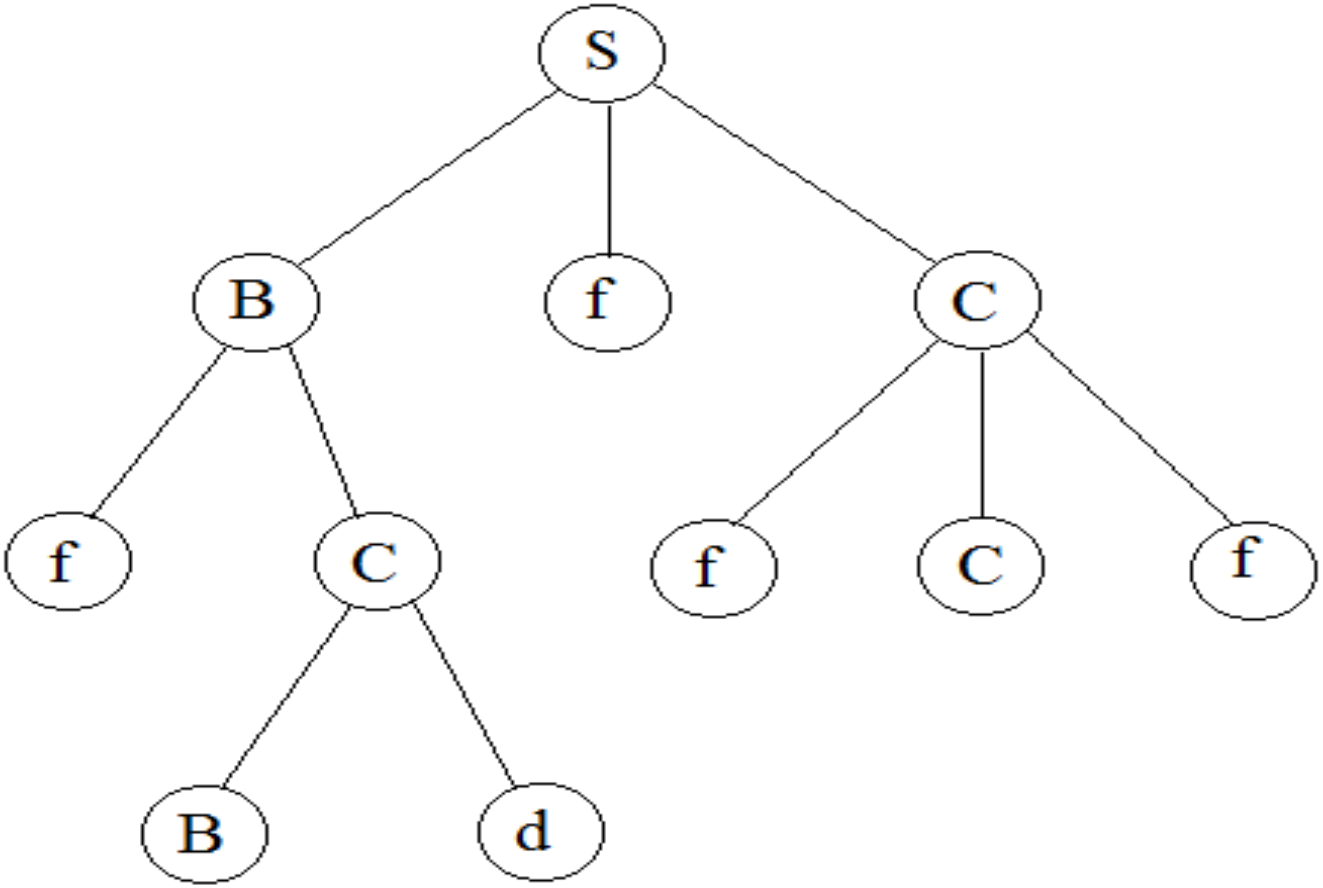
2.3. Синтаксические деревья

- Синтаксические деревья служат для отображения вывода некоторой сентенциальной формы.
- Синтаксическое дерево строится следующим образом.
- В качестве корня дерева выступает начальный символ грамматики. Далее рекурсивно выполняем следующее: если текущим узлом дерева является нетерминал, то в качестве дочерних узлов выступают символы правой части правила, используемого для замены этого нетерминала при выводе.
- Для рассмотренного выше вывода синтаксическое дерево имеет следующий вид:

