скобки | < | a | ∨ | b | > | . ↙↙

4) Если место “c” подчиняется месту “b”, а место “b” подчиняется месту “a”, то место “c” подчиняется месту “a”

5) Каждое место подчиняется самому себе

6) Других случаев подчинения в регулярных выражениях нет.

X - алфавит (без вспомогательных символов) Основным местом называют место, слева от которого находится символ алфавита X, а также начальное место 0 1 2 3 4 5 | (| a | v | b |) |

Место справа будет предосновным (выше 1, 3)

27. Правило отметки состояний регулярного выражения. Теорема обоснования допустимости входных слов конечным автоматом.

Все состояния регулярного выражения, включая начальное (нулевое), должны быть помечены. Такие места называются основными и нумеруются по порядку, начиная с нуля и начального состояния. **Теорема** Входное слово является допустимым, если начальное место регулярного выражения связано с конечным местом этого же регулярного выражения.

28. Правила алгоритма синтеза конечного автомата по заданному множеству регулярных выражений и их использование.

1) Разметить все места $R_1,\ \dots,\ R_N$
2) Отметить основные места
3) Выстроить все зависимости неосновных мест (a) $\rightarrow$ (b)
4) Строится таблица переходов

29. Операции на множестве конечных автоматов. Определение гомоморфизма на множестве конечных автоматов.

$S_1 =< X_1, Q_1, U_1, \delta_1, \lambda_1 > S_2 =< X_2, Q_2, U_2, \delta_2, \lambda_2 >$ Отображение (связь между элементами множеств (a)): g = < g_1, g_2, g_3 > {$g_1 : X_1 \rightarrow X_2$ после перехода совпадают и символы, и состояния, и выходы {$g_2 : Q_1 \rightarrow Q_2$ {$g_3 : U_1 \rightarrow U_2$

Отображение - это концепция, описывающая связь двух множеств, основанное на задавании правила сопоставления.

Гомоморфизм - концепция, описывающая связь между 2-мя КНА с помощью отображения Пример: A = { $a_1,\ \dots,\ a_n$ } B = { $b_1,\ \dots,\ b_n$ } b_1: A -> A отображает элементы множества внутри этого множества b_2: B -> B g: A -> B отображает элементы множества A в элементах множества B

Изоморфизм - отношения h_1 и h_2 - взаимооднозначны (движение в разные стороны). Это частный случай гомоморфизма.

30. Неотличимость и эквивалентность состояний КНА

Неотличимость или эквивалентность - если для любого состояний q’ КНА S_1 существует неотличимое состояние q’’ КНА S_2 и наоборот, то S_1 и S_2 неотличимы или эквивалентны. Задача минимизации КНА S - это задача КНА S_0, эквивалентного S, но имеющего *наименьшее* количество состояний.

31. Формулировка задача минимизации КНА. Теорема существования минимального КНА.

Задача минимизации КНА S - это задача КНА S_0, эквивалентного S, но имеющего *наименьшее* количество состояний. **Теорема** Для любого КНА S всегда существует минимальный КНА S_0, единственный, с точностью до изоморфизма

32. Описание алгоритма минимизации КНА

Алгоритм - 1) q’, q’’ $\in Q \rightarrow Q_1$; Относим в один класс, если выполняется следующее соотношение $\forall x \in X : \lambda(q', x) = \lambda(q'', x)$ - i+1) q’, q’’ $\in Q_i; \rightarrow Q_{i+1}$ $\forall x \in X : \delta(q', x), \delta(q'', x) \in Q_i, l$ В конце каждого i + 1 шага выполняется проверка условия: 1) (i+1) шаг не изменяется $\forall j[Q_{i,j} = Q_{i+1,j}]$, то алгоритм заканчивает работу 2) Переход к следующем шагу S: X = {a, b, c}; Q = $\{q_1, ..., q_g\}$ U = {0, 1}

