COMS22201: Language Engineering

Lab Exercises - Week 16 - Questions

15/02/2016, csxor@bristol.ac.uk

In this worksheet you will give a Haskell implementation of the semantics of arithmetic expressions of **While** by defining the semantic function from the last lecture with the following signature:

$$\mathcal{A}[\![.]\!] : Aexp \to (State \to Z)$$

For convenience, we will follow as closely as possible the Miranda implementation given in Appendix B of Neilson's book (making only those changes that are necessary to ensure the code correctly ports to Haskell).

1. Consider the following encoding of arithmetic expressions into Haskell data types (where the two imports avoid an unfortunate name-space conflict between our `Num`-eral type and Haskell's `Num`-eric class).

```
import Prelude hiding (Num)
import qualified Prelude (Num)

type Num = Integer
type Var = String

data Aexp = N Num | V Var
          | Add Aexp Aexp | Mult Aexp Aexp | Sub Aexp Aexp
          deriving (Show, Eq, Read)
```

   (a) Define a Haskell term `a::Aexp` representing the arithmetic expression $(x + y) * (z - 1)$.

   (b) Define a Haskell type `Z` as another synonym (just like `Num`) for Haskell's native `Integer` type.

(c) Define a Haskell function `n_val::Num->Z` to represent the semantic function $\mathcal{N}[\![.]\!] : Num \to Z$ for numerals (which is trivial here as we are using `Integer` for both syntax and semantics).

(d) Define a Haskell type `State` as the set of all possible functions from `Var` to `Z`.

(e) Define a Haskell term `s::State` representing the state $s_{x=1\ y=2\ z=3}$ (where all variables other than x, y and z are initialised to zero).

(f) Define a Haskell function `a_val::Aexp->State->Z` to represent the semantic function $\mathcal{A}[\![.]\!] : Aexp \to State \to Z$ for arithmetics.

(g) Use your code to evaluate the value of the arithmetic expression `a` in the state `s`.

(h) Modify your code to also support a unary negation operator and explain why it would *not* be acceptable to implement a semantic expression like $\mathcal{A}[\![-a]\!]\ s = \mathcal{A}[\![0 - a]\!]\ s$.

2. Write a concrete grammar for the arithmetic expressions of **While** that is unambiguous.

3. Prove that $\mathcal{A}[\![.]\!]$ is a total function.