Выбрать правильный ответ. Прочитать вопрос и ответ.

1. Синтаксис языков описывается: **1.** Кластером **2.** Формальной грамматикой **3**. Формой Бекуса-Наура **4**. Диаграммой Вирта

Ответ: 2, 3, 4

Язык, будь то естественный (естественная речь) или язык программирования, задается множеством правил, образующих грамматику языка. Синтаксис языка отделяется от семантики. Появление формальных грамматик было обусловлено стремлением формализовать естественные языки и языки программирования.
Правила описывают те последовательности слов, которые являются корректными, т.е. допустимыми предложениями языка. Грамматика позволяет осуществить грамматический анализ предложения, а значит, и явно описать его структуру.

2. Грамматика языка реализуется: **1**. Цепочкой символов **2**. Порождающей системой **3**. Распознавателями

Ответ: 2, 3

Одно из назначений грамматики – это определить бесконечное число предложений языка с помощью конечного числа правил.
Грамматика языка может быть реализована двумя способами: в виде порождающей системы и в виде устройств, называемых распознавателями.
1. Порождающие системы: Для описания синтаксиса языков программирования наибольшее распространение получили следующие порождающие системы: формальные грамматики, форма Бекуса-Наура и ее модификации, диаграммы Вирта [3].
В порождающих системах правила грамматики задаются продукциями, состоящими из двух частей: левой и правой. В левой части, записывается определение, в правой части варианты, из которых может состоять определение.
Предложения языка, порождаются из исходного символа применением продукций. Шаги замены можно интерпретировать как шаги построения дерева вывода
БНФ - метаязык, который используется для описания других языков.
символ "$\Rightarrow$" означает, что один символ слева от $\Rightarrow$ в соответствии с правилом грамматики заменяется цепочкой, находящейся справа от $\Rightarrow$. Обозначение $\Rightarrow$ называют также непосредственное следование.
Две последовательности связаны отношением $\Rightarrow^*$, когда вторая получается из первой применением некоторой последовательности продукции $\alpha \Rightarrow^* \alpha_m$ тогда, и только тогда, когда $\alpha \Rightarrow^+ \alpha_m$.
Диаграммы Вирта позволяют визуально представить правила порождения. Каждому правилу соответствует диаграмма.
Элементы ИЛИ обозначаются разветвлениями и вершинами, элементы И последовательными вершинами графа, метасимволы {...} циклом. Вершины могут быть двух типов: терминальные – кружки, и нетерминальные – прямоугольники. Каждому правилу соответствует диаграмма.
Итерационная форма описания. Вводится операция итерации – обозначается парой круглых скобок со звездочкой. Итерация вида $(a)^*$ определяется как множество, включающее цепочки всевозможной длины, построенные с использованием символа **a**. $(a)^* = \{a, aa, aaa, aaaa, \ldots\}$
Алфавит (или словарный состав) – конечное множество символов. Например, *алфавит* $V_1 = \{a,b\}$ – содержит два символа, *алфавит* $V_2 = \{01,11,10,00\}$ – содержит четыре символа.
Цепочкой символов α *в алфавите* V называется любая конечная последовательность символов этого алфавита.
Пустой цепочкой символов ε называется цепочка, не содержащая ни одного символа.

V^* (замыкание V*)* – обозначается множество всех цепочек составленных из символов алфавита V, включая пустую цепочку ε. Например, если V={0,1}, то $V^* = \{\varepsilon, 0, 1, 00, 11, 01, 10, 000, 001, 011, ...\}$.
V^+ – обозначается множество всех цепочек составленных из алфавита V, исключая пустую цепочку ε. Следовательно, $V^* = V^+ \cup \{\varepsilon\}$.
Декартовым произведением A×B множеств A и B называется множество $\{(a,b) \mid a \in A, b \in B\}$.
∅ – обозначается пустое множество.

2. Распознающие системы (автоматы): Грамматика может быть реализована в виде некоторого алгоритма, производящего действия и отвечающего на вопрос, принадлежит ли данная фраза языку.

Функцией перехода δ называется отношение, управляющее действиями распознавателя.

Функция перехода δ определяет для каждой пары (входной символ, состояние) множество состояний, в которых он будет находится на следующем шаге.

Классификация Хомского: Синтаксис языка L можно специфицировать, пользуясь системой изображения множеств, например: $L = \{0^n 1^n \mid n \geq 0\}$

Язык L включает строки, состоящие из нуля или нулей и того же числа последующих единиц. Пустая строка включается в язык.Такое задание синтаксиса намного проще синтаксисов большинства языков специфицируемых с помощью грамматики.

3. $G = (T, V, P, S_0)$, где **1**. $T \cap V \neq \varnothing$, **2.** $T \cap V = \varnothing$, **3**.$T \cap P = \varnothing$.

Ответ: 2

Грамматика $G = (T, V, P, S_0)$, где T - конечное множество терминальных символов (терминалов) алфавита; V – конечное множество нетерминальных символов алфавита, не пересекающихся с T, $T \cap V = \varnothing$, S_0 - начальный символ (или аксиома), $S_0 \in V$;
P - конечное множество правил порождения (продукций), $P = (T \cup V)^+ \times (T \cup V)^*$.
Элемент (α, β) множества P называется *правилом вывода* и записывается в виде α → β.
Цепочка $\beta \subset (T \cup V)^*$ *непосредственно выводима* (α→β) из цепочки $\alpha \subset (T \cup V)^+$ в грамматике G , если $\alpha = \xi_1\gamma\xi_2$, $\beta = \xi_1\delta\xi_2$, где $\xi_1, \xi_2, \delta \subset (T \cup V)^*$, $\gamma \subset (T \cup V)^+$ и правило вывода γ → δ содержится в P.
Цепочка $\beta \subset (T \cup V)^*$ *выводима* из цепочки $\alpha \subset (T \cup V)^+$ в грамматике G, (обозначим $\alpha \Rightarrow \beta$), если существуют цепочки $\gamma_0, \gamma_1, ..., \gamma_n$ $(n \geq 0)$, такие, что $\alpha = \gamma_0 \to \gamma_1 \to ... \to \gamma_n = \beta$.
Последовательность $\gamma_0, \gamma_1, ..., \gamma_n$ называется *выводом длины n*.
Правила вывода с одинаковыми левыми частями $\alpha \to \beta_1, \alpha \to \beta_2, ..., \alpha \to \beta_n$, записывают сокращенно $\alpha \to \beta_1 \mid \beta_2 \mid ... \mid \beta_n$, где β_i , i= 1, 2,..., n и называют *альтернативой* правил вывода из цепочки α.
Для обозначения терминалов грамматики мы будем употреблять прописные буквы (или строки прописных букв), а для обозначения не терминалов — заглавные буквы (или строки заглавных букв).

