**Методический материал для лабораторной работы № 4 . Понятие приведенной и неоднозначной КС-грамматики. Устранение из КС-грамматики бесполезных символов и** ε **-правил.**

**Определение.** Грамматика G = (T, V, P, S) называется контекстно-свободной ( КС ), если  
каждое правило из Р имеет вид A → β, где A ∈ V, β ∈ (T ∪ V)+

КС-грамматика – грамматика, в которой левые части всех правил состоят ровно из одного нетерминала.

Цепочка в алфавите T принадлежит языку, порождаемому грамматикой 〈 T, V, P, S〉,  
только в том случае, если существует ее вывод из начального символа S этой грамматики.  
Процесс построения такого вывода (а, следовательно, и определения принадлежности цепочки языку) **называется разбором**. Построение вывода можно осуществлять и в обратном порядке: в исходной цепочке ищем вхождение правой части некоторого правила и заменяем его на левую часть (это называется **сверткой**), в результате исходная цепочка «сворачивается» к некоторой сентенциальной форме, затем идет следующая свертка и т.д., пока не придем к цели грамматики — S . **Процесс разбора называют также анализом**.  
С практической точки зрения наибольший интерес представляет разбор по контекстно-свободным грамматикам. Их порождающей мощности достаточно для описания большей части синтаксической структуры языков программирования, для различных подклассов.  
КС-грамматик имеются хорошо разработанные практически приемлемые способы решения задачи разбора. Рассмотрим основные понятия и определения, связанные с разбором по КС-грамматике.

**Определение:** вывод цепочки β ∈ T \* из S ∈ V в КС-грамматике G = 〈 T, V, P, S 〉, называется левым (**левосторонним)**, если в этом выводе каждая очередная сентенциальная форма получается из предыдущей заменой самого левого нетерминала ( не путать с левосторонней регулярной грамматикой).

**Определение:** вывод цепочки β ∈ T \* из S ∈ N в КС-грамматик G = 〈 T, V, P, S 〉, называется правым (**правосторонним)**, если в этом выводе каждая очередная сентенциальная форма получается из предыдущей заменой самого правого нетерминала.  
В грамматике для одной и той же цепочки может быть несколько выводов, эквивалентных в том смысле, что в них в одних и тех же местах применяются одни и те же правила вывода, но в различном порядке (не путать с правосторонней регулярной грамматикой).  
Например, для цепочки **a + b + a** в грамматике:  
**Gexpr = 〈 {a, b, +}, {S, T}, {S → T | T + S ; T → a | b}, S 〉**можно построить выводы:  
(1) S → T + S → T + T + S → T + T + T → a + T + T → a + b + T → a + b + a  
(2) S → T + S → a + S → a + T + S → a + b + S → a + b + T → a + b + a  
(3) S → T + S → T + T + S → T + T + T → T + T + a → T + b + a → a + b + a

Здесь (2) — левосторонний вывод, (3) — правосторонний, а (1) не является ни лево-  
сторонним, ни правосторонним, но все эти выводы являются эквивалентными в указанном  
выше смысле.  
Для КС-грамматик можно ввести удобное графическое представление вывода, называемое **деревом вывода**, причем для всех эквивалентных выводов деревья вывода совпадают.

**Определение:** ориентированное упорядоченное дерево называется **деревом вывода**(или деревом разбора) в КС-грамматике G = 〈 T, N, P, S 〉, если выполнены следующие условия:  
(1) каждая вершина дерева помечена символом из множества V ∪ T ∪ {ε}, при этом  
корень дерева помечен символом S; листья — символами из T ∪ {ε};  
(2) если вершина дерева помечена символом A, а ее непосредственные потомки —  
символами a1, a2, ..., an, где каждое ai ∈ T ∪ V, то A → a1a2...an — правило вывода  
в этой грамматике;  
(3) если вершина дерева помечена символом A, а ее непосредственный потомок помечен символом ε, то этот потомок единственный и A → ε — правило вывода в  
этой грамматике.

На **рисунке 1** изображен пример дерева вывода для цепочки a + b + a в грамматике Gexpr.

