**Билет 1**.

**Детранслятор** преобразует программу с языка более низкого уровня к языку более высокого уровня. **Дизассемблер** – детранслятор на язык ассемблера.

**Недостижимые состояния КА**. Состояние $q\in Q$ КА $M(Q,V,δ,q\_{0},F)$ недостижимое, если $∀ω\in V^{+}$ невозможен переход КА из начального состояния в состояние q.

**Форма Бэкуса-Наура**. Основной словарь – основные символы. Вспомогательный словарь – металингвистические переменные. В левой части формулы (формы) – металингвистическая переменная, обозначающая соответствующую конструкцию. в правой – способы построения конструкции. Варианты в правой части разделяются символом |, обозначающем ИЛИ. Правая и левая части разделяются связкой $∷=$, что означает «определяется как». Имена металингвистических переменных заключаются в угловые скобки <>.

**4. Синтаксически управляемые грамматики.**

**Билет 3.**

**1. Этап лексического анализа.**

**2. Эквивалентность грамматик.**

**3. Лемма о разрастании КС-грамматик.**

**4. Этап распределения памяти.**

**Билет 11**.

**Языки низкого уровня**. Они машинно-ориентированные. Ассемблеры с макрогенераторами. Машинные коды. Ассемблеры.

**Регулярные выражения**. РМ обозначаются регулярными выражениями. РМ – множество цепочек, а РВ – формула, схематично показывающая, как было построено соответствующее ей РМ с пом. допустимых операций. Если p и q – РВ, обозначающие РМ P и Q, то p+q (p|q), pq, $p^{\*}$ - РВ, обозначающие РМ $P∪Q,PQ,P^{\*}$.

**Понятие цепных правил**. КС грамматика G(N, T, P, S) без циклов, если в ней нет выводов вида $A⇒^{\*}A,A\in N$. Циклы возможны только если в грамматике есть цепные правила вида $A\rightarrow B,A,B\in N$.

**LALR(1)-грамматика**. Представляет собой расширение алгоритма SLR(1). В ряде случаев работает тогда, когда построение SLR(1) таблицы разбора для данной грамматики невозможно из-за конфликтов сдвиг-свертка или свертка-свертка. Пусть есть грамматика, не разбираемая из-за конфликтов сдвиг-свертка или свертка-свертка по алгоритму SLR(1). В этом случае грамматика преобразуется следующим образом: ищется нетерминал, на котором возникла вызвавшая конфликт свертка. Обозначим его A. вводятся новые нетерминалы A1, A2, ..., An, по одному на каждое появление A в правых частях правил. Везде в правых частях правил A заменяется на соответствующее Ak. Набор правил с A в левой части повторяется n раз по разу для каждого Ak. правила с A в левой части удаляются, тем самым полностью удаляя A из грамматики. Используется в генераторе синтаксических анализаторов YACC.

**Билет 12**.

**Языки высокого уровня**. Непроцедурные, процедурные. 4GL: генераторы приложений, языки представления информации.

**Регулярные множества**. Пусть дан алфавит A и $P⊆A^{\*},Q⊆A^{\*}⇒$ **конкатенация** $PQ=\left\{pq | ∀p\in P,∀q\in Q\right\}$, **итерация** $P^{\*}=$

$=\left\{p^{n} | ∀p\in P\right\}$ $⇒$ для алфавита A РМ определяются рекурсивно: $∅$ - РМ; $\left\{λ\right\}$ – РМ; $\left\{a\right\}$ – РМ $∀a\in A$; если P и Q произвольные РМ $⇒$

$⇒P∪Q,PQ,P^{\*}$ - РМ; Ничто другое не явл. РМ. Множество P **замкнуто** относительно операции $Ω$, если $Ω\left(p,q\right)\in P ∀p,q\in P$. РМ замкнуты относительно операций пересечения, объединения, дополнения, итерации, конкатенации, гомоморфизма.