4. Языком, порождаемым грамматикой $G = (T, V, P, S_0)$, называется
1. Множество $L(G)=\{\alpha \subset T^* \mid S_0 \Rightarrow^* \alpha\} \cap \{\varepsilon\}$.
2. Множество $L(G)=\{\alpha \subset T^* \mid S_0 \Rightarrow^* \alpha\}$.
3. Множество $L(G)=\{\alpha \subset V^* \mid T_0 \Rightarrow^* \alpha\}$.

Ответ: 2

Языком, порождаемым грамматикой G = (T, V, P, S0), называется множество $L(G)=\{\alpha \subset T^* \mid S0 \Rightarrow^* \alpha\}$. Другими словами, L(G) - это все цепочки в алфавите T, которые выводимы из S0 с помощью P.
Цепочка $\alpha \subset (T \cup V)^*$, для которой $S0 \Rightarrow^* \alpha$ (то есть цепока, которая может быть выведена из начального символа), называется сентенциальной формой в грамматике G. Язык, порождаемый грамматикой, можно определить как множество терминальных сентенциальных форм.

Грамматики G1 и G2 называются эквивалентными, если L(G1) = L(G2). Например, пусть задана грамматика G2 = ({0,1}, {S}, P2, S), где P2: $S \rightarrow 0S1 \mid 01$, тогда грамматики G1 и G2 эквивалентны, т.к. обе порождают язык $L(G1) = L(G2) = \{0n1n \mid n>0\}$.
Грамматики G2 и G3 называются почти эквивалентными L(G2) = L(G3), если $L(G2) = L(G3) \cup \{\varepsilon\}$. Другими словами, грамматики почти эквивалентны, если языки, ими порождаемые, отличаются не более чем на ε.

5. α = defgabck : **1.** | α | = 7; **2.** | α | = 8; **3.** | ε | = 0.

Ответ: 2, 3

Длина цепочки α – число составляющих ее символов, обозначается | α |. Длина ε равна 0, | ε | = 0.

6. Грамматика: S→0|0A A→1B B→0|0A **1.** Контекстно-свободная **2**. Регулярная

Ответ: 2 (1 ?)

Таб. 1. Классификация Хомского

Тип	Ограничения на правила P	Пример правил грамматики $G = (T, V, P, S_0)$	Пример языка	Автомат или абстрактная машина
0	Общего вида, не накладывается никаких ограничений	$S_0 \rightarrow CD$, $C \rightarrow 0CA$, $C \rightarrow 1CB$, $AD \rightarrow 0D$, $BD \rightarrow 1D$, $A0 \rightarrow 0A$, $A1 \rightarrow 1A$, $B0 \rightarrow 0B$, $B1 \rightarrow 1B$, $C \rightarrow \varepsilon$, $D \rightarrow \varepsilon$	$\{\omega\omega \mid \omega \in \{0,1\}^*\}$, язык состоит из цепочек четной длинны, из 0 и 1	Машина Тьюринга
1	Контекстно-зависимая (КЗ) $\alpha \rightarrow \beta$, $\mid \alpha \mid \leq \mid \beta \mid$	$S_0 \rightarrow 0B0$ $B \rightarrow 1 \mid 0BC$ $C0 \rightarrow 100$ $C1 \rightarrow 1C$	$\{0^n1^n0^n \mid n\geq 1\}$	Машина Тьюринга с конечной лентой
2	Контекстно-свободная (КС) $A \rightarrow \alpha$	$S_0 \rightarrow AS_0B \mid AB$ $A \rightarrow 0$ $B \rightarrow 1$	$\{0^n1^n \mid n\geq 1\}$	с магазинной памятью (МП-автомат)
3	Регулярная $A \rightarrow bB$ $A \rightarrow a$	$S_0 \rightarrow 0 \mid 0A$ $A \rightarrow 0B$ $B \rightarrow 1 \mid 1A$	$\{0(01)^n \mid n\geq 0\}$	Конечный автомат (КА)

(Классификация Хомского)

Грамматики классифицируются по виду правил вывода P. В правилах вывода P большими латинскими буквами обозначают нетерминальные символы, маленькими латинскими буквами - терминальные.
Автомат эквивалентен грамматике, если он воспринимает весь порожденный грамматикой язык и только этот язык.
Классификация Хомского дает представление об общности грамматик, и об эквивалентных им формальных автоматах.

Грамматика типа 3 или автоматная (регулярная) A-грамматика – это грамматика G = (T, V, P, S), у которой правила вывода P имеют вид по классификации Хомского $A \rightarrow bB$ (праволинейное правило) или $A \rightarrow a$ (заключительное правило), где $A, B \in V$, $a,b \in T$.
Каждое правило такой грамматики содержит единственный нетерминал в левой части, всегда один терминал в правой части, за которым может следовать один нетерминал.
Такую грамматику также называют праволинейной.
Грамматика G = (T, V, P, S) называется леволинейной, если каждое правило из P имеет вид по классификации Хомского $A \rightarrow Bb$ либо $A \rightarrow a$, где $A, B \in V$, $a,b \in T$.
Таким образом, грамматику типа 3 можно определить как праволинейную либо как леволинейную.
Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых праволинейными грамматиками, совпадает с множеством языков, порождаемых леволинейными грамматиками.
Автоматной грамматике эквивалентен простейший распознаватель – недетерминированный конечный автомат. Автомат представляет собой устройство, у которого отсутствует вспомогательная память.

7. Язык: $\{0(01)^n \mid n \geq 0\}$ **1**. Контекстно-свободный **2**. Регулярный

Ответ: 2

(Классификация Хомского)

8. Распознаватель: $\{0(01)^n \mid n \geq 0\}$ **1**. МП-автомат **2**. Конечный автомат

Ответ: 2

(Классификация Хомского)

9. $KA = (Q, \Sigma, \delta, q_0, F)$, где **1**. $\delta: Q \rightarrow Q$; **2**. $\delta: Q \times \Sigma \rightarrow P(Q)$; **3**. $\delta: Q \times \Sigma \rightarrow F$;

Ответ: 2

Конечный автомат (КА) – это пятерка объектов $KA = (Q, \Sigma, \delta, q_0, F)$, где
Q - конечное множество состояний;
Σ - конечный алфавит входных символов;
δ - функция переходов, задаваемая отображением $\delta: Q \times \Sigma \rightarrow P(Q)$,
где P(Q) конечное множество подмножеств множества Q;
q_0 - начальное состояние автомата, $q_0 \in Q$;
$F \subseteq Q$ - множество заключительных состояний.

В каждый момент времени КА находится в некотором состоянии q $\in$Q и читает поэлементно последовательность символов $a_k \in \sum$, записанную на конечной ленте. При этом либо читающая головка машины двигается в одном направлении (слева направо), либо лента перемещается (справа налево) при неподвижной читающей головке.

КА называется недетерминированным, если для каждой конфигурации существует конечное множество всевозможных следующих шагов, любой из которых КА может сделать, исходя из этой конфигурации. КА называется детерминированным, если для каждой конфигурации существует не более одного следующего шага.
Для любого конечного недетерминированного автомата можно построить ему эквивалентный детерминированный автомат.
Класс языков, распознаваемый конечными автоматными, совпадает с классом языков, порождаемых автоматными грамматиками и наоборот.

