CPSC 521: midterm exam

Robin Cockett

November 6, 2014

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

1. (30 points) Explain your answers!

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

   (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) (3 points) Write
   $$\lambda xy.(\lambda y.xy)(\lambda x.xy)$$
   in de Bruijn notation.

   (d) (3 points) When is a term in β-normal form? Is the term in (c), above, in β-normal form? Provide two examples of terms which do not have a β-normal form.

   (e) (3 points) Explain what it means to say that β-reduction is confluent. Why does each λ-term have *at most* one normal form?

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

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

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

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

   (g) (3 points) What is an advantage of a leftmost outermost reduction strategy over a by-value reduction strategy?

2. (15 points) Consider the "printer" monad as defined by:

```
data Printer a s = Print [a] s

instance Monad (Printer a) where
     return s = Print [] s
     (Print as s) >>= f = case f s of Print bs s' -> Print (as++bs) s'

pput str = Print str ()
```

and the data for arithmetic functions:

```
data AExp a = Add (AExp a) (AExp a)
            | Mul (AExp a) (AExp a)
            | Num a
```

Recall that the translation from "do" syntax to core Haskell is given by:

$$\begin{array}{rcl} [\![\text{do } \{e\}]\!] & = & e \\ [\![\text{do } \{x \leftarrow t; r\}]\!] & = & t \;\; >>= q \;\; \text{where} \\ & & \qquad q\ x = [\![\text{do } \{r\}]\!] \\ [\![\text{do } \{p; r\}]\!] & = & p >>= \lambda\_.[\![\text{do } \{r\}]\!] \end{array}$$

Consider the following function:

```
aprint :: Aexp Int -> Printer Char Int
aprint (Num n) = do pput (show n)
                    return n
aprint (Add t1 t2) = do pput "("
                        n1 <- aprint t1
                        pput "+"
                        n2 <- aprint t2
                        pput ")"
                        return (n1+n2)
aprint (Mul t1 t2) = do pput "("
                        n1 <- aprint t1
                        pput "*"
                        n2 <- aprint t2
                        pput ")"
                        return (n1*n2)
```

Explain, briefly, what this function does. Translate the first two cases of the function into core Haskell *leaving the sequencing operator untouched*. On the first of these, demonstrate carefully how to remove the sequencing and return operators.

3. (15 points) Using the datatype `AExp a` of the previous question:

   (a) (5 points) Define the fold function in Haskell for arithmetic expressions.

   (b) (10 points) Rewrite the function `aprint` of the previous question as a fold over the tree of arithmetic expressions to have type:

   ```
   aprint':: Aexp Int -> ([Char], Int)
   ```

4. (35 points)

   (a) (5 points) Explain how "triples" are represented in the lambda calculus. What are the definitions of the *three* projection functions?

   (b) (10 points) How do you represent the trees of

   ```
   data Tree a b = Leaf a
                 | Node  b (Tree a) (Tree a)
   ```

   in the λ-calculus?

   What are the λ-terms for the constructors, the fold, the map function (this takes in two functions), and the case combinator for trees?

   (c) (5 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.

   (d) (10 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).

   (e) (5 points) Explain briefly why all computable functions can be represented in the λ-calculus.

5. (5 points!)

   (a) What was Turing's first name?

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

   (c) Can you name a graduate student of Alonso Church? Is Church still alive?

   (d) Can you name a graduate student of Haskell Curry? Is Curry still alive?

   (e) The sentence: "If this sentence is true then the temperature today in Calgary is 35C." is an example of Curry's paradox. Explain why it is a paradox. What does it have to do with the λ-calculus?