**1.** Синтаксис языков описывается:

**2.** Грамматика языка реализуется:

**3.**  G = (T, V, P, S0),

**4**. Языком, порождаемым грамматикой G = (T, V, P, S0), называется

**5.** α = defgabck : | α | = ?

**6.** Грамматика: S→0|0A  A→1B  B→0|0A

**7.** Язык: {0(01)n | n³0}

**8.** Распознаватель: {0(01)n | n³0}

**9.**  KA = (Q, å, d, q0, F),

**10.** Л**емма о накачке доказывает     
11. КА задается**

**12.** Конфигурация КА

**13.** L(КА) распознаваем КА

**14.** Правила КС-грамматики имеют вид

**15.** Бесполезные символы КС-грамматики **1.** устраняются  **2**. Недостижимы

**16.** G = (T, V, P, S), в нормальной форме Грейбах, если

**17.** Нормальная форма Xомского позволяет упростить рассмотрение свойств:

**18.** G = (T, V, P, S) в нормальной форме Xомского, если каждое правило из P имеет один из следующих видов:

**19.** Нетерминал КС-грамматики *рекурсивен*

**20.** МП = (Q, å, Г, d, q0, z0, F)

**21.**Конфигурация МП-автомата:

**22.** Детерминированным МП-автомат

**23.   1.** FIRST(A) =

**24.** Конфигурация “1-предсказывающего” алгоритма разбора имеет вид  (x, Xα^, π), где x – неиспользуемая часть входной цепочки, Xα – цепочка в магазине

**25.** Управляющая таблица применяется для распознавания

**26.** Для LR(k) языка можно построить анализатор

**27.** В LR(k) разборе применяются операции

**28.**  goto задается

**29.** Дляграмматики предшествования строится

**30.** LR(k)-алгоритм разбора описывается

**31.** G = (T, V, P, S) - грамматика типа 0 если

**32.** Конфигурация машины Тьюринга

**33.** G= (T, V, P, S) – *неукорачивающаяся грамматика*, если каждое правило из P имеет вид α → β, где

**34.** G = (T, V, P, S) - *контекстно-зависимая (КЗ)*, если каждое правило из P имеет вид α → β, где

**35.** *Грамматику типа 1* можно определить как

**36.** Любая регулярная грамматика является

**37.** Любая КС-грамматика является грамматикой типа

**38.** Любая КС-грамматика является  грамматикой типа

**39.** Любая КЗ-грамматика является грамматикой типа

**40.** Любая неукорачивающая грамматика является грамматикой типа

**41.** Каждый регулярный язык является

**42.** Каждый КС-язык является

**43.** Каждый КЗ-язык является языком типа

**44.** Определите  понятие семантики языков программирования

**45.** Простейшие переводы описываются:

**1. Синтаксис языков описывается:**

порождающими и распознающими системами

**2. Грамматика языка реализуется:**

ка?

**3.  G = (T, V, P, S0),**

где Т - конечное множество терминальных символов (терминалов) алфавита;

V – конечное множество нетерминальных символов алфавита, не пересекающихся с T

S0 - начальный символ (или аксиома), S0 ∈V;

P - конечное множество правил порождения (продукций), P = (T˅V)+ × (T˅ V)\*.

**4. Языком, порождаемым грамматикой G = (T, V, P, S0), называется**

множество L(G)={α ⊂T\* | S0 => α}.

Другими словами, L(G) - это все цепочки в алфавите T, которые выводимы из S0 с помощью P.

**5.  α = defgabck : | α | = ?**

длина = 8 используется в лемме о накачке

**6. Грамматика: S→0|0A  A→1B  B→0|0A**

{0(01)n | n>=0} регулярная

**7. Язык: {0(01)n | n³0}**

порожден грамматикой выше    

**8.  Распознаватель: {0(01)n | n³0}**

**…**

**9.  KA = (Q, å, d, q0, F),**

Q - конечное множество состояний;

а - конечный алфавит входных символов;

d - функция переходов, задаваемая отображением

d:Q\*a→ Q,

где Q - конечное множество подмножеств множества Q;

q0 - начальное состояние автомата, q0∈Q;

F ⊂ Q - множество заключительных состояний.

**10. Лемма о накачке доказывает**

что все достаточно длинные цепочки регулярного языка можно *накачать*, то есть **повторить** внутреннюю часть цепочки символов сколько угодно раз, производя новую подцепочку, также принадлежащее языку.

служит инструментом для доказательства нерегулярности некоторых языков.