10. Лемма о накачке доказывает 1. эквивалентность языков 2.нерегулярность языка

Ответ: 2

Теорема, называемая “лемма о накачке” утверждает, что все достаточно длинные слова регулярного языка можно накачать, то есть повторить внутреннюю часть слова сколько угодно раз, производя новое слово, также принадлежащее языку. Лемма описывает существенное свойство всех регулярных языков и служит инструментом для доказательства нерегулярности некоторых языков. Она является одной из нескольких лемм о накачке.
Свойство замкнутости регулярных языков, позволяет минимизировать построение автомата:

1. строить рапознаватели для одних языков, построенных из других с помощью операций;
2. определить, что два различных автомата определяют один язык.

Формальное утверждение:
Пусть L регулярный язык. Тогда существует целое $p \geq 1$ зависящее только от L, такое что цепочка w из L длины по меньшей мере p (p называется «длиной накачки») может быть записана как w = xyz, где y – это подцепочка, которую можно накачать (удалить или повторить произвольное число раз, так что результат останется в L), при этом:

1. $| y | \geq 1$, цикл y должен быть накачан хотя бы длиной 1;
2. $|xy | \leq p$, цикл должен быть в пределах первых p символов;
3. для всех $i \geq 0$, $xyiz \in L$, на x и z ограничений не накладывается.

11. КА задается **1.** пятеркой объектов **2.** таблицей переходов

Ответ: 1, 2

Конечный автомат можно задать в виде таблицы переходов и диаграммы (графа) переходов.
В таблице переходов двум аргументам ставится в соответствие одно значение.
Диаграммой переходов КА называется неупорядоченный граф, удовлетворяющий условиям:
- каждому состоянию q соответствует некоторая вершина, отмеченная его именем;
- диаграмма переходов содержит дугу из состояния q_k в состояние q_n, отмеченную символом a, если $q_k \in \delta(q_n, a)$. Дуга может быть помечена множеством символов, переводящих автомат из состояния q_k в состояние q_n;

- вершины, соответствующие заключительным состояниям $q_f \in F$, отмечаются двойным кружком.

12. Конфигурация КА **1**. $(q, \omega, F) \in Q \times \Sigma^*$, где $q \in F$, $\omega \in \Sigma^*$.
2. $(q, \omega) \in Q \times \Sigma^*$, где $q \in Q$, $\omega \in \Sigma^*$ **3.** $(q, \omega, \Sigma) \in Q \times \Sigma^*$, где $q \in Q$, $\omega \in \Sigma^*$.

Ответ: 2

Конфигурация КА – это пара множества $(q, \omega) \in Q \times \Sigma^*$, где $q \in Q$, $\omega \in \Sigma^*$. Конфигурация (q_0, ω) называется *начальной*, а (q, ε), где $q \in F$, - заключительной.
Определим бинарное отношение $\vdash$ на конфигурациях, соответствующее одному такту работы КА. Если $q' \in \delta(q,a)$, то $(q, a\omega) \vdash (q', \omega)$ для всех $\omega \in \Sigma^*$.

13. L(КА) распознаваем КА : **1.** $L(КА) = \{\omega \in \Sigma^* | (q_0,\omega,F) \rightarrow (q, \varepsilon)$, где $q \in F\}$
2. $L(КА) = \{\omega \in \Sigma^* | (q_0,\omega) \vdash (q, \varepsilon)$, где $q \in F\}$

Ответ: 2

Пусть {C} – множество конфигураций.

1. $C \vdash^0 C'$ означает, что $C = C'$
2. $C_0 \vdash^k C_k$, если существует последовательность конфигураций $C_1, C_2, \ldots, C_{k-1}$, для $k \geq 1$, в которых $C_i \vdash C_{i+1}$ для $0 \leq i < k$.
3. $C \vdash^+ C'$ означает, что $C \vdash^k C_k$ для некоторого $k \geq 1$, а $C \vdash C'$ означает, что $C \vdash^k C'$ для $k \geq 1$.

Конечеый автомат КА = $(Q, \Sigma, \delta, q_0, F)$ распознает входную цепочку $\omega \in \Sigma^*$, если $(q_0, \omega) \vdash^k (q, \varepsilon)$ для $q \in F$.
Языком L(КА), распознаваемым КА называется множество входных цепочек,
$L(КА) = \{\omega \in \Sigma^* | (q_0,\omega) \vdash^k (q, \varepsilon)$, где $q \in F\}$

14. Правила КС-грамматики имеют вид 1.$A \rightarrow SD$ **2.**$A \rightarrow SdD$ **3**.$A \rightarrow \alpha$, $\alpha \subset (T \cup V)^*$

Ответ: 3 (1, 2 ?)

Из четырех типов грамматик иерархии Хомского класс контекстно-свободных грамматик наиболее важен с точки зрения приложений к языкам программирования и компиляции. С помощью этого типа грамматик определяется большая часть синтаксических структур языков программирования.
Определение 6. Контекстно-свободная грамматика типа 2:
Грамматика $G = (T, V, P, S_0)$ называется контекстно-свободной (КС) (или бесконтекстной), если каждое правило из P имеет вид $A \rightarrow \alpha$, где $A \subset V$, $\alpha \subset (T \cup V)^*$.
Заметим, что, согласно определению каждая праволинейная грамматика – КС. Языки, порождаемые КС-грамматиками, называются КС-языками.
Пример, КС-грамматики, порождающей скобочные арифметические выражения: $G_1 = (\{v,+,*,(,)\}, \{S,F,L\}, P, S)$, где $P = \{S \rightarrow S + F, S \rightarrow F, F \rightarrow F * L, F \rightarrow L, L \rightarrow v, L \rightarrow (S)\}$.
Пример вывода, заменяя самый левый нетерминал (левосторонний вывод) сентенциальной формы : $S \Rightarrow S + F \Rightarrow F + F \Rightarrow L + F \Rightarrow v + F \Rightarrow v + F * L \Rightarrow v + L * L \Rightarrow v + v * L \Rightarrow v + v * v$.

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

Ответ: 1, 2

Для построения синтаксических анализаторов необходимо, чтобы КС-грамматика была в приведенной форме.
Если применить к КС-грамматике алгоритмы устранения бесполезных символов (1-2): непроизводящих и/или недостижимый символов, преобразования в грамматику без ε–правил (3) и устранения цепных правил (4), то получим приведенную КС-грамматику. Рассмотрим эти алгоритмы и их применение.
1. Символ $a \in T \cup V$ называется недостижимым в КС-грамматике G, если он не может появиться ни в одной из сентенциальных форм.
Нетерминальный символ $A \in V$ называется производящим, если из него можно вывести терминальную цепочку, т.е. если существует вывод $A \Rightarrow^{+} x$, где $x \in T^{+}$. В противном случае символ называется непроизводящим.
2. Символ $a \in T \cup V$ называется бесполезным в КС-грамматике G, если он не непроизводящий или недостижимый.