**НФ Хомского**. Каждая КС грамматика G(N,T,P,S) эквивалентна КС грамматике $G^{'}(N^{'},T,P^{'},S)$ в НФ Хомского, все порождающие правила из множества $P^{'}$ имеют вид $A\rightarrow BC | A\rightarrow a,A,B,C\in N^{'},a\in T$; либо $S\rightarrow λ$, если $λ\in L(G)$ и S не встречаются в правых частях правил.

**SLR(1) грамматика** – подкласс грамматик LALR(1). Расширяет грамматику LR(0). Применяется, если при построении распознавателя для LR(0) – грамматики возникают конфликты «сдвиг/свертка».

**Билет 13**.

**Языки 4GL**. Машинно-независимые, высокого уровня: непроцедурные, процедурные, **4GL** (генераторы приложений, языки представления информации).

**НФ Грейбах** – форма представления $G(N,T,P,S)$ все порождающие правила из множества P имеют вид $A\rightarrow bα,A\in N,b\in T,α\in N^{\*}$ либо $S\rightarrow λ$, если $λ\in L\left(G\right)$ и S не встречаются в правых частях правил.

**Свойства регулярных языков**. Бывают леволинейные, праволинейные. Если язык L(G) регулярный, то он КС. Для регулярных языков распознаватель – конечный автомат. Время распознавания линейно зависит от длины входной цепочки символов. В компиляторах конечные автоматы используются для выделения в исходном коде лексем, что позволяет сократить объем входной информации для синтаксического анализатора.

**Понятие ситуации**. Это множество правил КС-грамматики, в кот. указывается положение считывающей головки МП-автомата, кот. может возникнуть при разборе сентенциальной формы этой грамматики.

**Билет 15**.

**Понятие цепочки символов** – произвольная упорядоченная конечная последовательность символов алфавита.

**Соответствие распознавателей классам грамматик**. **4 типа**.

Грамматика с фраз. структурой – недет., 2 ст., неогр. пам. (машина Тьюринга) распознаватель; КЗ – недет., 2-ст., лин. огр. пам.; КС – недет., 1-ст., магаз. пам.; Рег. – недет., 1-ст., без памяти.

**МПА** $R(Q,A,Z,δ,q0,z0,F)$, где Q – множество состояний автомата, A – алфавит входных символов, Z – специальный конечный алфавит магазинных символов автомата, $A⊆Z$, $δ$-ф-ция переходов, q0 – начальное состояние автомата, z0 – начальный символ магазина, $F⊆Q$ – множество конечных состояний. В отличие от КА МПА имеет стек, в него помещаются терминальный и нетерминальные символы. **Расширенный МПА** может заменять не один символ, а цепочку на вершине стека. Ф-ция переходов для него отображает множество $Q×(A∪\{λ\})×Z^{\*}$. МПА допускает цепочку с **опустошением магазина**, если по окончании разбора автомат находится в одном из конечных состояний, а магазин пуст ($q,λ,λ$).

**Пополненная КС грамматика** строится на основе грамматики G(N,T,P,S): $G^{'}=G$, если не $∃p\in P:A\rightarrow αSβ,A,S\in N,α,β\in V^{\*}$ (аксиома S не встречается в правых частях правил);

$G^{'}=\left\{N∪\left\{S^{'}\right\},T,P∪\left\{S^{'}\rightarrow S\right\},S^{'}\right\}$, если $∃p\in P:A\rightarrow αSβ,$

$A,S\in N,α,β\in V^{\*}$ (аксиома S встречается в правых частях правил). Пополненная грамматика нужна, чтобы свертка к аксиоме служила распознавателю сигналом завершения алгоритма и не возникало неоднозначности (продолжать разбор или нет, если была выполнена свертка к аксиоме).

**Билет 16**.

**Операции над цепочками**. **Длина цепочки** – кол-во символов в ней: $\left|α\right|$. **Конкатенация** $θ$ цепочек $α, β$ - цепочка $γ=αθβ=αβ$. Цепочка $α$ явл. префиксом, $β$ – суффиксом строки $γ$. **Обращение** цепочки $α^{R}$ – запись символов цепочки $α$ в обратном порядке. **Итерация** цепочки $α^{n}$ – конкатенация цепочки $α$ самой с собой n раз, $n\in N$, $n\geq 0$.