**11. КА задается**

**таблицей и диаграммой переходов**

**12. Конфигурация КА**

пара множества (q, ω)∈Q×E\*, где q ∈Q, ω∈E\*. Конфигурация (q0, ω) называется *начальной*, а (q, ε), где qF, - заключительной. пара множества (q, ω)

**13. L(КА) распознаваем КА**

называется множество входных цепочек,

L(КА) ={ω ∈E\*| (q0,ω)|- k (q, ε), где q∈F}

**14. Правила КС-грамматики имеют вид**

A → α, где AV, α ⊂(T˅V)\*.

**15. Бесполезные символы КС-грамматики 1. устраняются  2. Недостижимы**

1). *Непроизводящих* и

2). *Недостижимых* символов;

3). преобразования в грамматику без ε–правил;

4). устранения цепных правил

**16. G = (T, V, P, S), в нормальной форме Грейбах, если**

если в ней нет ε-правил, т.е. правил вида A→ ε, и каждое правило из P отличное от S → ε, имеет вид A→ aα, где a T, α V\*.

**17.  Нормальная форма Xомского позволяет упростить рассмотрение свойств:**

кс-грамматик

P имеет один из следующих видов:

**18. G = (T, V, P, S) в нормальной форме Xомского, если каждое правило из P имеет один из следующих видов:**

1. A→ BC, где A, B, C∈V;

2. A→ a, где a∈T;

3. S→ ε, если ε∈L(G), причем S не встречается в правых частях правил.

**19. Нетерминал КС-грамматики *рекурсивен***

если A =>+ αAβ, для некоторых α и β. Если α = ε, то A называется *леворекурсивным*, если β = ε, то A называется *праворекурсивным*.

**20. МП = (Q, å, Г, d, q0, z0, F)**

Q – конечное множество состояний устройства управления;

f - конечный алфавит входных символов;

Г - конечный алфавит магазинных символов;

d - функция переходов, отображает множества Q×(a˅ {ε})×Г в множество конечных подмножеств множества Q×Г\*;

q0 - начальное состояние, q0∈Q;

z0 - начальный символ магазина, z0∈ Г;

F - множество заключительных состояний, F ∈Q.

**21.Конфигурация МП-автомата:**

тройка (q, ω, z)Q×E\*×Г\*, где

q – текущее состояние управляющего устройства;

ω – необработанная часть входной цепочки (первый символ цепочки ω находится под входной головкой; если ω = ε, то считается, что вся входная цепочка прочитана) ;

z – содержимое магазина (самый левый символ цепочки z считается верхним символом магазина; если z = ε, то магазин считается пустым).

**22. Детерминированным МП-автомат**

**Детерминированным автоматом с магазинной памятью** (англ. *deterministic pushdown automaton*) называется [автомат с магазинной памятью](https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D1%8B_%D1%81_%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C%D1%8E), для которого выполнены следующие условия:

1. 8*q*∈*Q*,*a*∈Σ∪{*ε*},*X*∈Γ⇒*δ*(*q*,*a*,*X*)

имеет не более одного элемента — *δ*:*Q*×Σ∪{*ε*}×Γ→*Q*×Γ∗

 .

 Если *δ*(*q*,*a*,*X*) непусто для некоторого *a*∈Σ, то *δ*(*q*,*ε*,*X*) должно быть пустым.

**23.   1. FIRST(A) =**

{a∈T | A =>+i aβ ∪{ε if A⇒∗ε}, где β(T˅V)\*}

это множество первых терминальных символов, которыми начинаются цепочки, выводимые из нетерминала AV

**24. Конфигурация “1-предсказывающего” алгоритма разбора имеет вид  (x, Xα^, π), где x – неиспользуемая часть входной цепочки, Xα – цепочка в магазине**

(x, Xa┴, π), где x – неиспользуемая часть входной цепочки, Xa – цепочка в магазине, а π – цепочка на выходной ленте, отражающая состояние алгоритма.

**25. Управляющая таблица применяется для распознавания**

цепочки?

управляет работой алгоритма. М задает отображение множества (Vp T {})× (T{ε}) в множество, состоящее из следующих элементов:

1. (β, i), где βVp\* правая часть правила вывода с номером i.

2. ВЫБРОС.

3. ДОПУСК.

4. ОШИБКА.

**26. Для LR(k) языка можно построить анализатор**

Грамматики, для которых можно построить детерминированный восходящий анализатор, называются LR(k)-*грамматиками* (входная цепочка читается слева (Left) направо, выходом анализатора является правый (Right) разбор, k-число символов входной цепочки, на которое можно “заглянуть” вперед для выделения основы).