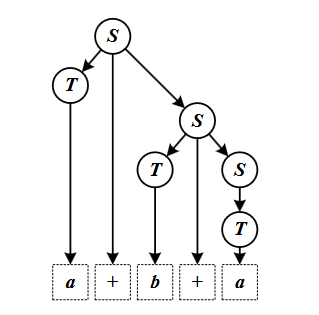


Рис 1 . Пример дерева вывода в грамматике Gexpr.

Если для каждой цепочки символов языка, заданного грамматикой, можно построить  
единственный левосторонний (и единственный правосторонний) вывод или, что то же самое,  
построить единственное дерево вывода, то такая **грамматика называется однозначной**. Иначе  
грамматика называется неоднозначной.

**Правила вывода, по которым можно определить, что КС-грамматика неодназначная.**

Для КС-грамматик существуют определённого вида правила, по наличию которых во всём  
множестве правил грамматики можно утверждать, что она является неоднозначной:  
1. А → АА | α  
2. А → АαА | β  
3. А → αА | Aβ | γ  
4. А → αА | αAβA | γ  
Здесь **A ∈ V; α, β, γ ∈ (V ∪ T)\*.**  
**Если в заданной грамматике встречается хотя бы одно из вышеперечисленных правил подобного вида, то такая грамматика точно будет неоднозначной. Однако отсутствие подобных правил во всём множестве правил грамматики не означает, что данная грамматика является однозначной**.  
**Пример неоднозначной грамматики.** Условный оператор, включённый во многие языки программирования, описывается с помощью грамматики G с правилами:  
S → if b then S else S | if b then S | a  
где **b** – логическое выражение, **а** – безусловный оператор. Эта грамматика неоднозначна, т.к. в  
ней возможны два левых вывода цепочки **if b then if b then a else a**:  
1) S ⇒ if b then S else S ⇒ if b then if b then S else S ⇒ if b then if b then a else S ⇒ if b then if b then  
a else a  
2) S ⇒ if b then S ⇒ if b then if b then S else S ⇒ if b then if b then a else S ⇒ if b then if b then a else  
a

В этой грамматике для цепочки **if b then if b then a else a** можно построить два различных дерева вывода, изображенных на рисунке 2 (а, б).

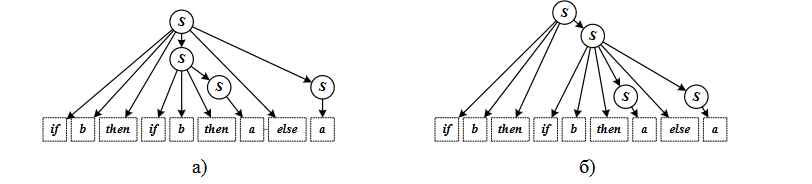


Рис.2. Деревья вывода для «if b then if b then a else a» в грамматике G.

Для грамматики из примера можно также показать, что она является неоднозначной, поскольку она содержит **правила четвёртого вида**.

В некоторых случаях КС-грамматика может содержать недостижимые и  
бесплодные символы, которые не участвуют в порождении цепочек языка и  
поэтому могут быть удалены из грамматики.

**Определение:** символ x ∈ (VT ∪ VN) называется недостижимым в  
грамматике G = (VT, VN, P, S), если он не появляется **ни в одной сентенциальной  
форме этой грамматики**.

**Определение:** символ A ∈ VN называется бесплодным в грамматике  
G = (VT, VN, P, S), если множество { α ∈ VT \* | A α} пусто ( **бесплодными символами могут быть только нетерминалы**).

**Определение:** КС-грамматика G называется приведенной, если в ней нет  
недостижимых и бесплодных символов.

**Алгоритм приведения грамматики:**

1. обнаруживаются и удаляются все бесплодные нетерминалы.
2. обнаруживаются и удаляются все недостижимые символы.

***Удаление символов сопровождается удалением правил вывода, содержащих  
эти символы.***

**Замечание:** если в этом алгоритме переставить шаги (1) и (2), то не всегда  
результатом будет приведенная грамматика. Для описания синтаксиса языков программирования стараются использовать однозначные приведенные КС-грамматики.