**Типы грамматик по Хомскому**.

**Тип 0. Грамматики с фразовой структурой**. На структуру правил не накладывается никаких ограничений. Для грамматики вида $G\left(T,N,P,S\right),V=N∪T$ правила имеют вид: $α\rightarrow β,α\in V^{+},β\in V^{\*}$.

**Тип 1. Контекстно-зависимые (КЗ)**. $α\_{1}Aα\_{2}\rightarrow α\_{1}βα\_{2};α\_{1},α\_{2}\in V^{\*}$, $A\in N,β\in V^{+}$. Один и тот же нетерминальный символ может быть заменен на ту или иную цепочку символов в зависимости от контекста, в котором он встречается. **Неукорачивающие**: $α\rightarrow β,α,β\in V^{+},\left|β\right|\geq \left|α\right|$.

**Тип 2. Контекстно-свободные**. $A\rightarrow β,A\in N,β\in V^{+}$. КС грамматики имеют в правой части как минимум 1 символ. Они неукорачивающие.

**Тип 3. Регулярные грамматики**. Леволинейные грамматики $A\rightarrow Bγ$, $A\rightarrow γ,A,B\in N,γ\in T^{\*}$. При выводе нетерминальный символ если и остается, то слева. Праволинейные грамматики $A\rightarrow γB,A\rightarrow γ,$

$A,B\in N,γ\in T^{\*}$. Леволинейные и праволинейные эквивалентны.

**Понятие лямбда-переходов**. При выполнении такта в стеке заменяется символ, соответствующий условию перехода, на цепочку, соответствующую правилу перехода. Первый символ этой цепочки становится новой вершиной стека. Автомат может и не извлекать символ из стека. Допускаются переходы, при кот. входной символ игнорируется, оставаясь на след. такт. Такие переходы наз-ся $λ$ – переходами.

**Понятие основы**. Основа цепочки $α$ – это вхождение правой части последнего правила в правом выводе этой цепочки. Это самая левая подлежащая свертке подстрока сентенциальной формы правостороннего вывода.

**Билет 18**.

**Операция обращения**. **Обращение** цепочки $α^{R}$ – запись символов цепочки $α$ в обратном порядке.

**Леволинейные и праволинейные грамматики**. Леволинейные грамматики $A\rightarrow Bγ$, $A\rightarrow γ,A,B\in N,γ\in T^{\*}$. При выводе нетерминальный символ если и остается, то слева. Праволинейные грамматики $A\rightarrow γB,A\rightarrow γ,A,B\in N,γ\in T^{\*}$. Леволинейные и праволинейные эквивалентны.

**Конфигурация МПА**. Определяется состоянием автомата, цепочкой еще не прочитанных символов, содержимым стека ($q,α,ω$). При выполнении такта в стеке заменяется символ, соответствующий условию перехода, на цепочку, соответствующую правилу перехода. Первый символ этой цепочки становится новой вершиной стека.

**Условие определения LL(1) грамматики**. КС грамм. наз. LL(1) грамм., если множества напряавляющих символов для правил, определяющих один и тот же нетерм. грамматики, не пересекаются $∀A\in N:A\rightarrow α\_{1}\left|α\_{2}\right|…|α\_{n}\in P,DS\left(A,α\_{i}\right)∩DS\left(A,α\_{j}\right)=∅,$

$i\ne j,i,j=1,2,..,n$.

**Билет 22**.

**Формальный язык** - множество всех цепочек терминальных символов, выводимых из аксиомы формальной грамматики. $L\left(G\right)=$

$=\left\{χ | S⇒^{\*}χ,χ\in T^{\*}\right\}$.

