apl. Prof. Dr. Thomas Noll
Philipp Berger, Matthias Volk

Compiler Construction 2018/19

— Exercise Sheet 8 —

Hand in until December 10th before the exercise class.

Exercise 1 (4 Points)

In this exercise we apply attribute grammars to evaluate simple arithmetic expressions.

(a) Write an *unambiguous* context-free grammar for arithmetic expressions that contains arbitrary numerical values (which you may represent by a single terminal symbol `num`), addition (`+`), multiplication (`*`), and parenthesis (`(`, `)`).
Hint: You can already incorporate the usual operator precedence (`(`, `)` $>$ `*` $>$ `+`) into the grammar.

(b) Define an attribute *value* to compute the value of an expression. That is, the value of a given expression should correspond to the value of attribute *value* at the root of the corresponding derivation tree. Make sure to follow the operator precedence as given in (a).

(c) Evaluate `(1+3*2)*6`. To this end, first construct the corresponding derivation tree, then set up the equation system, and solve it.

Exercise 2 (6 Points)

Give a context-free grammar for the language $\mathcal{L}$. Extend that grammar with attributes so that the language is the following set:

(a) $L = \left\{a^{(3^n+1)} \mid n \in \mathbb{N}\right\}$ $(\mathcal{L} := \{a\}^+)$.

(b) $L = \left\{a^i b^j c^k \mid 0 < i < j < k\right\}$ $(\mathcal{L} := \{a\}^+ \{b\}^+ \{c\}^+)$.

You may use any number of attributes, conditional updates, simple arithmetic and comparison between numbers. However, you may not use a predicate that directly checks whether a given number (in decimal or binary encoding) is a power of three or not.