**Примечание.** Если начальный символ грамматики окажется бесплодным, то следует  
удалить содержащие его правила, а сам символ оставить в алфавите нетерминалов, так  
как, по определению грамматики, множество нетерминалов обязано содержать начальный символ.

**Не приведенная G1 грамматика с правилами P**

S → A | B { A – бесплодный нетерминал}

B→ b | BC

C→ CC

D→ а { a- не достижимый терминал - не появляется ни в какой сентенциальной цепочке}

**Приведенная грамматика G1**

S → B

B→ b

**Для нахождения бесплодных и недостижимых символов  
полезен граф КС-грамматики:**

• каждому символу из T ∪ N соответствует единственная вершина, помеченная этим символом; если в P есть правило с пустой правой частью ε, то граф имеет вершину,  
помеченную ε ;

• вершина X соединяется с вершиной Y стрелкой (дугой), если  
в грамматике есть правило X→αYβ, α,β∈(T∪N)\*;

• X соединяется с вершиной ε, если в грамматике есть правило  
X→ε .

**Алгоритм 1 удаления бесплодных символов:**

1. Отметить терминальные вершины (вершины, помеченные  
терминальными символами), а также вершину ε, если таковая имеется.

2. Если в Р есть правило А→α, где α состоит из уже  
отмеченных в графе символов, а вершина А не отмечена,  
то отметить эту вершину. Повторять шаг 2 пока возможно.

3. Из грамматики удалить неотмеченные символы и правила,  
их содержащие.

**Алгоритм 2 удаление недостижимых символов:**

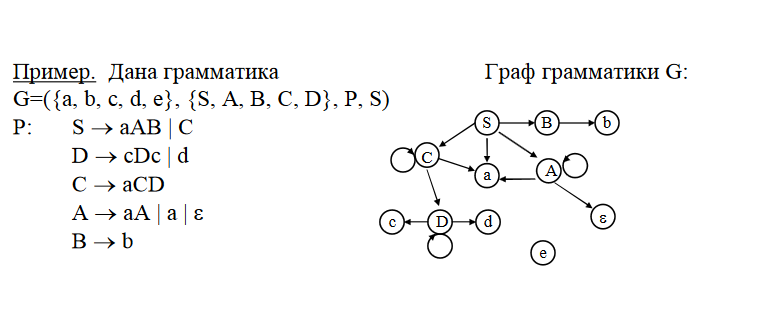
1. Отметить вершины, в которые есть путь из вершины S (достижимы из вершины S).

2. Удалить из грамматики неотмеченные символы и правила, их содержащие.

***Пример: из грамматики G получить приведенную грамматику G2.***

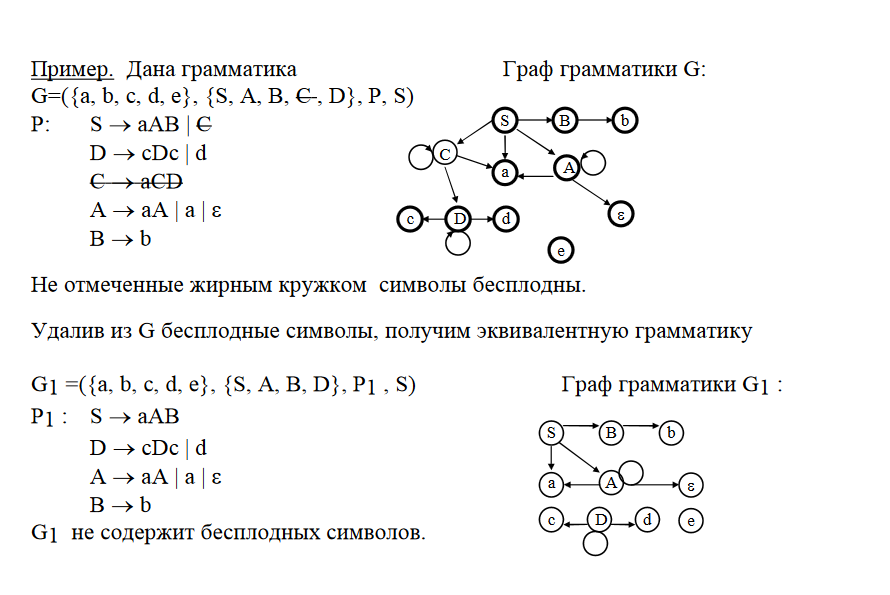
***Дана грамматика***G=({a, b, c, d, e}, {S, A, B, C, D}, P, S)  
P: S → aAB | C  
D → cDc | d  
C → aCD  
A → aA | a | ε  
B → b