**Регулярная грамматика**. Леволинейные грамматики $A\rightarrow Bγ$, $A\rightarrow γ,A,B\in N,γ\in T^{\*}$. При выводе нетерминальный символ если и остается, то слева. Праволинейные грамматики $A\rightarrow γB,A\rightarrow γ,$

$A,B\in N,γ\in T^{\*}$. Леволинейные и праволинейные эквивалентны.

**«Сдвиг-свертка»**. Восходящий распознаватель разбора цепочки просматривает ее слева направо и порождает правосторонний вывод. Дерево вывода строится от листьев к корню. Тут используется сдвиг-свертка. Сдвиг добавляет очередной символ строки в стек. Свертка производит замену верхних символов стека, совпадающих с правой частью правила, на нетерминал левой части.

**LL(1) – грамматика**. Входная цепочка символов читается слева направо, используется левосторонний вывод, число предварительно просматриваемых символов входной строки для выбора очередного правила грамматики - 1. Множество направляющих символов $DS(A,α)$ для правила $A\rightarrow α$, $A\in N,α\in V^{\*}$ $DS\left(A,α\right)=S(α)∪F(A)$. $S(α)$ это множ. термин. цепочек, выводимых из непустой строки $α$, укороченных до k символов. $F\left(A\right)$ это множ. термин. цепочек, кот. следуют в сентенциальных формах непосредственно за нетермин. А, укороченных до k символов.

**Билет 23**.

**КА**. $M(Q,V,δ,q0,F)$. Q – конечное множество состояний, V – конечное множество допустимых входных сиволов, $δ$ – ф-ция переходов, q0 – начальное состояние автомата, F – непустое множество конечных состояний. КА полностью определен, если в каждом его состоянии сущ. ф-ция перехода для всех входных символов: $∀a\in V,∀q\in Q∃δ\left(a,q\right)=R,R⊆Q$. В начале работы автомат всегда нах-ся в состоянии q0. На каждом такте он под воздействием очередного символа входной цепочки либо остается в текущем. Два КА эквивалентны, если они задают 1 и тот же язык.

**Тип 0. Грамматика с фразовой структурой**. На структуру правил не накладывается никаких ограничений. Для грамматики вида $G\left(T,N,P,S\right),V=N∪T$ правила имеют вид: $α\rightarrow β,α\in V^{+},β\in V^{\*}$.

**3. Замыкание алфавита**

**4. ???**

**Билет 24**.

**Понятие алфавита** – счетное множество допустимых символов языка.

**Принцип работы лексического анализатора**. Это процесс предварительной обработки исходной программы, на котором основные лексические единицы программы – лексемы – приводятся к единому формату и заменяются условными кодами или ссылками на соответствующие таблицы. Результат работы ЛА – поток образов лексем-дескрипторов и таблицы, в которых хранятся значения выделенных в программе лексем.

3. Формализация КС грамматики.

**LR(0) и LR(1) грамматики**. Входная цепочка символов читается слева направо, используется правосторонний вывод, текущий символ просматриваемой цепочки не участвует в анализе. При выполнении свертки к произв. нетерминалу, в стеке перед ним будут находиться только те символы, кот. могут встретиться слева от него. Это т.н. левый контекст. Для LR(0) грамматики во внимание принимается только он.

**Билет 25**.

**Синтаксис, лексика, семантика языка**. **Лексика** – совокупность слов (словарный запас) языка. **Синтаксис** – набор правил, определяющий допустимые конструкции языка, т.е. определение набора цепочек символов, принадлежащих языку. **Семантика** – это раздел языка, определяющий значение предложений языка (смысл всех допустимых цепочек языка).

**Распознаваемость языка**. Язык L(G) называется **распознаваемым**, если существует алгоритм, который за конечное число шагов позволяет определить, принадлежит ли произвольная цепочка $α$ над основным словарем грамматики языку L(G). Если при этом число шагов алгоритма зависит от длины цепочки и может быть оценено до начала его выполнения, язык L(G) называется **легко распознаваемым**.

