Precedence Climbing

Matěj Chalk, xchalk00@stud.fit.vutbr.cz
Kateřina Pilátová, xpilat05@stud.fit.vutbr.cz

October 31, 2017

Recursive descent is a technique for top-down parsing of mathematical expressions. The idea of recursive-descent parsing is to transform each nonterminal of a context-free grammar into a subroutine that will recognize exactly that nonterminal in the input. The output of the parser may be in the form of an abstract syntax tree, reverse-polish notation, invocations of an analyser and code generator (for one-pass compilers), or a numerical result (as in a calculator). In all of these cases, however, the parser must generate output that follows the precedence and associativity of operators, while still remaining efficient when there are many levels of precedence (as in the C and C++ languages).

The classic solution to recursive-descent parsing of expressions is to create a new nonterminal for each level of precedence. However, this solution has two main drawbacks – the speed of the algorithm is proportional to the number of precedence levels, and the set of operators and their precedence and associativity are hard-coded.

Precedence climbing is a method that solves both of these problems. The main subroutine is parameterised by a minimal precedence level. It calls itself recursively, increasing the precedence level when a binary operator with higher (or equal, depending on associativity) precedence is encountered, or returning when the precedence is lower. This approach eliminates the problem of having to always iterate towards the highest precedence level one step at a time (as the classic solution must do), which means that adding more precedence levels does not make it less efficient. Furthermore, operators and their precedence levels are stored in a table, making the method more flexible.