**Шаг 1. Строим граф КС-грамматики. (рисунок 3)**

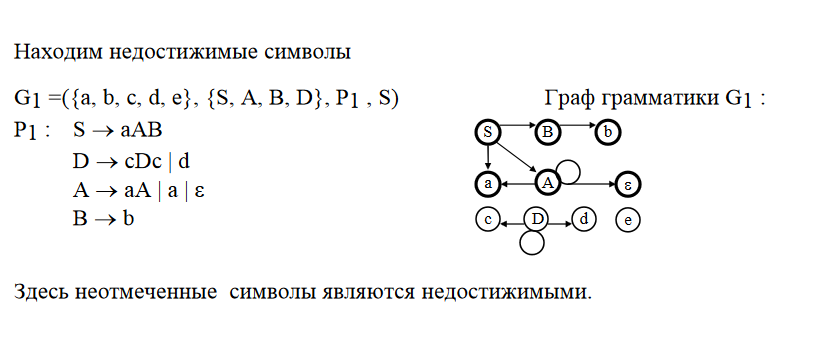


**Рис. 3 . Граф КС-грамматики**

**Шаг 2 . Удаляем из грамматики бесплодные символов ( рисунок 4 )**

**Рис. 4. Удаление из грамматики бесплодных символов в соответствии с алгоритмом 1**

**Шаг 3 . Удаляем из грамматики недостижимые символы ( рисунок 5 )**

**Рис. 5. Удаление из грамматики недостижимых символов.**

**Удалив из G1 недостижимые символы, получим эквивалентную грамматику G2:**

G2 =({a, b}, {S, A, B}, P2 , S)  
P2 : S → aAB

A → aA | a | ε  
B → b

L(G)=L(G1 )=L(G2 )={ an b | n≥1 }

G2 – приведенная однозначная грамматика, удалили бесплодные символы, а затем недостижимые.

Некоторые применяемые на практике алгоритмы разбора по КС-грамматикам требуют, чтобы в грамматиках не было правил с пустой правой частью, т. е. чтобы КС-грамматика  
была неукорачивающей. Для любой КС-грамматики существует эквивалентная неукорачивающая.

**Устранение правил с пустой правой частью из КС-грамматики**

1. Построить множество Х={A∈N | A⇒ε}.  
2. Удалить правила с пустой правой частью.  
3. Если S∈X, то S’ – новый начальный символ, S’→S | ε ∈P.  
4. ∀ A∈X правило вида B→α1A1α2A2...αnAnαn+1,  
где αi∈((N – X)∪T)\*  
заменить 2n правилами, соответствующими всем возможным комбинациям  
вхождений А между αi:  
B→α1α2...αnαn+1  
B→α1α2...αnAnαn+1  
...  
B→α1α2A2...αnAnαn+1  
B→α1A1α2A2...αnAnαn+1  
Замечание: если все αi=ε ∀ i=1,...,n+1, то правило B→ε не включать в новую  
грамматику.  
5. Удалить бесполезные символы и правила, их содержащие.

**Пример. Дана исходная грамматика G, получить эквивалентную грамматику G2, не содержащую правил с пустой правой частью.**

P:

S → BС|Ab

B→ε  
C→c  
 A → Aa | ε

* **Строим множество X= { A, B}, удаляем из грамматики G правила B**→ε **и A** →ε и получаем :  
  P:

S → BС|Ab

C→c  
 A → Aa

* **Применяем пункт 4 алгоритма и получаем:**

S → BС|С|Ab|b

C→c  
A → Aa|a

* **Применяем пункт 5 алгоритма и удаляем бесполезные символы и получаем эквивалентную грамматику G2**

P2 : S → С|Ab|b ( удаляем бесплодный символ B и правило его содержащее)

C→c  
 A → Aa|a