Parse Tree, Parser and Top-down Parsing

Parse Tree: [Section 2.2.3]

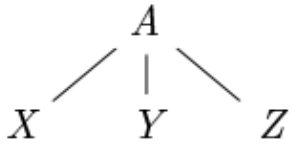
- A parse tree pictorially shows how the start symbol of a grammar derives a string in the language.
- From left to right, the leaves of a parse tree form the *yield* of the tree, which is the string generated or derived from the nonterminal at the root of the parse tree.
- Formally, given a context-free grammar, a parse tree according to the grammar is a tree with the following properties:
  - The root is labeled by the start symbol.
  - Each leaf is labeled by a terminal or by ε.
  - Each interior node is labeled by a nonterminal.
  - If A is the nonterminal labeling some interior node and X, Y, Z are the labels of the children of that node from left to right, then there must be a production A → XYZ. Here, X, Y, Z each stand for a symbol that is either a terminal or nonterminal. As a special case, if A → ε is a production, then a node labeled A may have a single child labeled ε.

So, an interior node and its children correspond to a production; the interior node corresponds to the head of the production, the children to the body.

A Production in CFG

$A \rightarrow XYZ$

An Interior Node of a Parse Tree

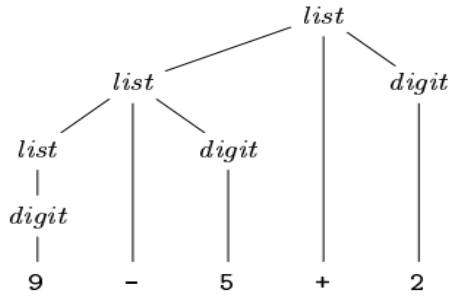


Example:

A given CFG, G:

list → *list* + *digit*
list → *list* – *digit*
list → *digit*
digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

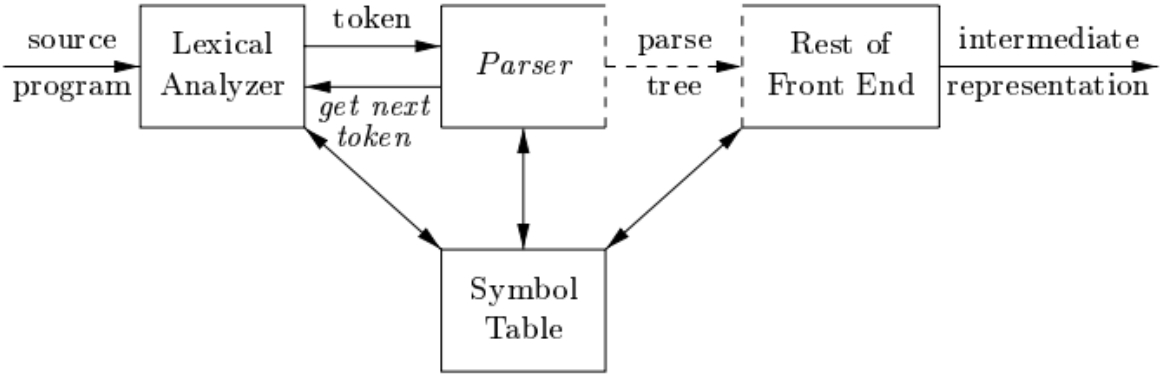
Parse tree of the string `9-5+2` according to G:



The process of finding a parse tree for a given string of terminals is called *parsing* that string.

Parser: [Section 4.1.1]

The parser obtains a string of tokens from the lexical analyzer and verifies that the string of token names can be generated by the grammar for the source language.



There are three general types of parser for grammars: *universal*, *top-down* and *bottom-up*. The methods commonly used in compilers can be classified as top-down or bottom-up.

- Universal parsing methods can parse any grammar. These general methods are, however, too inefficient to use in production compilers.
- Top-down methods build parse tree from the top (root) to the bottom (leaves).
- Bottom-up methods start from the leaves and work their way up to the root.

In both top-down and bottom-up methods, the input to the parser is scanned from left to right, one symbol at a time.

Top-down parsing: [Section 2.4.1]

The top-down construction of a parse tree is done by starting with the root and repeatedly performing the following two steps:

1. At node N, labeled with nonterminal A, select one of the productions for A and construct children at N for the symbols in the production body.
2. Find the next node at which a subtree is to be constructed, typically the leftmost unexpanded nonterminal of the tree.

