Operator Precedence and Associativity

Here is an ambiguous context-free grammar similar to the ambiguous grammar for expressions discussed in class:

\[
\langle expr \rangle ::= \langle term \rangle \mid \langle expr \rangle + \langle expr \rangle \mid \langle expr \rangle \ast \langle expr \rangle
\]

\[
\langle term \rangle ::= x \mid y \mid z \mid (\langle expr \rangle)
\]

As an example, for \(x + y + z\) this grammar allows two different parse trees. In one tree the first + is “deeper” in the tree (i.e., the + operator is left-associative). The resulting assembly code produced by a compiler may look something like

\[
\text{ADD R1, x, y}
\]

\[
\text{ADD R2, R1, z}
\]

Here R1 and R2 are registers. The first instruction adds the values in variables x and y and stores the result in R1. The second instruction adds the values in R1 and z and stores the result in R2.

In the other possible parse tree, the second + is deeper (i.e., the + operator is right-associative). The assembly code may look something like

\[
\text{ADD R1, y, z}
\]

\[
\text{ADD R2, x, R1}
\]

As another example, for \(x + y \ast z\) this grammar allows two parse tree. For the tree where the + is deeper, the + operator has higher precedence than the \(\ast\) operator. For the tree where the \(\ast\) is deeper, the \(\ast\) operator has higher precedence than the + operator. Clearly, these different parse trees result in different computed values: \((v_1 + v_2)v_3\) vs \(v_1 + v_2v_3\), where \(v_1\) denotes the value stored in variable \(x\), etc.

In real languages, operator precedence and associativity is well-defined: for example, + and \(\ast\) are left-associative, and \(\ast\) has higher precedence. To remove the ambiguity and to achieve this precedence and associativity, the grammar can be modified as follows:

\[
\langle expr \rangle ::= \langle term \rangle \mid \langle expr \rangle + \langle term \rangle
\]

\[
\langle term \rangle ::= \langle factor \rangle \mid \langle term \rangle \ast \langle factor \rangle
\]

\[
\langle factor \rangle ::= x \mid y \mid z \mid (\langle expr \rangle)
\]