COMS22201: Language Engineering

Lab Exercises - Week 17 - Questions

22/02/2016, csxor@bristol.ac.uk

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

$$\mathcal{B}[\![.]\!] : Bexp \to (State \to T)$$

Once again, 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. Add the following encoding of Boolean expressions in Haskell to the code you developed in the previous lab sheet for arithmetic expressions:

```
data Bexp = TRUE | FALSE
          | Eq Aexp Aexp | Le Aexp Aexp
          | Neg Bexp | And Bexp Bexp
```

   (a) Define a Haskell term `b::Bexp` representing the arithmetic expression $\neg((x + y) = 4)$.

   (b) Define a Haskell type `T` as a synonym for Haskell's `Bool` type.

   (c) Define a Haskell function `b_val::Bexp->State->T` to represent the semantic function $\mathcal{B}[\![.]\!] : Bexp \to (State \to T)$ for Booleans.

   (d) Use your code to evaluate the value of the Boolean expression `b` in the state $\mathtt{s}_{x=1\ y=2\ z=3}$.

2. Prove that $\mathcal{B}[\![.]\!]$ is a total function.

3. Prove $\mathcal{A}[\![a]\!]s = \mathcal{A}[\![a]\!]s'$ for all states s and s' such that $sx = s'x$ for all free variables x in a.

4. Prove $\mathcal{B}[\![b]\!]s = \mathcal{B}[\![b]\!]s'$ for all states s and s' such that $sx = s'x$ for all free variables x in b.

5. Prove the so-called *substitution lemma* for arithmetic expressions, which states that $\mathcal{A}[\![\, a[y \mapsto a']\,]\!]s = \mathcal{A}[\![a]\!](s[y \mapsto \mathcal{A}[\![a']\!]s])$ for all a, a', y, s.

6. Now consider an extended language **While'** which supplements **While** with a standard implication connective '$\rightarrow$'.

   (a) Modify your Haskell code to support this connective (ensuring that your semantic function is compositionally defined).

   (b) Write an unambiguous concrete grammar for the Boolean expressions of **While'** that respects the usual precedence and associativity conventions (where implication binds loosely and associates to the right).

   (c) Prove that for every Boolean expression of **While'** there is an equivalent Boolean expression in **While**.