16. G = (T, V, P, S), в нормальной форме Грейбаха, если
1. нет ε-правил **2.** $A \rightarrow a\alpha$, где $a \in T$, $\alpha \subset V^{*}$ **3.** $A \rightarrow \alpha a$, где $a \in T$, $\alpha \subset V^{*}$

Ответ: 1, 2

Частный случай не леворекурсивной грамматики – грамматика в нормальной форме Шейлы Грейбах.
Определение 7. КС грамматика G = (T, V, P, S) называется грамматикой в нормальной форме Грейбах, если в ней нет ε-правил, т.е. правил вида $A \rightarrow \varepsilon$, и каждое правло из P отличное от $S \rightarrow \varepsilon$, имеет вид $A \rightarrow a\alpha$, где $a \in T$, $\alpha \subset V^{*}$.

17. Нормальная форма Хомского позволяет упростить рассмотрение свойств:
1. МП-автомата **2.** КС-грамматики. **3.** Регулярного языка

Ответ: 2 (1, 3 ?)

Также полезно представлять грамматику в нормальной форме Хомского, что позволяет упростить рассмотрение ее свойств.

18. G = (T, V, P, S) в нормальной форме Хомского, если каждое правило из P имеет один из следующих видов:
1. $A \rightarrow BC$, где $A, B, C \in V$;
2. $A \rightarrow a$, где $a \in T$;
3. $S \rightarrow \varepsilon$, если $\varepsilon \in L(G)$, причем S не встречается в правых частях

Овтвет: 1, 2, 3

КС грамматика G = (T, V, P, S) называется грамматикой в нормальной форме Хомского, если каждое правло из P имеет один из следующих видов:
1. $A \rightarrow BC$, где $A, B, C \in V$;
2. $A \rightarrow a$, где $a \in T$;
3. $S \rightarrow \varepsilon$, если $\varepsilon \in L(G)$, причем S не встречается в правых частях правил.

19. Нетерминал КС-грамматики *рекурсивен* 1. $A \Rightarrow^{+} \alpha\beta$ 2. $A \Rightarrow^{+} \alpha A\beta$

Ответ: 2

Нетерминал КС-грамматики называется рекурсивным, если $A \Rightarrow^{+} \alpha A \beta$, для некоторых α и β. Если $\alpha = \varepsilon$, то A называется леворекурсивным, если $\beta = \varepsilon$, то A называется праворекурсивным. Грамматика, имеющая хотя бы один леворекурсивный нетерминал, называется леворекурсивной. Грамматика, имеющая хотя бы один праворекурсивный, нетерминал называется праворекурсивной.
При нисходящем синтаксическом анализе требуется, чтобы приведенная грамматика рассматриваемого языка не содержала левой рекурсии.
Для любой КС-грамматики существует эквивалентная грамматика без левой рекурсии.

20. МП = $(Q, \sum, \Gamma, \delta, q_0, z_0, F)$ **1.** $\delta: Q \times \sum$ **2.** $\delta: Q \times \sum \times F$ **3.** $\delta: Q \times (\sum \cup \{\varepsilon\}) \times \Gamma$

Ответ: 3

Автоматы с магазинной памятью (МП - автоматы) представляют собой модель распознавателей для языков, задаваемых КС-грамматиками. МП - автоматы имеют вспомогательную память, называемую магазином см. 1.10. В магазин можно поместить неограниченное количество символов. В каждый момент времени доступен только верхний символ магазина.

МП автомат – это семерка объектов МП = $(Q, \sum, \Gamma, \delta, q_0, z_0, F)$:
Q – конечное множество состояний устройства управления;
$\sum$ - конечный алфавит входных символов;
Γ - конечный алфавит магазинных символов;
δ - функция переходов, отображает множества $Q \times (\sum \cup \{\varepsilon\}) \times \Gamma$ в множество конечных подмножеств множества $Q \times \Gamma^{*}$;
q_0 - начальное состояние, $q_0 \in Q$;
z_0 - начальный символ магазина, $z_0 \in \Gamma$;
F - множество заключительных состояний, $F \subseteq Q$.

21. Конфигурация МП-автомата: **1.**$(q,\omega,\alpha) \in Q \times \sum^{*}$ **2.**$(q,\omega,\alpha) \in Q \times \sum^{*} \times \Gamma^{*}$ **3**.$(q,\omega,\alpha) \in Q \times \sum^{*}$

Ответ: 2

Конфигурацией МП-автомата называется тройка $(q, \omega, z) \in Q \times \sum^{*} \times \Gamma^{*}$, где
q – текущее состояние управляющего устройства;
ω – необработанная часть входной цепочки (первый символ цепочки ω находится под входной головкой; если $\omega = \varepsilon$, то считается, что вся входная цепочка прочитана) ;
z – содержимое магазина (самый левый символ цепочки z считается верхним символом магазина; если $z = \varepsilon$, то магазин считается пустым).
Такт МП-автомата будем описывать бинарным отношением ($\vdash$), определенным на множестве конфигураций. Так же, как и для конечных автоматов, можно определить транзитивное ($\vdash^{+}$) и рефлексивно-транзитивное ($\vdash^{*}$) замыкания отношения ($\vdash$).
Начальной конфигурацией МП-автомата называется конфигурация вида (q_0, ω, z_0), где устройство управления находится в начальном состоянии, на входной ленте записана цепочка $\omega \in \sum^{*}$, которую необходимо распознать, а магазин содержит только начальный символ z_0.
Заключительной конфигурацией МП-автомата называется конфигурация вида (q, ε, α), где $q \in F$ – одно из заключительных состояний устройства управления, входная цепочка

прочитана до конца, а в магазине записана некоторая, заранее определенная цепочка $\alpha \in \Gamma^*$.
Есть два способа определить язык, допускаемый МП-автоматом. Во-первых, множеством входных цепочек, для которых существует опустошающая магазин последовательность операций и множеством входных цепочек, для которых существует последовательность операций, приводящая автомат в заключительное состояние.
Языком, *определяемым* (допускаемым, распознаваемым) МП-автоматом, называется множество цепочек, допускаемых этим автоматом.

22. Детерминированным МП-автомат

1. $\delta(q,\alpha,Z)$ содержит не более одного элемента для каждого $\alpha \in \sum$ и $\delta(q, \varepsilon, Z) = \varnothing$;
2. $\delta(q, \alpha, Z) = \varnothing$ для всех $\alpha \in \sum$, и $\delta(q,\varepsilon,Z)$ содержит не более одного элемента.

