CPSC 521: midterm exam

Robin Cockett

March 9, 2020

This exam is worth 20% of the course. There are four questions worth 100 points and a bonus question for 5 points:

1. (30 points) Explain your answers!

   (a) (3 points) Write

   $$\lambda xy.(\lambda y.x(\lambda x.yx))(\lambda x.xy)$$

   in de Bruijn notation.

   (b) (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?

   (c) (5 points) What does it mean for two λ-terms to be β-equivalent?

   - Explain what it means to say that β-reduction is confluent.
   - Explain why two λ-terms are β-equivalent if and only if there is a λ-term to which they can both be β-reduced.

   (d) (7 points) When is a λ-term in normal form?

   - Give two examples of λ-terms which *are* in normal form.
   - Give two examples of terms which *are not* in normal form and, furthermore, have no normal form.
   - Explain why each λ-term has *at most* one normal form?

   (e) (12 points) Demonstrate leftmost outermost β-reduction on the following terms to reduce them to a normal form:

   (i) $(\lambda yz.y(zy))(\lambda x.xz)(\lambda z.zx)$

   (ii) $(\lambda xy.xx(yy))(\lambda xy.x)(\lambda x.xx)(\lambda x.xx)$

   (iii) $(\lambda yx.x(xy))(\lambda xy.yx)(\lambda nxy.y(nxy))$

2. (25 points) Consider the data type of λ-terms:

```
data Lambda a = App (Lambda a) (Lambda a)
              | Abst a (Lambda a)
              | Var a
```

   (a) (5 points) Define the fold function in Haskell for λ-terms:

```
foldLambda:: (c -> c -> c) -> (a -> c -> c) -> (a -> c)
                           -> (Lambda a) -> c
```

   (b) (10 points) Write a Haskell function `free: Eq a => Lambda a -> [a]` which returns the list of free variables of the λ-term. (Important: here should be no repeated variables in the list of free variables!)

   (c) (10 points) How do you represent the λ-terms in the λ-calculus? What are the constructors `App`, `Abst`, and `Var`?

3. (35 points)

   (a) (10 points) How are the natural numbers represented in the λ calculus? (Hint the Church numerals). What are the constructors `Zero` and `Succ`?

   (b) (5 points) How are "pairs", (t_1, t_2), represented in the λ-calculus? How are the projections π_0 and π_1 defined?

   (c) (5 points) How is the case operator defined on the Church numerals?

```
case n of
     Zero -> t
     Succ m -> s
```

   (d) (5 points) Explain what a fixed point combinator is. Prove that

$$\mathbf{Y} := \Delta\Gamma \quad \text{where} \quad \Delta := (\lambda xy.xyx) \quad \text{and} \quad \Gamma := (\lambda yx.y(xyx))$$

   is a fixed point combinator.

   (e) (10 points) Explain how the recursively defined `add` function

```
add n m = case n of
               Zero -> m
               Succ m -> Succ (add n m)
```

   is programmed in the λ-calculus using a fixed point combinator, Y. You may assume the `case` combinator for the natural numbers.

4. (10 points) Explain briefly why all the partial computable functions can be represented in the λ-calculus.

5. (5 points bonus!)

   (a) What was Turing's middle name?

   (b) When did Turing die? How did he die?

   (c) Was Kleene a student of Alonso Church?

   (d) Was Haskell Curry a student of Alonso Church?

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