$\frac{\triangle}{\triangle}$=

Q / X	a	b	c
1	2/0	4/1	4/1
2	1/1	1/0	5/0
3	1/1	6/0	5/0
4	8/0	1/1	1/1
5	6/1	4/1	3/0
6	8/0	9/1	6/1
7	6/1	1/1	3/0
8	4/1	4/0	7/0

Q / X	a	b	c
9	7/0	9/1	7/1

1) (1, 4, 6, 9) (2, 3, 8) (5, 7) Это классы эквивалентных состояний
2) (1, 4, 6) (2, 3, 8) (5, 7) (9)
3) (1, 4) (2, 3, 8) (5, 7) (9) (6)
4) (1, 4) (2, 8) (5, 7) (9) (6) (3)
5) —-//—- Q_0 = $\{q_1^o, q_2^o, q_3^o, q_4^o, q_5^o, q_6^o\}$

Q / X	a	b	c
1	2/0	1/1	1/1
2	1/1	1/0	3/0
3	5/1	1/1	6/0
4	3/0	4/1	3/1
5	2/0	4/1	5/1
6	1/1	5/0	3/0

33. Теорема Клини о синтезе и анализе КНА

Проблема синтеза: Для любого регулярного выражения R всегда существует КНА, допускающий все множество входных слов и только его, представимое этим регулярным выражением. **Проблема анализа:** Для любого КНА допускается такое множество входных слов, которое может быть представлено некоторым регулярным выражением R.

34. Описание алгоритма анализа КНА

Алгоритм: (j) - номер такта = 0, 1, 2, ... ($x^j \in X$) - входной символ на j-ом такте ($q^j \in Q$) - состояние на j-ом такте ($u^j \in U$) - выходной символ на j-ом такте α - выходное слово НАЧАЛО 0) j = 0; $q^j = q_0$; $x^j = \emptyset$; - автомат стоит $u^j = \emptyset$; 1) j = j + 1 2) $q^j = \delta(x^j, q^{i-1})$ 3) $u^j = \lambda(x^j, q^{i-1})$ 4) j != $|\alpha|$ -> на п.1 5) если $u^j \in U_F$ $\alpha \in L(S)$

Q / X	X	Y
1	2/1	3/0
2	2/1	3/0

Q / X	X	Y
3	4/2	3/0
4	4/2	5/3
5	4/2	5/3

R_1 = < x > < y > xxxy

j	x^j	q^j	u^j
0	-	1	-
1	x	2	1
2	x	2	1
3	x	2	1
4	y	3	0

35. Определение формальной грамматики

Формальная грамматика G - это кортеж < N, T, R, S >, где N - конечное непустое множество не терминальных, вспомогательных символов; T - конечное непустое множество терминальных (основных), причем (N ∩ T = ∅): R - конечное множество упорядоченных пар α, β; R = $\{(\alpha, \beta)\}$ = $\{\alpha \rightarrow \beta\}$: S - начальный символ аксиом для системы, $S \in N$ Свойства системы: - рефлексивность; - антисимметричность; - транзитивность $R = \{\alpha \rightarrow \beta\} | \alpha \in (N \cup T)^* * N * (N \cup T)^*$ $\beta \in (N \cup T)^*$ "->" - выводимость "*" - множество всех подмножеств Пример: N = { S } - S - начальный символ T = { a, b }; R = { S -> aS, S -> b } Генерация строк S -> b S -> aS -> ab S -> aS -> aaS -> aab Продукция - это правило, как один символ заменяется на последовательность символов.

36. Выводимость в формальных грамматиках. Дерево (граф) вывода.

Определение выводимости

Будем говорить, что из некоторого слова γ выводимо некоторое слово ϵ ($\gamma \Rightarrow_\alpha \epsilon$), причем каждое $\gamma, \epsilon \subset (N \cup T)^*$

{$\gamma = \delta_1 \alpha \delta_2$ {$\epsilon = \delta_1 \beta \delta_2$ $(\alpha \rightarrow \beta) \in R$

Будем говорить, что γ выводимо из ϵ, если существует такие $\gamma_0 = \gamma$ и $\gamma_n = \epsilon$ Последовательность называется выводом длины n