Ответ: 1, 2

На практике часто используются детерминированные МП-автоматы.
Детерминированным МП-автоматом (ДМП-автоматом) с магазинной памятью называется МП-автомат ДМП = $(Q, \sum, \Gamma, \delta, q_0, Z_0, F)$, у которого для каждого $q \in Q$ и $Z \in \Gamma$ выполняется одно из условий:
1. $\delta(q, \alpha, Z)$ содержит не более одного элемента для каждого $\alpha \in \sum$ и $\delta(q, \varepsilon, Z) = \varnothing$;
2. $\delta(q, \alpha, Z) = \varnothing$ для всех $\alpha \in \sum$, и $\delta(q, \varepsilon, Z)$ содержит не более одного элимента.
Если каждое правило КС-грамматики начинается с терминального символа, причем альтернативы начинаются с различных символов, то достаточно просто построить *детерминированный* синтаксический анализатор.
МП-автомат и расширенный РМП-автоматы – детерминированные автоматы.

23. **1.** $FIRST(A) = \{a \in V \mid A \Rightarrow^{+}_{i} \alpha\beta, \text{ где } \beta \in (T \cup V)^*\}$
2. $FIRST(A) = \{a \in T \mid A \Rightarrow^{+}_{i} \alpha\beta, \text{ где } \beta \in (T \cup V)^*\}$

Ответ: 2

КС-грамматика G = (T, V, P, S) без ε-правил называется простой LL(1) грамматикой (s-грамматикой, разделенной грамматикой), если для каждого $v \in V$ все его альтернативы начинаются различными терминальными символами. Единица в названии алгоритма означает, что при чтении анализируемой цепочки, находящейся на входной ленте, входная головка может заглядывать вперед на один символ.

Множество FIRST (первый) – это множество терминальных символов, которыми начинаются цепочки, выводимые из нетерминала A:
$FIRST(A) = \{a \in T \mid A \Rightarrow^{+}_{i} a\beta, \text{ где } \beta \in (T \cup V)^*\}$

24. Конфигурация “1-предсказывающего” алгоритма разбора имеет вид (x, $X\alpha\perp$, π), где x – неиспользуемая часть входной цепочки, $X\alpha$ – цепочка в магазине 1. π – используемая часть входной цепочки.
2. π – цепочка на входной ленте.
3. π – цепочка на выходной ленте.

Ответ: 3

Магазин содержит цепочку $X\alpha\perp$ (см. рис. 1.17), где $X\alpha$ – цепочка магазинных символов (X - верхний символ магазина), а символ ($\perp$) – специальный символ, называемый *маркером*

дна магазина. Если верхним символом магазина является *маркер дна*, то магазин пуст. Выходная лента содержит цепочку номеров правил π, представляющую собой текущее состояние левого разбора.
Конфигурацию "1-предсказывающего" алгоритма разбора будем представлять в виде (x, Xα⊥, π), где x – неиспользуемая часть входной цепочки, Xα – цепочка в магазине, а π – цепочка на выходной ленте.

25. Управляющая таблица применяется для распознавания
1. КС-языка **2.** LR(k) языка

Ответ: 2

26. Для LR(k) языка можно построить анализатор
1. горизонтальный **2.** нисходящий **3.** восходящий

Ответ: 3

Грамматики, для которых можно построить детерминированный восходящий анализатор, получили название LR(k)-грамматик (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).
Наиболее наглядно LR(k)-грамматику можно определить в терминах деревьев вывода. Грамматика G = (T, V, P, S) является LR(k)-грамматикой, если просмотрев только часть кроны дерева вывода в этой грамматике, расположенной слева от данной внутренней вершины, часть кроны, выведенную из нее, и следующие k символов входной цепочки, можно установить правило вывода, которое было применено к этой вершине при порождении входной цепочки.
Для любой LR(k)-грамматики G = (T, V, P, S) можно построить детерминированный анализатор, который выдает правый разбор входной цепочки.
Анализатор состоит из магазина, входной ленты, выходной ленты и управляющего устройства (пара функций f и g) см. рис. 1.18.

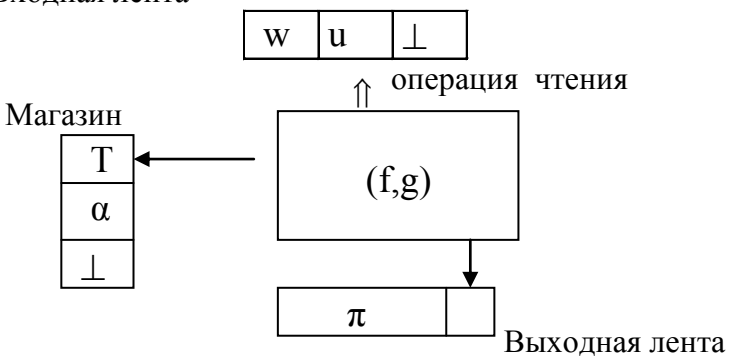

Рис. 1.18. LR(k)-анализатор

27. В LR(k) разборе применяются операции **1.** Сдвига **2.** Свертки **3.** Переноса

Ответ: 2, 3

LR(k)-алгоритм разбора
1. j := 0.
2. j := j+1. Если j > n, то выдать сообщение об ошибке и перейти к шагу 5.

3. Определить цепочку u следующим образом:
   - если k = 0, то u= $\mathbf{t_j}$;
   - если k ≥ 1 и j + k – 1 ≤ n, то u= $t_j\mathbf{t_{j+1}\ldots t_{j+k-1}}$ – первые k-символов цепочки $t_jt_{j+1}\ldots t_n$;
   - если k ≥ 1 и j + k – 1 > n, то u= $t_j\mathbf{t_{j+1}\ldots t_n}$ – остаток входной цепочки.
4. Применить функцию действия f из строки таблицы M, отмеченной верхним символом магазина T, к цепочке u:

4. f(u) = ПЕРЕНОС (П). Определить функцию перехода $g(t_j)$ из строки таблицы M, отмеченной символом T из верхушки магазина. Если $g(t_j)$ = T' и $T'V_p\{\perp\}$, то записать T' в магазин и перейти к шагу 2. Если $g(t_j)$= ОШИБКА, то выдать сигнал об ошибке и перейти к шагу 5;
5. f(u) = (СВЕРТКА, i) (С) и A→α – правило вывода с номером i грамматики G. Удалить из верхней части магазина |α| символов, в результате чего в верхушке магазина окажется символ $T'V_p\{\perp\}$, и выдать номер правила i на входную ленту.Определить символ T'’= g(A) из строки таблицы M, отмеченной символом T', записать его в магазин и перейти к шагу 3.
6. f(u) = ОШИБКА (О). Выдать сообщение об ошибке и перейти к шагу 5.
7. f(u) = ДОПУСК (Д). Объявить цепочку, записанную на входной ленте, правым разбором входной цепочки z.

5. Останов.

28. Отношение OBLOW задается 1. вектором 2. матрицей 3. Выражением

Ответ: 2

29. Для грамматики предшествования строится **1.** алгоритм “перенос-свертка”
2. Функции переноса и свертки **3.** Функции эквивалентности

Ответ: 1, 2

Существует подкласс LR(k)-грамматик, который называется грамматиками предшествования для них легко строится алгоритм типа “перенос-свертка”.

