CPSC 521: midterm exam

Robin Cockett

November 2009

This exam is worth 20% of the course. There are 100 points available.

1. (25 points)

   Consider the following Haskell code:

```
data Exp f v = Var v | Opn f [Exp f v]
    deriving Show

instance Monad (Exp f) where
    return x = Var x
    Var x >>= f = f x
    (Opn opn args) >>= f = Opn opn (map (\e -> e>>=f) args)

sst::Eq v => (v,Exp f v) -> (Exp f v) -> (Exp f v)
sst (v,exp1) exp2 =
   do w <- exp2
      if w==v then exp1 else (return w)
```

   (i) Explain how this code implements a substitution!

   (ii) Translate the above "do" syntax into "core" Haskell explaining the steps.

   (iii) Write the fold(right) for lists giving its type.

   (iv) Write a substitution function for a sequence of substitutions:

   `substitute::Eq v => [ (v,Exp f v)] -> (Exp f v) -> (Exp f v)`

   such that

$$\begin{aligned}\texttt{substitute}\ []\ t &= t\\ \texttt{substitute}\ [t_1/x_1, t_2/x_2, ...]\ t &= (\texttt{substitute}\ [t_2/x_2, ...]\ t)[t_1/x_1]\end{aligned}$$

2. (20 points)

   Demonstrate leftmost outermost reduction on the following λ-terms:

   (a) $(\lambda zx.x(zx))x(\lambda y.yx)$

   (b) $(\lambda xy.y(xx))(\lambda x.y)(\lambda x.xx)(\lambda x.xx)$

   (c) $(\lambda yx.x)((\lambda x.xx)(\lambda x.xx))(\lambda xy.x)$

   What are the advantages and disadvantages of this reduction strategy? What is by-value reduction? What is lazy reduction?

3. (20 points)

   (i) Explain how conditional statements

   $$\textsf{if } e \textsf{ then } t_1 \textsf{ else } t_2$$

   are programmed in the λ-calculus.

   (ii) Explain what a fixed point combinator is. Prove that

   $$\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

   is a fixed point combinator.

   (iii) Explain how to program the `gcd` function using fixed points:

   $$\begin{aligned} \texttt{gcd}(n,m) &= \texttt{if } n < m \texttt{ then } \texttt{gcd}(m-n,n) \\ & \texttt{elseif } m < n \texttt{ then } \texttt{gcd}(n-m,m) \\ & \texttt{else } n \end{aligned}$$

   You may assume that you already have basic arithmetic functions $n < m$,$n - m$ defined.

4. (15 points)

   (i) How do you represent the λ-calculus in the λ-calculus? (Hint: give a Haskell data definition for λ-terms – on an arbitrary type of variables – and translate it).

   (ii) Denote the representation of a λ-term (with natural numbers as variables) by $\underline{N}$: explain how to write a function H such that $\mathsf{H}\underline{N} = \underline{\underline{N}}$.

   (iii) Which of the following are true:

      (a) For every λ-term M there is a λ-term N such that $MN = N$;

      (b) For every λ-term M there is a λ-term N such that $M\underline{N} = N$;

      (c) For every λ-term M there is a λ-term N such that $M\underline{\underline{N}} = \underline{N}$.

5. (20 points)

   (i) Explain what it means to say that β-reduction is confluent.

   (ii) When is a λ-term in normal form? Why are two normal form λ-terms which are not α-equivalent not (β-)equal?

   (iii) How do you represent the natural numbers in the λ-calculus? Why, in this representation, are all the numbers distinct?

   (iv) Explain what is wrong with the reasoning which says "to tell whether two λ-terms are equal simply reduce them until they become the same."

   (v) (5 point bonus!)

   A λ-term N is said to be *hopelessly cyclic* if every β-reduction sequence eventually revisits N (for example Ω is hopelessly cyclic). A term is said to be *never hopelessly cyclic* if it never reduces to a hopelessly cyclic term.

   Give an argument to show that one cannot decide whether a term is never hopelessly cyclic