**Эквивалентность КА**. 2 КА эквивалентны, если они задают один и тот же язык.

**Множество направляющих цепочек** $DS(k,A,α)$ для правила $A\rightarrow α$,

$A\in N,α\in V^{\*}$ определим как $DS\left(k,A,α\right)=S(k,α)∪F(k,A)$. Здесь $S(k,α)$ – множество цепочек-предшественников для строки $α$ длины k, определяемое как $S\left(k,α\right)=\left\{β | β\in T^{\*},\left|β\right|=k,α\in V^{+},\right\}$

$α⇒^{\*}βγ,γ\in V^{\*}\}$. Это множ. термин. цепочек, выводимых из непустой строки $α$, укороченных до k символов. $F\left(k,A\right)$ – множ. цепочек-последователей для нетерм. А длины k $F\left(k,A\right)=\left\{β \right|β\in T^{\*},\left|β\right|=k,$

$S⇒^{\*}ωAβγ,S-аксиома,A\in N,ω,γ\in V^{\*},A⇒^{\*}λ\}$. Это множ. термин. цепочек, кот. следуют в сентенциальных формах непосредственно за нетермин. А, укороченных до k символов.

**Билет 29**.

**Интерпретатор** не формирует готовой программы, а выполняет исходный текст программы по частям.

**Принцип рекурсии**. Явная – сивол определяется сам через себя. Неявная – символ определяется через цепочку правил: $A\rightarrow Dx,D\rightarrow A$ Чтобы рекурсия не была бесконечной, должны сущ. и другие правила, определяющие тот же символ.

**Граф переходов КА**. Это ориентированный граф, в кот. состояния КА – вершины, а переходы $δ\left(p,a\right)=q,a\in V,pq\in Q$ – дугам (p,q), помеченным символом a.

**Распознаватели без возвратов**. Основаны на определении метода, по которому выбирается одна из возможных альтернатив. Остальные альтернативы не учитываются. Время работы обладает линейной, а не экспоненциальной зависимостью от длины входной цепочки. Такие алгоритмы могут потребовать дополнительных ограничений на правила грамматики.

**Билет 30**.

**Кросс-компилятор** выполняет трансляцию программы на одной платформе, формируя объектный код для др. платформы.

**Синтаксические диаграммы**. В этой форме каждому нетерминалу соответствует диаграмма в виде направленного графа. Типы вершин: точка входа. На диаграмме не обозначается, из нее просто начинается входная дуга графа нетерминал. На диаграмме обозначается прямоугольником, в котором написано его обозначение. Цепочка терминалов. Обозначается овалом, внутри которого записана цепочка. Узловая точка. Обозначается точкой или закрашенным кружком. точка выхода. Никак не обозначается, в нее просто входит выходная дуга графа.

**Минимизация КА**. Это построение эквивалентного КА с меньшим числом состояний. Устранение недостижимых состояний, объединение эквивалентных состояний. Состояние $q\in Q$ КА $M(Q,V,δ,q\_{0},F)$ недостижимое, если $∀ω\in V^{+}$ невозможен переход КА из начального состояния в q.

**S – грамматика** – КС грамматика с требованиями. $∀p\in P:A\rightarrow aβ$,

$A\in N,a\in T,β\in V^{\*}$; т.е. правая часть правил начинается с терминала. $∀p\in P:A\rightarrow a\_{1}β\_{1} \left| a\_{1}β\_{1} \right|… | a\_{n}β\_{n},A\in N,a\_{i}\in T,β\_{i}\in V^{\*},a\_{i}\ne a\_{j}$,

$i,j=1,2,…,n;i\ne j$; т.е. правые части правил, определяющие 1 и тот же нетерминал, начинаются с разных терминалов.

№ ???