30. LR(k)-алгоритм разбора описывается
1. конфигураций (αT, ax, π) 2. правилами сдвига 3. системой уравнений.

Ответ: 1

Работу алгоритма типа “перенос-свертка” удобно описывать в терминах конфигураций вида ($\perp X_1 \ldots X_m, a_1 \ldots a_n, p_1\ldots p_r$), где :
$\perp X_1 \ldots X_m$ – содержимое магазина, X_m – верхний символ магазина и $X_i \in V \cup$ T, символ ($\perp$) для магазина;
$a_1 \ldots a_n$ – оставшаяся непрочитанная часть входной цепочки, a_1 – текущий входной символ;
$p_1\ldots p_r$ – разбор, полученный к данному моменту времени.

31. G = (T, V, P, S) - грамматика типа 0 если
1. Нет рекурсивных правил **2.** на правила нет ограничений **3.**правила ограничены

Ответ: 2

Грамматика G = (T, V, P, S) называется *грамматикой типа 0*, если на правила вывода не накладывается никаких ограничений (кроме тех, которые указаны в определении грамматики).
Грамматики типа 0 - это самые общие грамматики.

32. Конфигурация машины Тьюринга **1.** $\Sigma\times Q\times\Sigma$ **2.** $\Sigma^*\times Q\times\Sigma^+$ **3.** $\Sigma\times Q\times\Sigma^+$

Ответ: 2

Машина Тьюринга (служит для проверки разрешимости алгоритма), в иерархии Хомского занимает самый верхний уровень:
M=(Q, Σ, R, L, B, δ, q_0,)
Q - множество состояний автомата
Σ - входной словарь
R, L - правые и левые символы, не входящие в $Q\cup\Sigma$
B - пробел ($B \in \Sigma$)
δ - множество правил переписывания
q_0 - исходное состояние
$\delta = q_ia_i\rightarrow q_ka_l$, a_i - входные символы $q_ia_i\rightarrow q_kR$,
$\delta = Q\times\Sigma$ в $Q\times(\Sigma\cup\{0,L\})$ $q_ia_j\rightarrow q_kL$

Машина Тьюринга - детерминированный преобразователь. Конфигурация машины - множество элементов следующего вида: $\Sigma^*\times Q\times\Sigma^+$, то есть последовательности $w_1q_1a_jw_2$, где w_1 и $w_2 \in\Sigma^*$, $a_j\in\Sigma$, $q_i\in Q$.
Правила из δ задают последовательные переходы от одной конфигурации (до тех пор, пока не будет достигнута заключительная конфигурация).
Конфигурация $w_1q_ia_jw_2$ называется заключительной, если к ней неприменимо ни одно из правил δ (т.е. правил с заголовком q_1a_j).

33. G = (T, V, P, S) – *неукорачивающаяся грамматика*, если каждое правило из P имеет вид $\alpha \rightarrow \beta$, где **1**. $\alpha \subset (T\cup V)^+$, $\beta \subset (T\cup S)^+$ и $|\alpha| \le |\beta|$.
2. $\alpha \subset (T\cup V)^+$, $\beta \subset (T\cup V)^+$ и $|\alpha| \le |\beta|$.

Ответ: 2

Грамматика G = (T, V, P, S) называется *неукорачивающей грамматикой*, если каждое правило из P имеет вид $\alpha \rightarrow \beta$, где $\alpha \subset (T\cup V)^+$, $\beta \subset (T\cup V)^+$ и $|\alpha| \le |\beta|$.

34. G = (T, V, P, S) - *контекстно-зависимая (КЗ)*, если каждое правило из P имеет вид $\alpha \rightarrow \beta$, где **1.** $\alpha = \xi_1A\xi_2$; $\beta = \xi_1\gamma\xi_2$; $A\subset V$; $\gamma \subset T$; $\xi_1,\xi_2 \subset (T\cup V)^*$.
2. $\alpha = \xi_1A\xi_2$; $\beta = \xi_1\gamma\xi_2$; $A\subset V$; $\gamma \subset (T\cup V)^+$; $\xi_1,\xi_2 \subset (T\cup V)^*$.

Ответ: 2

Грамматика G = (T, V, P, S) называется *контекстно-зависимой (КЗ)*, если каждое правило из P имеет вид $\alpha \rightarrow \beta$, где $\alpha = \xi_1A\xi_2$; $\beta = \xi_1\gamma\xi_2$; $A\subset V$; $\gamma \subset (T\cup V)^+$; $\xi_1,\xi_2 \subset (T\cup V)^*$.

35. *Грамматику типа 1* можно определить как **1.** неукорачивающую
2. контекстно-зависимую.

Ответ: 1, 2

Грамматику типа 1 можно определить как неукорачивающую либо как контекстно-зависимую.
Выбор определения не влияет на множество языков, порождаемых грамматиками этого класса, поскольку доказано, что множество языков, порождаемых неукорачивающими грамматиками, совпадает с множеством языков, порождаемых КЗ-грамматиками.

36. Любая регулярная грамматика является
1. LL(k)-грамматикой **2.** КС-грамматикой **3.** LR(k)-грамматикой

Ответ: 2

Любая регулярная грамматика является КС-грамматикой;

37. Любая КС-грамматика является **1.** LL(k)-грамматикой **2.** КЗ-грамматикой

Ответ: 2

Любая КС-грамматика является КЗ-грамматикой;

38. Любая КС-грамматика является **1.** LR(k)-грамматикой **2.** неукорачивающей

Ответ: 2

Любая КС-грамматика является неукорачивающей грамматикой;

39. Любая КЗ-грамматика является грамматикой типа **1.** ноль **2.** типа 1

Ответ: 1

Любая КЗ-грамматика является грамматикой типа 0.

40. Любая неукорачивающая грамматика является грамматикой типа
1. Ноль **2.** типа 4

Ответ: 1

Любая неукорачивающая грамматика является грамматикой типа 0.

41. Каждый регулярный язык является **1.** КС-языком **2.** Усеченным языком

Ответ: 1

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

42. Каждый КС-язык является **1.** регулярным языком **2.** КЗ-языком **3.** Автоматным

Ответ: 2

Каждый КС-язык является КЗ-языком, но существуют КЗ-языки, которые не являются КС-языками (например, $L = \{a^n b^n c^n \mid n>0\}$);

43. Каждый КЗ-язык является языком типа **1.**автоматным **2.** регулярного **3.** ноль

Ответ: 3

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.
Например, КЗ-грамматика $G_1 = (\{0,1\}, \{A,S\}, P_1, S)$ и КС-грамматика $G_2 = (\{0,1\}, \{S\}, P_2, S)$, где $P_1 = \{S \rightarrow 0A1,\ 0A \rightarrow 00A1, A \rightarrow \varepsilon\}$ и $P_2 = \{S \rightarrow 0S1 \mid 01\}$
описывают один и тот же язык $L = L(G_1) = L(G_2) = \{ 0^n1^n \mid n>0\}$. Язык L называют КС-языком, т.к. существует КС-грамматика, его описывающая. Но он не является регулярным языком, т.к. не существует регулярной грамматики, описывающей этот язык.