$\gamma \Rightarrow_G^* \epsilon$ $\gamma_0 = \gamma$; $\gamma_n = \epsilon$ $\gamma_0 \Rightarrow_G \gamma_1 \Rightarrow_G \gamma_2 ... \Rightarrow_G \gamma_n = \epsilon$; $\gamma_0, \gamma_1, ..., \gamma_n$ - вывод n-малого **Пример:** G = < T, N, R, S > T = { a, b, c, *} N = { S, A } R = { S -> S * S, (1) S -> A + A, (2) S -> A, (3) A -> A * A, (4) A -> a} (5) $S \Rightarrow^2 A + A \Rightarrow^4 A \cdot A + A \Rightarrow^5 a \cdot A + A \Rightarrow^5 a \cdot a + A \Rightarrow^5 a \cdot a + a$;

D - помеченное дерево (граф вывода), если 1) корень помечен S, 2) $D_1, ..., D_n$ - поддерево, то корень каждого D_i помечен либо $A_i \in N$ (больше одной вершины у D_i), либо $a_i \in N$ (одна вершина у D_i) **Замечание:** Каждое D_i - дерево вывода в грамматике G = < T, N, R, A_i >, в котором A_I - аксиома

Пример: S A + A A * A a a a

37. Свойства формальных грамматик (однозначность, неоднозначность). Примеры формальных грамматик, обладающих заданными свойствами.

Грамматика называется **однозначной**, если каждая строка, принадлежащая языку, может быть выведена из стартового символа **единственным способом** (единственное дерево вывода). $(a^n b^n | n \geq 0)$ N = { S } T = { a, b }; R = { S -> aSb } S -> E_s (пустая строка) . S / |
a S b /|
a S b . | . E aabb

Грамматика называется **неоднозначной**, если существует хотя бы одна строка, которая может быть выведена разными способами (имеет более одного дерева вывода). a+a * a N = { S } T = { a, +. * } R = {S -> S +S S -> S * S S -> a}

Свойство однозначности характеризует грамматику, а не язык, поскольку один и тот же язык может быть описан различными грамматиками

38. Определение формального языка.

Языком L(G), порожденным грамматикой G, называется множество всех строк, которые могут быть получены из стартового символа S, с помощью применения правил продукции R.

39. Виды классификации формальных грамматик. Распознающие, порождающие, преобразующие формальные грамматики.

О. Распознающие

позволяет ответить на вопрос, является ли цепочка правильной (принадлежит или нет?) ###### О. Порождающие Позволяет строить любую правильную цепочку, давая при этом описание структуры, и не дает строить неправильные цепочки ###### О. Преобразующие Для любой правильной цепочки, позволяет строить отображение её, задавая порядок реализации

Распознающая - известна цепочка (a * a + a), строим цепочку. Порождающая - используем грамматику, строим все цепочки. Примитивно-рекурсивная функция и машина Тьюринга - это тип 0

40. Виды классификации формальных грамматик. Классификация грамматик по Н.ХОМСКОМУ

Тип	Название	Вид порождающих правил
3	Регулярные	*A* -> *aB*, A -> *a*N = { A, B }; T = { a } Самые строгие правила
2	КС (контекстно-свободные)	*A* -> *aB* N = { A, B }, T = { a }правила не зависят от контекста
1	НС, КЗ (контекстно-зависимые)	$\alpha\ A\ \beta \rightarrow \alpha bC\beta$N = { A, B }; T = { a, b, c }зависят от контекста
$\emptyset$	рекурсивно-избыточные (РИ)	Нет ограничений

Примеры: Регулярные - конечные автоматы Контекстно-свободные - автомат с магазинной памятью Контекстно-зависимые - физические законы Рекурсивно-избыточные - машина Тьюринга

41. Определение иерархии формальных грамматик и языков.

$3 \subset 2 \subset 1 \subset 0$

42. Формальные свойства грамматик и языков.

