

Exercise 1 (Evaluation of Arithmetic Expression):

(3 Points)

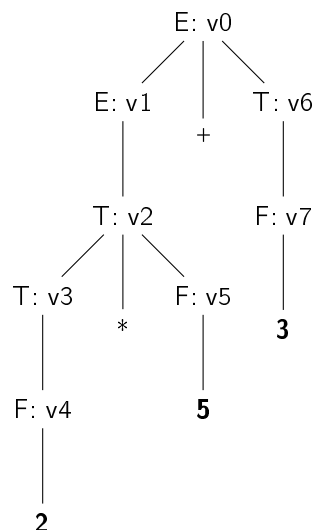
Write an unambiguous grammar for arithmetic expressions (containing addition, multiplication and parenthesis). Define an attribute *Val* to find the value of an expression. Evaluate $2 * 5 + 3$. For this, give the parse tree of this expression, set up the corresponding equation system and solve it.

Lösung:

The attribute grammar for the arithmetic expression is as follows:

$$\begin{aligned} E &\rightarrow E + T & Val.0 &= Val.1 + Val.3 \\ E &\rightarrow T & Val.0 &= Val.1 \\ T &\rightarrow T * F & Val.0 &= Val.1 * Val.3 \\ T &\rightarrow F & Val.0 &= Val.1 \\ F &\rightarrow (E) & Val.0 &= Val.2 \\ F &\rightarrow \text{num} & Val.0 &= \text{num} \end{aligned}$$

The annotated parse tree for $2 * 5 + 3$ as follows:



The set of variables is $\{v0, \dots, v7\}$. The equation system as follows:

$$\begin{aligned} v0 &= v1 + v6 \\ v1 &= v2 \\ v2 &= v3 * v5 \\ v3 &= v4 \\ v4 &= 2 \\ v5 &= 5 \\ v6 &= v7 \\ v7 &= 3 \end{aligned}$$

Exercise 2 (Attributed Grammars):

(4 Points)

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

1. $L = \{a^{2^n} | n \in \mathbb{N}\}$.
2. $L = \{a^{n^2} | n > 0\}$.

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 two or not.

Lösung:

Consider the grammar G :

$$\begin{aligned} S &\rightarrow a S \\ S &\rightarrow a \end{aligned}$$

The language of G is $\{a\}^+$. We have the following synthesized attributes $\{c, f, b\}$. The semantic rules for each production is given as follows:

1.

$$\begin{aligned} S &\rightarrow a S \quad \text{if } f.2 < c.2 - 1 \text{ then } f.0 = f.2 + 1; c.0 = c.2; b.0 = \text{false}; \\ &\quad \text{else } f.0 = 0; c.0 = c.2 + f.2 + 1; b.0 = \text{true} \\ S &\rightarrow a \quad f.0 = 0; c.0 = 1; b.0 = \text{true}; \end{aligned}$$
2.

$$\begin{aligned} S &\rightarrow a S \quad \text{if } c.2 = f.2 * f.2 - 1 \text{ then } f.0 = f.2 + 1; c.0 = c.2 + 1; b.0 = \text{true}; \\ &\quad \text{else } c.0 = c.2 + 1; f.0 = f.2; b.0 = \text{false} \\ S &\rightarrow a \quad f.0 = 2; c.0 = 1; b.0 = \text{true}; \end{aligned}$$

Exercise 3 (Circularity Test):

(3 Points)

In this task we implement a semantic check. In our language *WHILE* we require that every variable identifier is *declared* before the variable is used (read or set). Additionally, a variable defined inside a scope like an if statement or a while loop is not visible outside this scope. We do not care whether a variable has been *initialised* before it is read. Examples:

This is valid:

```
1  int x; int y;
2  if (x <= y) {
3      write("...");
4  }
5  write(x);
6  $
```

This is not valid (y is undefined and z is undefined outside the if-statement):

```
1  int x;
2  if (x <= y) {
3      int z;
4      // ...
5  }
6  write(z);
7  $
```

Implement `Checker.checkDeclaredBeforeUsed()`. Hint: for this you do not need to implement any attributed grammars and their evaluation. Instead simply walking the abstract syntax tree once and checking the required property suffices.

Lösung: _____

_____.