44. Для линейного клеточного автомата КА= <Grid, N, Q, δ, $Grid_0$,> функция перехода состояний клетки c_i^k **1.** δ: $c_i^k \rightarrow c_i^{k+1}$ **2.** δ: $c_i^k \xrightarrow{neib} c_i^{k+1}$

Ответ: 2

Клеточным автоматом называется дискретная динамическая система, определяемая некоторой периодической многомерной решеткой, из неограниченного итерированного, или мозаичного, массива конечных автоматов (клеток), каждый из которых взаимодействует со своими соседями. Состояние i-й клетки в момент времени k характеризуется некоторой переменной, которая может быть целым, действительным или комплексным числом, либо представлять собой набор из нескольких чисел или объектом, в зависимости от решаемой задачи.
Для КА задается правило перехода состояния клетки из текущего в следующее. Правило перехода определяюеся локально: состоянием клетки и/или состояниями соседних клеток. Правила не меняются со временем.
Состоянием решётки называется совокупность состояний всех клеток решётки. Каждое изменение состояния решётки называется итерацией, а каждая итерация соответствует некоторому дискретному интервалу времени. Состояния клеток изменяются синхронным образом через дискретные интервалы времени в соответствии с правилами.
Глобальное поведение КА полностью определяется в терминах локальных зависимостей между клетками, что позволяет моделировать создание сложного глобального поведения из локальных взаимодействий в пространстве достижимых состояний. Проектирование правил перехода, задающих требуемое глобальное поведение, является сложной задачей.
КА характеризуется параллельной архитектурой получившей название "не-фон-неймановской", так как созданная им ранее последовательная архитектура уже была названа "фон-неймановской". Клеточные автоматы моделируют параллельные вычисления подобно тому, как машины Тьюринга моделируют последовательные вычисления.
Сложность КА возрастает с количеством клеток и/или взаимодействий в правилах переходов, что требует увеличения количества памяти и вычислений.
Самый простой линейный КА имеет одномерную решетку, заполненную клетками, которые могут принимать значение ноль или единица. Такой КА также называют булевым КА.

КА = <Grid, N, Q, δ, $Grid_0$,> – одномерный или линейный КА, где $Grid = [c_1, c_2,\ldots,c_N]$ – решетка, состояшая из N позиций. Каждая позиция i содержит клетку c_i, $i = 1,2,\ldots,N$.
Q – конечное множество состояний клетки c_i, при $Q = \{0,1\}$ клетка c_i может находиться в состоянии нуль или единица.

$neib_i$ – конечное множество локальных клеток, влияющих на значение данной, за исключением её самой c_i, называется окрестностью (или соседями) клетки, которую задают вводя метрику, поэтому о решётке говорят, как о дискретном метрическом пространстве. Соседями $neib_i$ для клетки c_i является множество локальных позиций решетки Grid с дистанцией от c_i цепочки не больше заданного радиуса r.
Для одномерной решетки каждая клетка c_i имеет ровно два соседа, причем c_N сосед c_1, то есть первая позиция решетки связывается с последней, образуя тор.

δ: $c_i^k \xrightarrow{neib_i} c_i^{k+1}$ – локальная функция перехода состояния клетки c_i^k на шаге работы k, k = 0,1,… в новое состояние c_i^{k+1} на шаге k+1 в соответствии с состояниями соседей $neib_i$.
При Q ={0,1} количество возможных состояний $neib_i = 2^{2r+1}$.
Состояния клеток КА могут меняться все одновременно (синхронный КА) или неодновременно (асинхронный КА). Состояния клетки c_i на такте m выполнения КА записывается как c_i^m, тогда $Grid_k = [c_1^m, c_2^m, \ldots, c_N^m]$.

В синхронном КА правило перехода δ имеет вид $\delta:[c_1^k, c_2^k, \ldots, c_N^k] \xrightarrow{neib} [c_1^{k+1}, c_2^{k+1}, \ldots, c_N^{k+1}]$, то есть $\delta(Grid_k)$.
$Grid_0$ – начальная конфигурация автомата, $Grid_0 = [c_1^0, c_2^0, \ldots, c_N^0]$.
Определим бинарное отношение $\vdash$ на конфигурациях, соответствующее одному такту k работы КА $\delta(Grid_{k-1}) \vdash^m \delta(Grid_k)$.
Пусть {Grid} – множество конфигураций КА.

1. $Grid \vdash^0 Grid'$ означает, что $Grid = Grid'$
2. $Grid_0 \vdash^k Grid_k$, если существует последовательность конфигураций $Grid_1$, $Grid_2$, …, $Grid_{k-1}$, для $k \geq 1$, в которых $Grid_m \vdash Grid_{m+1}$ для $0 \leq m < k$, то :

$Grid_0 \vdash \ldots \vdash Grid_m = [c_1^m, c_2^m, \ldots, c_N^m] \vdash^{m+1} Grid_{m+1} = [\delta(neib_1^m), \ldots, \delta(neib_N^m)] \ldots \vdash^k Grid_k$.

Пример. Пусть задан КА = <Grid, N = 29, Q ={0,1}, $c_{n+1i}=\delta(c^n_{i-1} \mid c^n_i \mid c^n_{i+1})$, $Grid_0=[c_1=\{0\},..., c_{14}=\{1\}, c_{15} = \{0\},\ldots, c_{29}=\{0\}]$ >

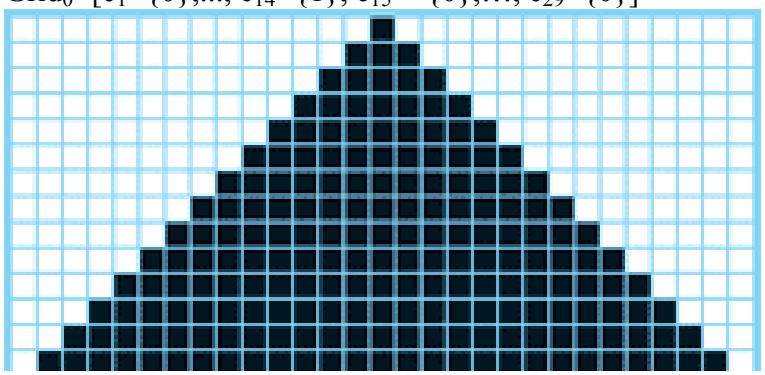

Рис. 5.1. Конфигурации булевого КА

Множество конфигураций булевого КА порождает побитовое (из нулей и единиц) изображение фрактального множества (треугольника Серпинского), которое иллюстрирует фрагмент на рис. 5.1.

