Functional Programming – Series 6b

In this series of exercises we will write a *type checker* for expressions.

Exercise 1. Given are embedded languages for *expressions* and for *types*:

$$\begin{aligned}&\textbf{data } \mathit{Expr} = \mathit{Const}\ \mathit{Int}\\ &\qquad\qquad | \ \mathit{Var}\ \mathit{String}\\ &\qquad\qquad | \ \mathit{BinOp}\ \mathit{String}\ \mathit{Expr}\ \mathit{Expr}\\ &\qquad\qquad | \ \mathit{App}\ \mathit{Expr}\ \mathit{Expr}\end{aligned}$$

$$\begin{aligned}&\textbf{data } \mathit{Type} = \mathit{IntType}\\ &\qquad\qquad | \ \mathit{FunType}\ \mathit{Type}\ \mathit{Type}\end{aligned}$$

Define a function

$$\mathit{typeOf} :: \mathit{Env} \to \mathit{Expr} \to \mathit{Type}$$

Here, the *environment Env* contains the types of elementary variables and functions,denoted by *String*s. It is defined as follows:

$$\textbf{type } \mathit{Env} = [(\mathit{String}, \mathit{Type})]$$

The environment should contain type for at least the operations $+$, $*$, $-$, where these operations are indicated as strings "+", etcetera.

Exercise 2. Extend the embedded languages above with an expression for *if-the-else*, and adapt the types where approprite. The environment now should contain operations && and || for booleans as well.

Exercise 3. Extend the expressions and types with 2-tuples and 3-tuples.

Exercise 4. Add lambda expressions to your language, where a lambda expression contains the types of the formal parameters, in the following form:

$$\backslash x :: a \to \mathit{expr}$$

where a is a type.

Remark. Note that above no *type variables* are requested. That would require *substitution* and *unification*, which is at this moment not requested.