**27. В LR(k) разборе применяются операции**

перенос свертка ошибка допуск

**28.  goto задается**

Функция GOTO(*I,X*) определяется как замыкание множества всех ситуаций [ *A→ αX∙β* ], таких что [ *A→ α∙Xβ*] ϵ *I*

Если *I-*множество ситуаций, допустимых для некоторого активного префикса γ , то GOTO(*I,X*) – это множество ситуаций, допустимых для активного префикса γ*X.*

Аргументами функции GOTO(*I,X*) являются множество ситуаций *I* и символ грамматики *X.*

**29.  Для грамматики предшествования строится**

матрицу предшествования грамматики. Строки матрицы предшествования помечаются первыми (левыми) символами, столбцы — вторыми (правыми) символами отношений предшествования. В клетки матрицы на пересечении соответствующих столбца и строки помещаются знаки отношений. При этом пустые клетки матрицы говорят о том, что между данными символами нет ни одного отношения предшествования.

**30.  LR(k)-алгоритм разбора описывается**

КС-грамматика G = (T, V, P, S) называется LR(k)- грамматикой для k ≥ 0, если из существования двух правых выводов для *пополненной* грамматики G' = (T, V', P', S') полученной из G:

S' =>r\*αAw =>r αβw,

S' =>r\*γBx =>r αβy,

для которых FIRSTk(w) = FIRSTk(y) следует, что αAy= γBx.

Способ использования расширенного магазина состоит изтрехшагов:

Шаг 1. Определение активных префиксов;

Шаг 2. Построение управляющей таблицы;

Шаг 3. Применение алгоритма «перенос-свёртка».

**31. G = (T, V, P, S) - грамматика типа 0 если**

|  |  |  |  |
| --- | --- | --- | --- |
| Общего вида, не накладывается никаких ограничений | S0→CD, C→ 0CA,  C→ 1CB,AD → 0D,  BD→1D, A0 → 0A,  A1→1A, B0 → 0B,  B1→ 1B, C → ε,  D → ε | {ωω|ω{0,1}\*, язык состоит из цепочек четной длины, из 0 и 1} | Машина Тьюринга |
|  | | | |

**32. Конфигурация машины Тьюринга**

Конфигурацией машины Тьюринга назовем тройку(q,α,i), где q∈Q — текущее состояние машины Тьюринга; α∈(Γ\{B})\* — строка, яв-ляющаяся непустой частью ленты; i — целое, определяющее позицию головки ленты, отсчитываемую от левого конца ленты.

**33. G = (T, V, P, S) – *неукорачивающаяся грамматика*, если каждое правило из P имеет вид α → β, где**

при условии, что P не содержит S → ε и S не встречается в правах частях остальных правил.

**34. G = (T, V, P, S) - *контекстно-зависимая (КЗ)*, если каждое правило из P имеет вид α → β, где**

α → β, где α (TV)+, β (T V)+ и | α | | β |.

|  |  |  |  |
| --- | --- | --- | --- |
| Контекстно- зависимая (КЗ) α → β, | α | <= | β | | S0 → 0B0  B → 1 | 0BC  C0 → 100  C1 → 1C | {0n1n 0n | n1} | Машина Тьюринга  с конечной лентой |

**35. *Грамматику типа 1* можно определить как**

**а вот так как выше**

**36. Любая регулярная грамматика является**

любая регулярная грамматика является КС-грамматикой и укорачивающей КС-грамматикой

3**7. Любая КС-грамматика является грамматикой типа**

A → α, где AV, α ⊂(T˅V)\*.

**38. Любая КС-грамматика является  грамматикой типа**

**<2 ?**

**39. Любая КЗ-грамматика является грамматикой типа**

любая КЗ-грамматика является грамматикой типа 0.

α → β, где α (TV)+, β (T V)+ и | α | | β |.

**40. Любая неукорачивающая грамматика является грамматикой типа**

любая неукорачивающая грамматика является грамматикой типа 0.

**41. Каждый регулярный язык является**

каждый регулярный язык является КС-языком, но существуют КС-языки, которые не являются регулярными (например, L = {anbn | n>0});

**42. Каждый КС-язык является**

каждый КС-язык является КЗ-языком, но существуют КЗ-языки, которые не являются КС-языками (например, L = {anbncn | n>0});

**43. Каждый КЗ-язык является языком типа**

каждый КЗ-язык является языком типа 0. УКС-язык, содержащий пустую цепочку, не является КЗ-языком. Если язык задан грамматикой типа k, то это не значит, что не существует грамматики типа k’ (k’>k), описывающей тот же язык. Поэтому, когда говорят о языке типа k, обычно имеют в виду максимально возможный номер k.

**44. Определите  понятие семантики языков программирования**

это правила придания смысла синтаксически правильным программам. В конечном счете эти правила определяют ту последовательность действий вычислительной машины, которую она должна выполнить, работая по данной программе.

**45. Простейшие переводы описываются:**

Простейшие переводы можно задать при помощи гомоморфизма.