CPSC 417: exercises for the midterm

Robin Cockett

October 29, 2007

The exam will use questions from this sheet. If a question is labeled **hard** or **very hard** it will not be on the exam. BUT a question labeled **harder** might be!

Haskell questions:

1. Describe the translation from list comprehension syntax to core Haskell code. Translate the following function:

   ```
    pairs:: [a] -> [b] -> [(a,b)]
    pairs xs ys = [(x,y)| x <- xs, y <- ys, not x==y]
   ```

   Rewrite this function using the "do" syntax.

2. Describe the translation from the "do" syntax to core Haskell code. Translate the function `mapSM` below and explain its purpose:

   ```
   module StateMonad where

    data SM s a = SM (s -> (a,s))

    instance Monad (SM s) where
       return k = SM (\s -> (k,s))
       (SM f) >>= g = SM (\s -> (\(a ,s') -> runSM (g a) s')(f s))

    runSM::(SM s a) -> s -> (a,s)
    runSM (SM f) = f

    mapSM:: (a -> (SM s b)) -> [a] -> (SM s [b])
    mapSM f [] = return []
    mapSM f (t:ts) = do t' <- f t
                        ts' <- mapSM f ts
                        return (t':ts')
   ```

3. Given the following representation of λ-terms:

```
data  LTree = Var Int
            | Abs Int LTree
            | App LTree LTree
```

(a) Write the Haskell code to determine the list of free variables of a λ-term so represented.

```
freeVars:: LTree -> [Int]
```

(b) Write the Haskell code to perform a substitution of a (free) variable in a λ-term.

```
subst:: LTree -> (Int,LTree) -> LTree
```

(c) Write the Haskell code to determine whether two terms are α-equivalent.

```
alphaEquiv:: LTree -> LTree -> Bool
```

4. What is the fold function for the following datatype:

```
data Rose a b = Rose [(a,Rose a b)]
              | RVar b
```

Use this fold function to define:

(a) A function which collects the variables of such a rose tree into a list (which is a left to right traversal of the variables).

```
varleaves :: Rose a b -> [b]
```

(b) Supposing that the nodes are numbered by intergers, which finds the least integer in the tree:

```
leastnode :: Rose Int b -> Int
```

5. What is the map function for `Rose a b`?

The computational interpretation of the λ-calculus

1. Explain how one can represent pairs in the λ-calculus. What are the fst and snd functions demonstrate that they do work to produce the required components from a pair.

2. The Boolean values True and False are usually represented as True $= \lambda xy.x$ and False $= \lambda xy.y$. How do you program an if ... then ... else statement in the λ-calculus?

   Usually Booleans are introduced as a datatype

   ```
   data Bool = True | False
   ```

   is the representation of this datatype different?

3. Explain how to represent the natural numbers in the λ-calculus.

4. Explain how the predeccessor function is defined on these numbers not using fixed points.

5. Explain what the primitive recursive functions on the natural numbers are and show how, without using fixed points, one can represent all the primitive recursive functions in the λ-calculus.

6. (Hard) Explain the significance of the Ackermann function and show how it can be represented without using fixed points. Conclude that even without fixed points one can represent more than just primitive recursive functions.

7. Explain what a fixed point combinator is. Give three examples. Prove that each is a fixed point combinator.

8. Explain how to program the `monus` function using fixed points.

9. Explain how to program the `factorial` function using fixed points.

10. (Harder) Show how one can program the following using a fixed point combinator (mutual recursion!):

```
F [] = 0
F(n:ns) = n+ G ns
G [] = 0
G(n:ns) = n - F ns
```

    You may assume you have the addition and subtraction, and a function to determine whether a list is empty.

11. In the λ-calculus how do you represent the following binary trees:

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

12. Write the associated map and fold function for the above datatype.

13. (Harder) Write the case function for the above datatype.

14. In the λ-calculus how do you represent the following binary trees:

```
data Rose a b = RVar a | Rose [(a,Rose a b)].
```

15. Write the associated map and fold function for the above datatype.

16. (Harder) Write the case function for the above datatype.