Применение клеточных автоматов
В конце 1980-х посредством клеточных автоматов исследовались модели формальных принципов организации биологической жизни и эволюции жизнеподобных структур – “организмов” (искусственных кибернетических сущностей). Результаты исследований сформировали направление “искусственная жизнь”, Д.Холланд, С.Кауфман и К.Лангтон [37,38]. К.Лангтон сформулировал основной принцип: “логическая форма” организма может быть отделена от материальной основы его конструкции”.

Клеточные автоматы позволяют аппроксимировать и моделировать много естественных систем.
При решении дифференциальных уравнений в частных производных пользуются дискретной решеткой, заполненной клетками, а правила переходов соответствуют разностным аналогам дифференциальных уравнений. Динамика исследуемой системы описывается локальными правилами КА, в отличие от обыкновенных дифференциальных уравнений (ОДУ). Для КА достаточно знать законы развития системы (правила переходов) на микро- или мезоуровне в небольших пространственных областях (клетках), из которых состоит макросистема. Важно, что эти локальные правила одинаковы для всех клеток.
В КА в отличие от дифференциальных уравнений (ДУ) могут использоваться не только дискретные, но и целочисленные переменные. Дискретность переменных позволяет рассматривать большой класс разрывных недифференцируемых функций. Дискретные свойства КА заметно уменьшаются при работе с большими значениями переменных. Всегда существует минимальный дискретный шаг изменения переменной. При численном решении ОДУ или ДУ в частных производных можно уменьшать шаг дискретности до сколь угодно малых величин.
Вероятностные КА применяют для решения задач типа "реакция – диффузия – конвекция" с применением процедуры Монте-Карло.

45. Для одномерного клеточного автомата при r=2 **1.**$|neib_i|=2$ **2.**$|neib_i|=\mathbf{2^3}$ 3.$|neib_i|=5$

Ответ: 3

Для одномерного КА количество соседей всегда равно 2r+1=| neibi |. Например, при r = 3 и N = 129, соседями для c2 neib2 = {c128, c129, c1, c2, c3, c4, c5}, | neibi |= 7.

46. L-система строит **1.** геометрические объекты **2.** фрактальные множества

Ответ: 1, 2

Свойство – “подобия” используется в качестве определяющего во многих работах по фракталам. Слово "фрактал" [1] было введено Бенуа Мандельбротом в конце 1970'х. Часть фрактальной структуры подобна целому. Не имеет значения как сделано разбиение малой части, часть содержит не меньше деталей, чем целое. Отсюда применение термина фрактал, как производного от латинского прилагательного fractus – дробный и глагола frangere – ломать.
Понятие L-системы, тесно связано с самоподобными фракталами, и появилось только в 1968 году благодаря А. Линденмайеру [11]. Изначально L-системы были введены при изучении формальных языков, а также использовались в биологических моделях, селекции. С их помощью можно строить многие известные самоподобные фракталы, включая снежинку Коха и ковер Серпинского, кривые Пеано (работы Пеано, Гильберта, Серпинского) и т.д.
L-системы подобны грамматикам с одной лишь принципиальной разницей; если в грамматиках на каждом шаге вывода заменяется единственное вхождение нетерминала, то в L-системах все нетерминалы замещаются параллельно.
В качестве подсистемы вывода, для графической реализации L-систем, используется *тертл*-графика (turtle – черепаха). Черепашка, двигаясь по экрану дискретными шагами как правило прочерчивает свой след, но при необходимости может перемещаться без рисования. Для ее движения задаются три параметра (x,y,a): (x,y) – координаты черепашки, a – направление движения.

Алгоритм работы L-системы заключается в следующем. По правилам порождения строится цепочка символов или кодовое слово, символы представляют собой команды для черепашки и определяют траекторию ее движения. Черепашка выполняет последовательность команд, задаваемых словом. Символы слова читаются слева направо и коддируют следующие команды:
F – переместиться вперед на один шаг, прорисовывая след.
b – переместиться вперед на один шаг, не прорисовывая след.
[– открыть ветвь (подробнее см. ниже)
] – закрыть ветвь (подробнее см. ниже)
+ – увеличить угол *a* на величину *q*
- – уменьшить угол *a* на величину *q*
Размер шага и величина приращения по углу *q* задаются заранее и остаются неизменными для всех перемещений черепашки.
Команды ветвления (обозначаются], [) используются для построения деревьев растений, а вспомогательные переменные (обозначаются X, Y и т.д.) существенно облегчают построение некоторых L-систем.
Формально, детерминированная L-система состоит из *алфавита*, слова инициализации, называемого *аксиомой* или *инициатором*, и набора *порождающих правил*, указывающих, как следует преобразовывать слово при переходе от уровня к уровню (от итерации к итерации). Символы +, -,], [при подстановки правил порождения не заменяются, а просто остаются на тех местах, где они встретились.

47. TL-система строит **1.** фрактальные множества **2.** объекты

Ответ: 1, 2

48. Динамические автоматы определяются

1. Фрактальными множествами **2.** Суперпозицией push и pop 3. Грамматиками

Ответ: 1, 2, 3 (?)

Д-автоматы ведут себя подобно МП-автоматам (МП – магазинная память). При работе Д-автоматы используют вместо стека фрактальные множества подобные пыли Кантора, так как они достаточно просто могут быть построены на основе СИО.
Для однозначного определения Д-автоматов подмножества фрактала должны быть изолированы друг от друга, каждому подмножеству поставим в соответствие символ стека. Вершина стека - всегда символ, соответствующий текущему подмножеству. Тогда Д-автомат кодирует точки фрактала, идентифицируя вершину стека и определяет, какое подмножество активно.
Точки траектории x соответствуют один-к-одному состоянию стека с алфавитом $\Sigma=\{1,...,N\}$. Текущая траектория x фрактала соответствует текущему состоянию стека.
Аналогами функций $push_i$ и pop_i являются: для $push_i$ символ $a_j \in \Sigma$ должен изменить состояние автомата с $\rho_i(x)$ на $\rho_{i+1}(x)$; для pop символ a_j может быть выполнен, только когда текущее подмножество фрактала – это a_j (такое условие может быть предписано правилами переписывания) и оно состоит из изменения состояния с $\rho_i(x)$ на $\rho_{i-1}(x)$.

Автомат Мура - Суперпозицией push и pop
Автомат на основе треугольника Серпинского – Грамматиками
Все КС-языки могут быть распознаны Д-автоматом, построенным по аналогии с МП-автоматом.
Так как любой КС язык может быть распознан МП-автоматом, то можно построить механизм распознавания любого КС языка Д-автоматом.

Заключение: Динамические автоматы организует вычисления на основе фрактальных множеств и операторов разбиения и сдвига. Класс распознает КС языки, позволяет избежать сложности в хранении траектории и состояния автомата, интерпретирует значения метрического пространства в виде состояний и энтропии.
Д-автоматы ведут себя подобно МП-автоматам и имеют подобные сходства с нейро-сетями. Д-автоматы отображают пространственные отношения между автоматами на метрическом пространстве.