№	Название	Типы ФГ			
		3	2	1	0
1	$L_G = L(G) = \emptyset$	+	+	-	-
2	|L(G)| = ∞	+	+	0	0
3	L(G) = T*	+	-	-	-
4	$L(G_1) \leq L(G_2)$?	+	-	-	-
5	$L(G_1) \equiv L(G_2)$?	+	-	-	-
6	$L(G_1) \cap L(G_2) = \emptyset$	+	-	-	-
7	$\forall \alpha, \beta \in (T \cup N)\star : \alpha \Rightarrow \beta$	+	+	+	-
8	$\exists \alpha : \alpha \rightarrow \geq 1$	+	-	-	-
9		да	-	?	да

43. Автоматные грамматики и языки. Их связь с конечными автоматами.

Теорема Если G грамматика типа 3, то всегда существует КНА S такой, что допускаемое им множество входных слов в точности совпадает с множеством слов, заданной грамматикой G типа 3 G = < N, T, R, S > S = < A, Q, U, δ, λ > 1) T $\Rightarrow$ A 2) N $\Rightarrow$ Q 3) S $\Rightarrow$ q_0 4) $[B \rightarrow aD] \Rightarrow [\delta(a, B) = D]$ 5) $[B \rightarrow a] \Rightarrow [\delta(a, B) \in F]$

44. Синтез (восстановление) автоматных грамматик.

Есть обучающий язык L* -> Строится КНА S -> Минимизация (при необходимости) -> Построение ФГ - > Проверка

45. Анализ автоматных грамматик.

1-й подход: G -> S -> {$\alpha \in L(G)$. $\uparrow$ {$\alpha \notin L(G)$. α Строим КНА по грамматике и подаем слово α 2- подход: S -> {$\alpha \in L(G) \uparrow$ {$\alpha \notin L(G)$ α К уже существующему КНА подаем слово α

46. КС грамматики и языки. Нормальная форма ХОМСКОГО

G = < N, T, R, S > Нормальная форма Хомского: 1) A -> BC - два не терминальных символа 2) A -> a -один терминальный символ Преимущества: 1) КС грамматика легко преобразуется к НФХ 2) Удаление ненужных объектов Другими словами, все подобные правила продукции записываются в одно, с разными выходами.

47. КС грамматики и языки. Нормальная форма ГРЕЙБАХ.

G = < N, T, R, S > Нормальная форма Грейбах: 1) $A \to \alpha\alpha$ $A \in N$; $\alpha \in T$; $\alpha \in (N \cup T)\star$ α - либо символ, либо не терминальный символ $G \to G \leftrightarrow G$

48. Анализ и синтез КС грамматик.

Подходы: 1) ФГ преобразует к НФХ, либо к НФГ. Подаем слово α $\alpha \in L(G)$? 2) ФГ преобразуем к НФХ, либо к НФГ. И добавляем автомат с магазинной памятью, который хранит память в стеке.

49. Определение автомата с магазинной памятью.

КНА с магазинной памятью - это мат. модель, которая расширяет возможности обычного КНА, используя стек (магазин)

M = < Q, E, Г, δ, q_0, Z_0, F > Q - конечное множество состояний E - конечный алфавит входных символов Г - алфавит магазина δ - функция перехода q_0 - начальное состояние Z_0 - начальный символ в магазине F - множество финальных состояний

50. Функционирование автомата с магазинной памятью.

Пример: l = ($a^n b^n | n \geq 0$) Q = { q_0, q_1 }, E = { a, b } Γ = { A, Z } Z_0 = { Z } F = { q_1 } S = { $(q_0, a, Z) \to (q_0, AZ)\}$ - при чтении a помещаем A в стек $(q_0, a, A) \to (q_0, AA)$ - при чтении a помещаем еще A в стек $(q_0, b, A) \to (q_0, E)$ - при чтении b удаляем верхнее A $(q_0, b, Z) \to (q_1, Z)$ - если читаем b, а на вершине Z, то переходим в финальное состояние $(q_1, b, Z) \to (q_1, Z)$ - конец aaabbb 1) q_0 ; Z 2) a ; AZ 3) a ; AAZ 4) a ; AAAZ 5) b ; AAZ 6) b ; AZ 7) b ; Z