λ-Calculus theory

1. What does α-equivalence mean? Which of the following terms are α-equivalent

$$\lambda xy.y(\lambda z.xz) \quad \lambda yx.y(\lambda z.xz) \quad \lambda zx.x(\lambda y.zy) \quad \lambda ab.b(\lambda a.ab)$$

2. What is a rewrite rule? Give some examples.

3. What is a redex? What is a contractum?
4. When is a λ-term in β-normal form?
5. What does confluence mean?
6. What does local confluence mean?
7. (Harder) what is a critical pair?
8. Does local confluence imply (global) conflence? Give counter-examples.
9. When is a rewrite system terminating? Prove Newman's lemma: that a terminating rewrite system is confluent.
10. (Harder) Prove in detail that the rewiting system with the rules

$$\begin{aligned} (x \cdot y) \cdot z &\longrightarrow x \cdot (y \cdot z) \\ x \cdot e &\longrightarrow x \\ e \cdot x &\longrightarrow x \end{aligned}$$

    is terminating and confluent.
11. (Harder) Prove that there is always a reduction strategy which completely rewrites all colored redex in a term.
12. Given that the λ-calculus is confluent with respect to β-reduction (under α-equivalence):
    (a) Prove that a λ-term has at most one β-normal form
    (b) Prove that if two terms are β-equal then they can be β-reduced to the same term.
    (c) Give an example of a term which has no β-mormal form.
    (d) (Harder) Give an example of a term which has no β-normal form whose reduction graph, furthermore, is infinite.
13. Are the following two λ-terms equal?

$$(\lambda xy.x(xy))(\lambda xyz.y(xyz))(\lambda xy.x(xy)) \quad \lambda xy.x((\lambda xy.xy)(\lambda xyz.y(xyz))(\lambda xy.x(xy))$$

14. (a) Explain the by-value reduction strategy: what are its shortcomings?
    (b) What is the difference between leftmost outermost (normal order reducion) and outermost reduction (by-name).
    (c) What is head reduction? When is a term in head normal form?
    (d) Demonstrate the different reduction strategies on the following λ-terms:
        i. $(\lambda x.x(\lambda nsz.s(nsz))x)(\lambda sz.s(sz))$
        ii. $(\lambda xy.x)(\lambda z.z)((\lambda x.xx)(\lambda x.xx))$
        iii. $(\lambda xy.yx)(\lambda xy.y)(\lambda x.xx)(\lambda x.xx)$

15. Show that if a term N has a normal form then it has a head normal form: show that the converse is false.

16. Which of the following are true? Justify your answer.

    - Every λ-term is α,β-equivalent to a term in normal form.
    - There is only one fixed point combinator upto α,β-equivalence.
    - When reducing a λ-term to normal form using β-reduction one must be careful (normal order reduction) so that one never reaches a term from which infinitely many β-reduction are possible.
    - If a term has a normal form then, because the λ-calculus is confluent no matter what order you apply β-reductions you will reach the normal form.
    - If a λ-calculus term has a normal form there is a fixed equation in the size of the term which bounds the number of β-reductions necessary to reduce it to that normal form.
    - Two λ-terms are α,β-equivalent if and only if they both have β-normal forms which are α-equivalent.
    - There are some functions which can be computed by the λ-calculus which cannot be computed by a Turing Machine.

17. (Hard) Show that if H is in head normal form and $H \xrightarrow[*]{\beta} H'$ then H' is in head normal form.

18. (Hard) Show that it is undecidable whether a term has a head normal form.

19. (Very Hard) A λ-term N is solvable if, once one closes the term by abstracting all its free variables (i.e. form $\lambda x_1..x_n.N$ where $FV(N) = \{x_1,..,x_n\}$) then there are λ-terms $M_1,....,M_k$ such that $(\lambda x_1..x_n.N)M_1...M_k = \mathsf{I} = \lambda x.x$:

    (a) Show that $\mathsf{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ is solvable;

    (b) Give an example of a term which is not solvable;

    (c) Show that a λ-term, N, is solvable if and only if for any λ-term P, there are $M_1,...,M_k$ such that $(\lambda x_1...x_n.N)M_1...M_k = P$;

    (d) Prove that if N has a head normal form then N is solvable (hint: assume N is in head normal form so is of the form $\lambda x_1...x_n.x_iM_1...M_r$ how would $K_r = \lambda y_1...y_k y_{k+1}.y_{k+1}$ be useful?)

    (e) Prove this N is solvable then N has a head normal form (hint: you know $(\lambda x_1...x_n.N)M_1...M_r$ has a head normal form ... but now assume that $(\lambda x_1...x_n.N)$ has no head normal form – that is the head reduction process *does not terminate*).