

Exercise 6: Evaluation of Prefix Expressions

Arithmetic expressions can be written in prefix notation, where the operator *precedes* the operands (e.g. + a b). Here are some examples of prefix expressions and their meaning in the usual infix notation:

+ 3 * 6 2 = 3 + (6 * 2)
* + 1 2 3 = (1 + 2) * 3
+ * 2 4 * 6 8 = (2 * 4) + (6 * 8)

- a) Write a context-free grammar for prefix expressions. A prefix expression is either a number (terminal symbol number) or begins with the operator "+" or "*", followed by two prefix expressions (use recursion). The precedence rules between "+" and "*" can be ignored because they are implicit in the prefix notation.
- b) Turn your grammar into an attributed grammar so that the value of a prefix expression is returned as an output attribute. The value of number can be taken from t.numVal.

Solution

- a) Context-free Grammar

```
Expr
= number
| "+" Expr Expr
| "*" Expr Expr .
```

- b) Attributed Grammar

```
Expr <↑x>        (. int x, y; .)
= number        (. x = t.numVal; .)

| "+"
  Expr <↑x>
  Expr <↑y>        (. x += y; .)

| "*"
  Expr <↑x>
  Expr <↑y>        (. x *= y; .) .
```