**Замыкание алфавита** $A\left(A^{\*}\right)$ - множество всех возможных цепочек над алфавитом A.

**Классификация КС грамматик по Хомскому**.

**Контекстно-свободные**. $A\rightarrow β,A\in N,β\in V^{+}$. КС грамматики имеют в правой части как минимум 1 символ. Они неукорачивающие. Почти эквивалентый им класс – укорачивающие КС грамматики $A\rightarrow β,$

$A\in N,β\in V^{\*}$. Их используют для построения синтаксического анализатора компиляторов.

**Синтаксический анализ снизу-вверх**. В основе лежит правосторонний разбор. Исходной сентенциальной формой является разбираемая строка языка, а целью – аксиома. Выполняется последовательность операций перенос и свертка. Перенос добавляет очередной символ строки в стек. Свертка производит замену верхних символов стека, совпадающих с правой частью правила, на нетерминал левой части. Если в результате серии этих операций получена аксиома грамматики, то распознаватель принял входную строку, иначе строка не принадлежит языку. Исходная грамматика не должна содержать циклов и -правил.

**Множество направляющих символов** $DS(A,α)$ для правила $A\rightarrow α$,

$A\in N,α\in V^{\*}$ $DS\left(A,α\right)=S(α)∪F(A)$. $S(α)$ – множество символов-предшественников $S\left(α\right)=\left\{b \right|b\in T,α\in V^{+},a⇒^{\*}bγ,γ\in V^{\*}\}$.

$F\left(A\right)=\left\{b | b\in T,S⇒^{\*}ωAbγ,S-аксиома,A\in N,ω,γ\in V^{\*},A⇒^{\*}λ\right\}$ – множество символов-последователей.

№ ???

**Операция итерации -** цепочки $α^{n}$ – конкатенация цепочки $α$ самой с собой n раз, $n\in N$, $n\geq 0$.

**Контекстно-зависимые грамматики**. $α\_{1}Aα\_{2}\rightarrow α\_{1}βα\_{2};$

$α\_{1},α\_{2}\in V^{\*},A\in N,β\in V^{+}$. Один и тот же нетерминальный символ может быть заменен на ту или иную цепочку символов в зависимости от контекста, в котором он встречается. **Неукорачивающие**: $α\rightarrow β$, $α,β\in V^{+},\left|β\right|\geq \left|α\right|$. КЗ и неукорачивающие грамматики эквивалентны.

**Разбор с возвратом**. Алгоритм запоминает все возможные следующие состояния, выбирает одно из них, переходит в него и так до тех пор, пока не будет достигнуто конечное состояние, тогда строка принимается либо до тех пор, пока автомат не перейдет в такую конфигурацию, когда следующее состояние не будет определено, в этом случае автомат возвращается на несколько шагов назад, где возможен выбор другого варианта следующего состояния, выбирает другой вариант и продолжает работу.

**Множество символов предшественников**. $S(α)$ – множество символов-предшественников $S\left(α\right)=\left\{b \right|b\in T,α\in V^{+},a⇒^{\*}bγ,γ\in V^{\*}\}$.

**Билет Unknown1.**

**1. Назначение компоновщика**

**2. Способы задания рег. языков**

**3. Рекурсивность грамматики**

**4. Грамматики операторного предшествования**

**Билет Unknown2\_v1.**

**1.классификация трансляторов**

**2.виды распознователей**

**3.отличие детенрменированного и недедерменированного КА**

**4.LR(k) грамматика**

**Билет Unknown2\_v2.**

**1.классификация трансляторов**

**2.llr(k)граматика(в общем рассказать)**

**3.КА**

**4. классификация автоматов вроде(право- левосторонний какой вывод и тд)**

**Билет Unknown3.**

**1. Синтаксический анализ сверху вниз**

**2. Неукорачивающие грамматики**

**3. Множество символов последователей**

**4. Операция подстановки**

**Билет Unknown4.**

**1. формальные грамматики.**

**2. современные системы программирования.**

**3. однонаправленные КС, или что-то такое**

**4. ???**

**Билет Unknown5.**

**1. Процедурные языки программирования**

**2. Свойства регулярных выражений**

**3. Цикл**

**4. Классификация КС грамматик**