Синтаксическое дерево



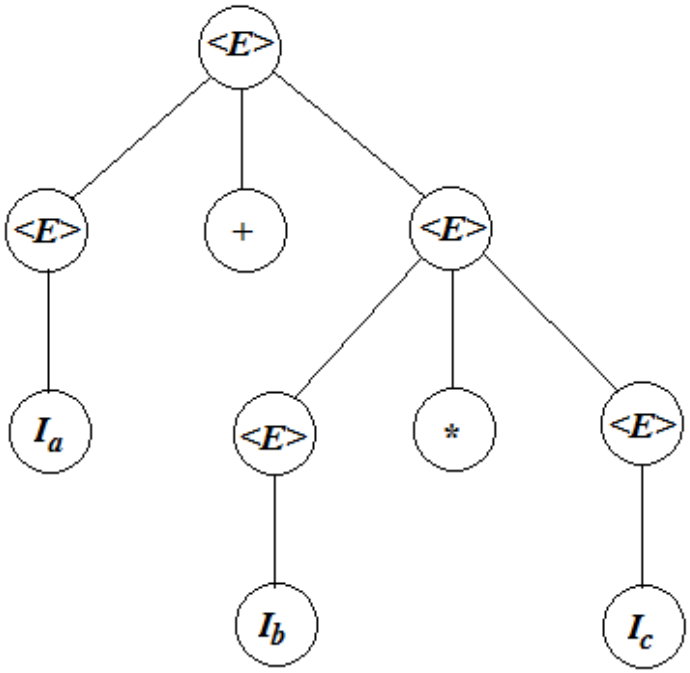
Правый и левый выводы

- Обход концевых узлов дерева слева направо даёт текущую сентенциальную форму.
- Для каждого вывода имеется соответствующее синтаксическое дерево, но несколько выводов могут иметь одно и то же дерево.
- Если в сентенциальной форме при выводе всегда заменять самый левый нетерминал, то мы получим левосторонний (или левый) вывод. Если заменять самый правый нетерминал, то получим правосторонний (правый) вывод. Каждый из этих выводов единственен.
- Предложение грамматики *неоднозначно*, если для его вывода существуют два или более синтаксических деревьев.
- Грамматика *неоднозначна*, если она допускает неоднозначные предложения, в противном случае она *однозначна*.

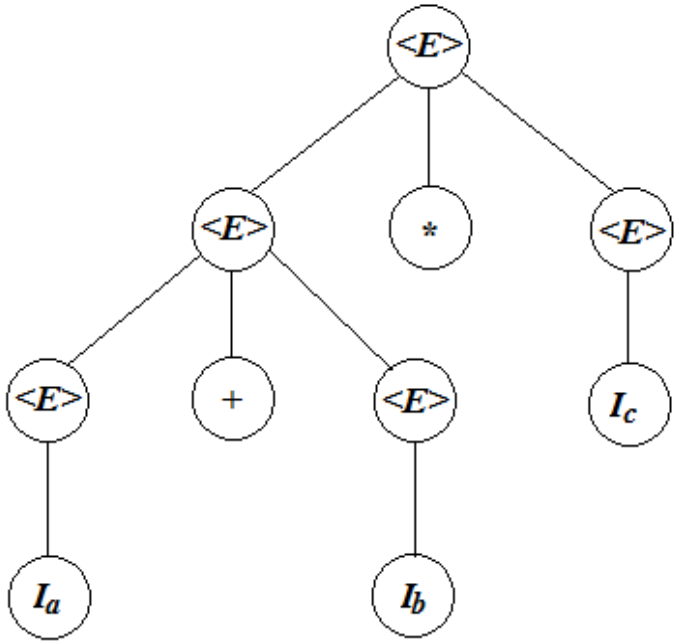
Пример

-
-
- $\langle E \rangle \rightarrow \langle E \rangle + \langle E \rangle$
- $\langle E \rangle \rightarrow \langle E \rangle * \langle E \rangle$
- $\langle E \rangle \rightarrow I$
-
- Возьмём выражение $a + b * c$.
-
- $I_a + I_b * I_c$,
-

Синтаксические деревья



а)



б)

2.4. Задача грамматического разбора

- *Разбор* сентенциальной формы означает построение вывода для неё, а также, возможно, построение синтаксического дерева.
- Программу разбора называют также *распознавателем*, т.к. она распознаёт предложения заданной грамматики.
-
- Различают две категории алгоритмов разбора: *нисходящие* (сверху вниз) и *восходящие* (снизу вверх).

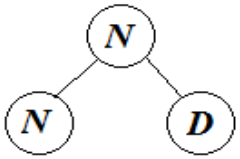
Пример

- ***N → D | ND***
- ***D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9***
- Пусть дано предложение (число) ***35***.

