Compiler Construction (List 6)

Hans de Nivelle

25.11.2015

This exercise is about bottom-up parsing, and about comparing bottom-up and top-down parsing. It is not a programming exercise.

1. Consider the following grammar for expressions:

$$
\begin{array}{lcl}
E & \to & E + F \\
E & \to & E - F \\
E & \to & F \\
 & & \\
F & \to & F \times G \\
F & \to & F \ / \ G \\
F & \to & G \\
 & & \\
G & \to & - \ G \\
G & \to & (\ E\ ) \\
G & \to & \text{identifier} \\
G & \to & \text{number} \\
G & \to & \text{identifier} \ (\ A\ ) \\
 & & \\
A & \to & E \\
A & \to & A\ ,\ E
\end{array}
$$

   Give reasonable attribute functions for this grammar. The functions can either produce parse trees, or evaluate the expressions immediately.

2. In order to make the language parsable by a top-down parser, the grammar has to be rewritten into a DFA-based grammar without shared prefixes and left recursion. Give such a modified grammar. Give the FIRST and the FOLLOW sets for the non-terminals. Also add an additional rule $S \to E\ \#$, where $\#$ is the EOF symbol.

3. In one uses a bottom-up parser, the grammar can be used without any changes.

   Produce a prefix automaton for this grammar. (The automaton should have 24 states.) There is no need to draw the automaton, but I want to see the item sets that it is based on. Give reasonable FOLLOW sets for the reductions.