Programming Language Design 2024
Syntax

Hans Hüttel

12 February 2024

1 Problems we will definitely talk about

1. Addition ($+$) and multiplication ($\times$) are associative in the mathematical sense, so $x + (y + z) = (x + y) + z$ and $x \times (y \times z) = (x \times y) \times z$. This allows us to leave out parentheses.

   Most programming languages allow expressions such as $x + y + x$ and $x * y * x$ without parentheses, but specify that the meaning of these are equivalent to $(x + y) + x$ and $(x * y)*$x, respectively, and not to $x + (y + x)$ or $x * (y * x)$.

   An alternative would be to let the grouping be unspecified (as in mathematics).

   What are the benefits and hazards of doing this in a programming language? Assume that there are no side effects.

2. Some LISP variants allow a single bracket to close all open (normal) parentheses. What are the benefits and hazards of allowing this? Describe how you would implement this.

3. Assuming the usual operator precedences, convert the following infix expression to prefix (polish) and postfix (reverse polish) notation. Keep the constants in the same order as below.

   $$(2 + 3 * (4 - 5)) * 6/7$$

   Next, write a program/function (in any language of your choosing) that inputs an infix expression and outputs the equivalent prefix and postfix expressions, while keeping the constants in the same order. The syntax of infix expressions is given by the formation rules

   $$E ::= n \mid (E) \mid E \; op \; e$$

   where *op* is one of the infix operators $+$, $-$, $*$ and / with the usual precedence and associativity. Assume that numbers n are one digit only. You can decide whether or not spaces are allowed. Use the expression given above as a test case.

2 More problems

a. When we define a function or procedure $f(x_1 \ldots x_n)$ with n formal parameters, we of course want to check that every invocation $f(a_1 \ldots a_m)$ satisfies that $m = n$. This should be a purely syntactic property. Where does it live? Can it be handled at the lexical level, at the grammatical level or not even there? What is the appropriate way of handling it?