CPSC 521: Fall 2016 midterm exam

Robin Cockett

November 8, 2016

This exam is worth 20% of the course. There are four questions and a total of 100 points available:

1. (30 points) Explain your answers!

   (a) (4 points) Explain what it means for two λ-terms to be α-equivalent and how to use de Bruijn notation to decide whether two terms are α-equivalent. Demonstrate the technique on the following two terms:

   $$(\lambda xx.x)(\lambda x.xx) \quad \text{and} \quad (\lambda zy.z)(\lambda v.vv)$$

   (b) (3 points) What does it mean for two λ-terms to be β-equivalent? Is it possible to decide whether two terms are β-equivalent?

   (c) (4 points) When is a term in β-normal form? Provide an example of a term which has a β-normal form and yet may be β-reduced an arbitrary many times.

   (d) (4 points) Explain why each λ-term is β-equivalent to *at most* one β-normal form.

   (e) (12 points) Demonstrate leftmost outermost β-reductions on the following λ-terms:

      (i) $(\lambda zx.z(xz))(\lambda z.zx)(\lambda x.xx)$

      (ii) $(\lambda xy.xyx)(\lambda xz.z(yx))yy$

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

   (f) (3 points) Give an advantage and a disadvantage of a leftmost outermost reduction strategy over a by-value (rightmost innermost) reduction strategy.

2. (35 points) Consider the following Haskell code:

```
data Exp f v = Var v | App f [Exp f v]

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

subst::Eq v => (v,Exp f v) -> (Exp f v) -> (Exp f v)
subst (x,t) s = do y <- s
                   if x==y then t else return y

substs:: Eq v => [(v,Exp f v)]  -> (Exp f v) -> (Exp f v)
substs subs t = foldlist (\sub s -> subst sub s) t subs
```

(a) (5 points) Explain how this code implements substitution.

(b) (6 points) Write the fold(right) for lists required for `foldlist` in the above code.

(c) (12 points) Translate the above "do" syntax for `subst` into "core" Haskell explaining the steps.

(d) (12 points) Write the fold function for `Exp f v` and show how you can use it to collect the free variables of an expression.

3. (30 points)

(a) (12 points) How do you represent the datatype of trees

```
data Tree f v = Leaf v
              | Node f (Tree f v) (Tree f v)
```

in the λ-calculus?

What are the λ-terms for the constructors, the fold, and the case combinator for trees?

(b) (7 points) Explain what a fixed point combinator is. Prove that

$$\mathbf{Y} := \Theta\Theta \quad \text{where } \Theta := \lambda xf.f(xxf)$$

is a fixed point combinator.

(c) (7 points) Explain how the recursive `factorial` function

```
factorial n = if (iszero n) then (succ zero)
              else n * (factorial (pred n))
```

is programmed in the λ-calculus (you may assume the if combinator and the arithmetic functions).

(d) (4 points) Is it possible to decide whether a given λ-term normalizes to $\mathsf{True} := \lambda xy.x$? Explain your answer.

4. (5 points!)

   (a) What was Curry's first name? Who was Curry's (formal) thesis supervisor?

   (b) What is the Turing award? Name two Turing award recipients.

   (c) What did Turing do during World War II? Where did he do this?

   (d) When/how did Turing die? At which University was he working at the time?

   (e) The sentence: "If this sentence is true then Calgary is smaller than Edmonton." is an example of Curry's paradox. Explain why it is a paradox. What does it have to do with the λ-calculus?