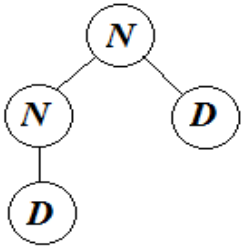
- ***N => ND => DD => 3D => 35***

Нисходящий разбор

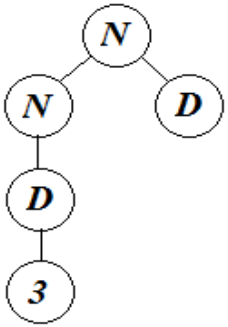
N ==> ND



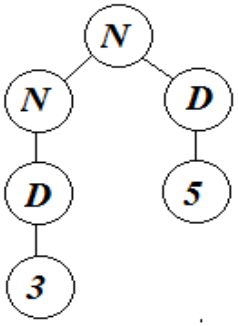
ND ==> DD



DD ==> 3D



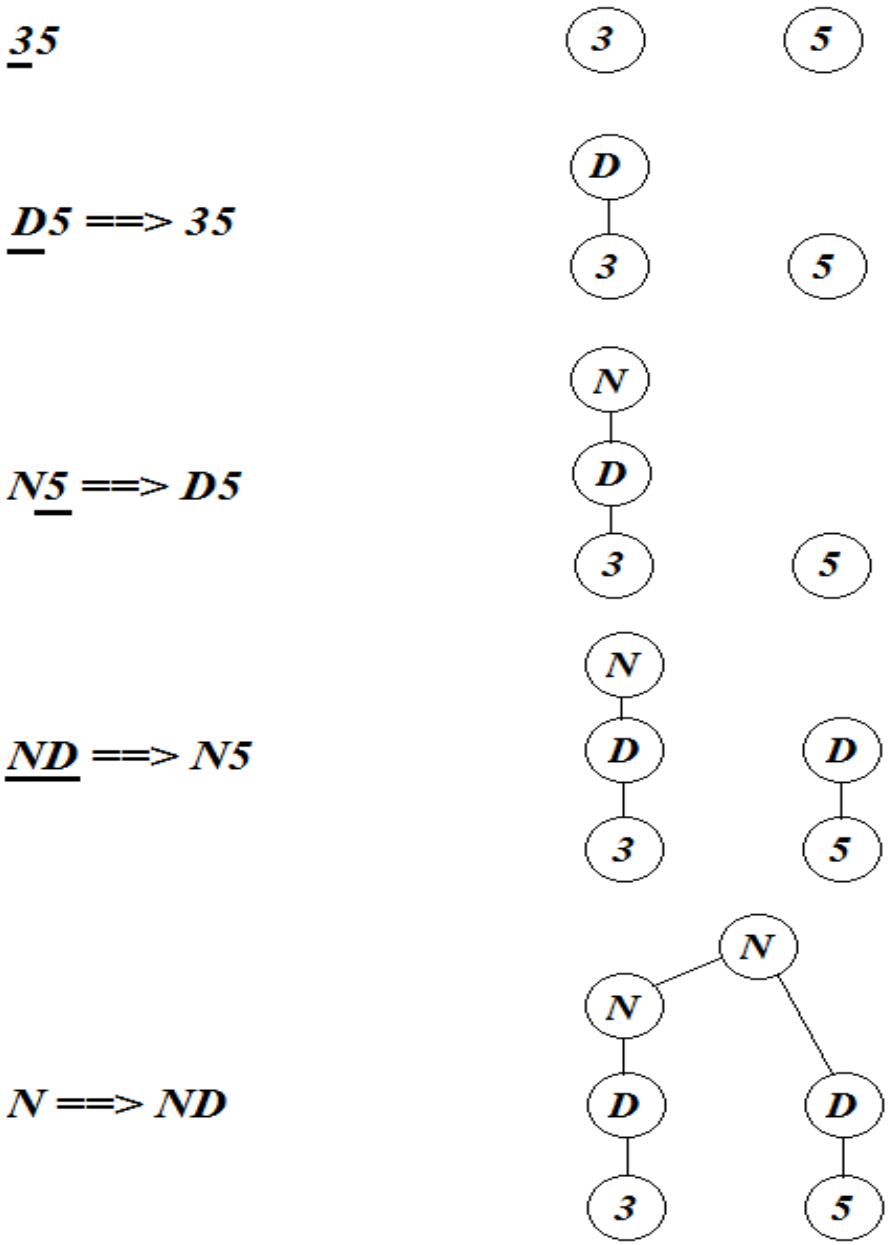
3D ==> 35



Восходящий грамматический разбор

-
-
- ***N => ND => N5 => D5 => 35***

Восходящий разбор



Проблемы грамматического разбора

-
-
- $X \rightarrow x_1 \mid x_2 \mid \ldots \mid x_n$
-

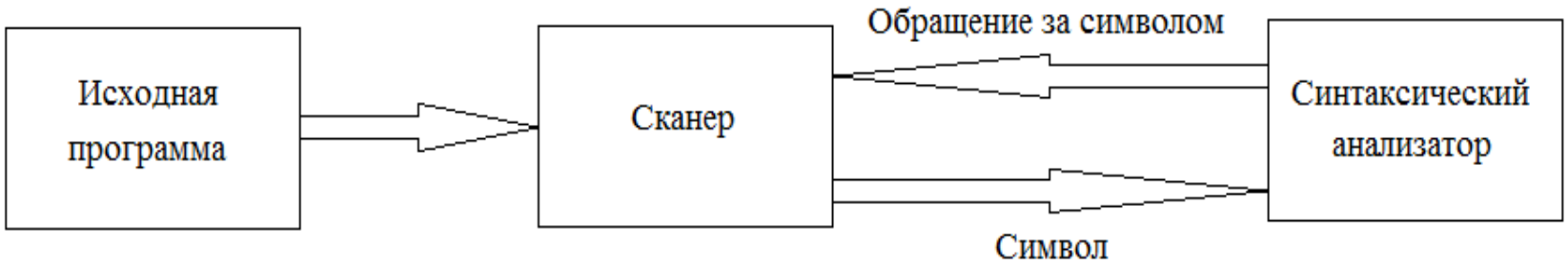
2.5. Другие способы представления синтаксиса

- *<список идент. > ::= <идент. > | <список идент.>, <идент >*
- *<список идент. > ::= <идент. >{ ,<идент. >}*
- *<усл. опер.> ::= if(<логич. выр.>) <оператор> [else <оператор>]*

3. ПОСТРОЕНИЕ ЛЕКСИЧЕСКОГО БЛОКА

-
-

Схема использования лексического блока



3.1. Грамматика для лексем

-
- идентификаторы;
- ключевые слова (являются подмножеством идентификаторов);
- целые числа;
- однолитерные разделители (+, - , * , …);
- двухлитерные разделители (// , /* , …).

Грамматика

-
- ***<идентиф. > ::= буква | <идентиф. >буква | <идентиф. >цифра***
- ***<целое> ::= цифра | <целое>цифра***
- ***<разделитель> ::= + | - | (|) ...***
- ***<разделитель> ::= <SLASH>/ | <SLASH>* | <COLON>= …***
- ***<SLASH> ::= /***
- ***<COLON> ::= :***