In general, the selection of a production for a nonterminal may involve *trial-and-error*; that is, we may have to try a production and backtrack to try another production if the first is found to be unsuitable. A production is *unsuitable* if, after using the production, we cannot complete the tree to match the input string.

The current terminal being scanned in the input is frequently referred to as the *lookahead* symbol. Initially, the lookahead symbol is the first, i.e., leftmost, terminal of the input string.

$$
\begin{array}{rcl}
\mathit{stmt} & \rightarrow & \textbf{expr}\ ; \\
 & | & \textbf{if}\ (\ \textbf{expr}\ )\ \mathit{stmt} \\
 & | & \textbf{for}\ (\ \mathit{optexpr}\ ;\ \mathit{optexpr}\ ;\ \mathit{optexpr}\ )\ \mathit{stmt} \\
 & | & \textbf{other} \\
 & & \\
\mathit{optexpr} & \rightarrow & \epsilon \\
 & | & \textbf{expr}
\end{array}
$$

Fig: A grammar for some statements in C and Java

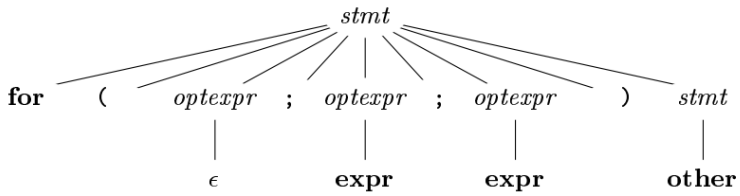


Fig: A parse tree according to the above grammar for input string **for (; expr ; expr) other**

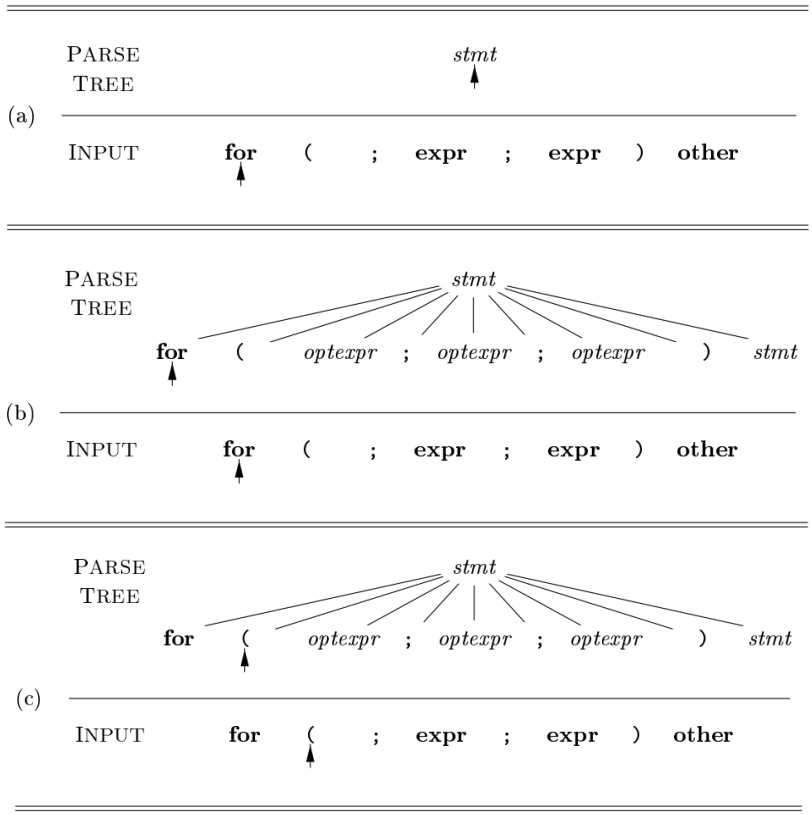


Fig: Top-down parsing while scanning the input from left to right

Predictive parsing: [Section 2.4.2]

- Recursive-descent parsing is a top-down method of syntax analysis in which a set of recursive procedures is used to process the input.
- One procedure is associated with each nonterminal of a grammar.
- Predictive parsing is a simple form of recursive-descent parsing in which the lookahead symbol unambiguously determines the flow of control through the procedure body for each nonterminal.
- Parsing begins with a call of the procedure for the starting nonterminal symbol.