Регулярная грамматика

- Каждое правило имеет вид:
-
- $U \rightarrow a$
- или
- $U \rightarrow Ba$,

-

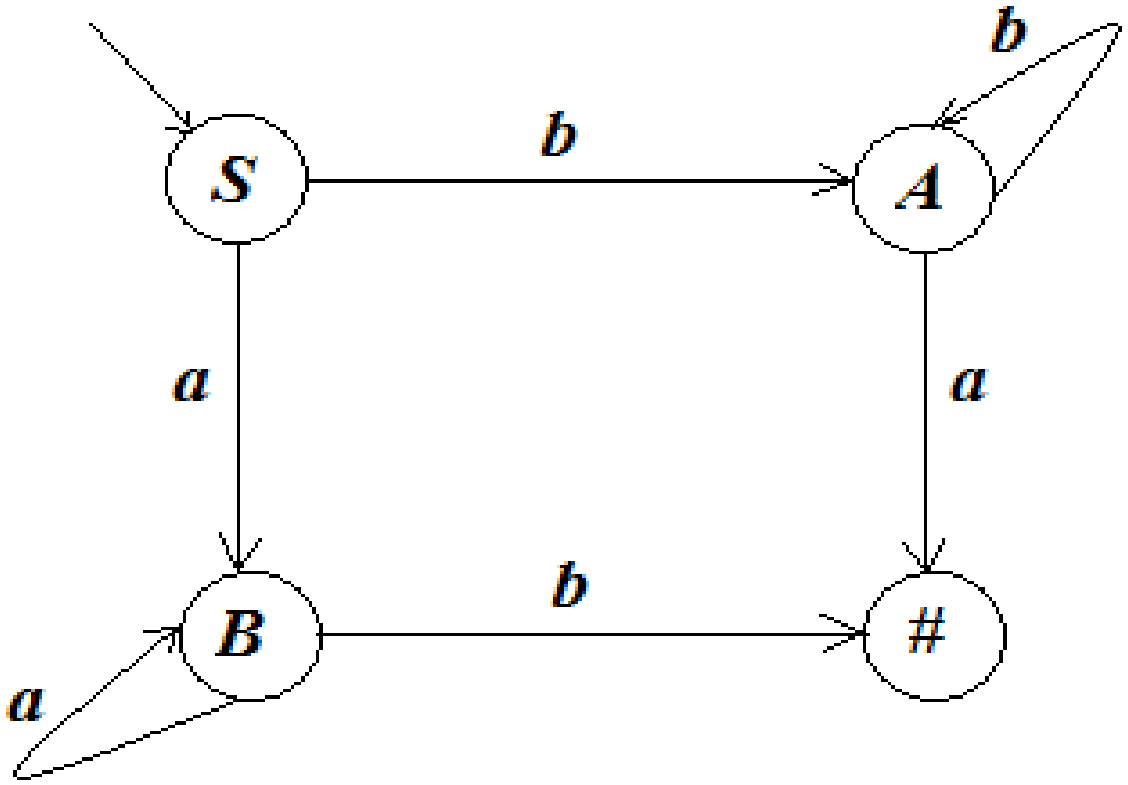
3.2. Регулярные грамматики и конечные автоматы

- Любая регулярная грамматика ***G = <N,T,P,S>*** может быть представлена ориентированным графом с помеченными узлами и дугами.
- Каждому элементу из множества ***N*** сопоставляется узел графа.
- Кроме того, вводится специальный “конечный узел”, помечаемый символом #.
- Вершины графа соединяются следующим образом: если в грамматике имеется правило ***A → Bα***, то из вершины ***A*** в вершину ***B*** проводится дуга, помеченная символом ***α***; если в грамматике имеется правило ***A → α***, то из вершины ***A*** в вершину ***#*** проводится дуга, помеченная символом ***α***.
- Таким образом, каждому правилу грамматики соответствует дуга в графе.

Пример

- Пусть дана грамматика ***G.***
-
- ***G = < {A,B,S}, {a,b}, P, S >***
- ***P:***
- ***S → Ab | Ba***
- ***A → Ab | a***
- ***B → Ba | b***
-
- Эта грамматика может быть представлена ориентированным графом:

Граф грамматики



Аналогия с конечным автоматом

- Рассмотрим теперь некоторый путь в графе, соединяющий вершину ***S*** с вершиной ***#***.
- Каждый такой путь соответствует выводу цепочки, например:
-
- ***S Þ Ab => Abb => abb***
- ***S => Ba => Baa => Baaa => baaa***
-
- Метки дуг пути, взятые в обратном порядке, образуют выведенную цепочку.
- Граф регулярной грамматики аналогичен графу переходов некоторого конечного автомата. Это не случайно – между регулярными грамматиками и конечными автоматами существует взаимно-однозначное соответствие. Конечный автомат допускает цепочки, порождаемые соответствующей ему регулярной грамматикой.

Конечный автомат

- Приведём определение конечного автомата.
- *Детерминированный конечный автомат* – это пятёрка
- ***KA = <K,T,M,S,Z>,*** где
- ***K*** – конечное множество состояний (конечный алфавит состояний),
- ***T*** – конечный входной алфавит,
- ***M*** – функция переходов,
- ***S*** – начальный символ грамматики (***S*** принадлежит ***K***),
- ***Z*** - непустое множество заключительных состояний (***Z*** **-** это
- подмножество ***K).***
- Со входной цепочкой ***t*** конечный автомат работает следующим образом:
- ***M(Q,ε) = Q*** при любом состоянии ***Q***,
- ***M(Q,at) = M(M(Q,a),t)*** где ***a*** принадлежит ***T, t*** принадлежит T^*.
-
- Говорят, что конечный автомат *допускает* (*принимает*) цепочку ***t*** (т.е. цепочка ***t*** считается *допустимой*), если ***M(S,t) = P***, где ***P*** принадлежит ***Z***.
- Рассмотренные автоматы называются детерминированными, поскольку на каждом шаге входной символ однозначно определяет следующее состояние автомата.
- Конечный автомат может задаваться матрицей переходов, списками переходов или графом переходов.