CPSC 417: Midterm 2006

Robin Cockett

November 7, 2006

This exam is worth 20% of the course. It is a take home exam you must return it on Tuesday 14th in class. It is an open book exam but you may not discuss it with each other. There are 45 points available (5 of which are bonus).

Part I: Haskell
(15 points)

1. (5 points) Describe the translation from list comprehension syntax core Haskell code. Translate the following function:

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

2. (10 points) 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 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
   ```

Part II: Computational interpretation of λ-terms

(15 points)

1. (3 points) 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.

   Explain how True and False are represented. How do you program an **if ... then ... else** statement in the λ-calculus?

2. (3 points) Explain how to represent the natural numbers in the λ-calculus. Explain what the primitive recursive functions are and show how, without using fixed points, one can represent all the primitive recursive functions.

3. (6 points) Explain what a fixed point combinator is. Give three examples. Show how one can program the following using a fixed points combinator:

   ```
   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.

4. (3 points) 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).
   ```

Part III: Reduction strategies

(15 points)

Normal-order reduction on the λ-calculus uses a leftmost, outermost reduction strategy. Here are the operational rules for normal order reduction. The rules are organized as two groups (1) those which force evaluation under abstractions (2) those which evaluate the application chains by working leftmost:

$$\frac{N \leadsto_c x}{N \leadsto_n x} \quad \frac{M \leadsto_c \lambda x.M' \quad M' \leadsto_n M''}{M \leadsto_n \lambda x.M''} \qquad \frac{M \leadsto_c M'N'}{M \leadsto_n M'N'}$$

$$\frac{}{x \leadsto_c x} \qquad \frac{}{\lambda x.N \leadsto_c \lambda x.N} \qquad \frac{M \leadsto_c x \quad N \leadsto_n N'}{MN \leadsto_c xN'}$$

$$\frac{M \leadsto_c \lambda x.M' \quad M'[N/x] \leadsto_c M''}{MN \leadsto_c M'} \qquad \frac{M \leadsto_c M_1M_2 \quad M_2 \leadsto_n M_2' \quad N \leadsto_n N'}{MN \leadsto_c M_1M_2'N'}$$

Normal order reduction

Recall a normal order reduction will find a normal form if there is one.

A head reduction reduces the term to its principal head normal form. Most functional languages (such as Haskell) use a head reduction strategy.

A term is in head normal form if it is of the form $\lambda x_1..x_n.yM_1...M_r$ (where $n, r \geq 0$). A head reduction always reduces the head redex if there is one. When $M \equiv \lambda x_1...x_n.\underline{(\lambda x.M)N_1}...N_{r+1}$ (where $n, r \geq 0$) then the underlined term is the head redex. Every term has at most one head redex, if it has none then it is in head normal form. A head redex is always the leftmost uppermost redex (but the converse is, of course, not true).

Here are the operational rules for head reduction. They are split into two parts (1) the rules $t \leadsto_h s$ which allow the reduction to go under a λ-abstraction to find the leading chain of applications (2) the rules $t \leadsto_c s$ to process the application chain.

$$\frac{N \leadsto_c x}{N \leadsto_h x} \quad \frac{M \leadsto_c \lambda x.M' \quad M' \leadsto_h M''}{M \leadsto_h \lambda x.M''} \qquad \frac{M \leadsto_c M'N'}{M \leadsto_h M'N'}$$

$$\frac{}{x \leadsto_c x} \qquad \frac{}{\lambda x.N \leadsto_c \lambda x.N} \qquad \frac{M \leadsto_c x}{MN \leadsto_c xN}$$

$$\frac{M \leadsto_c \lambda x.M' \quad M'[N/x] \leadsto_c M''}{MN \leadsto_c M'} \qquad \frac{M \leadsto_c M_1M_2}{MN \leadsto_c M_1M_2N}$$

Head reduction

These questions are more challenging!

1. (3 points) Explain the by-value reduction strategy: what are its shortcomings?

2. (2 points) Show that head reduction applied to a term in head normal form will return the term unchanged (harder: can you show that any returned term - it may not terminate - will always be in head normal form).

3. (2 points) Show that if a term N has a normal form then it has a head normal form (i.e. $N = H$ where H is in head normal form) and show that the converse is false.

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

5. (3 points) Show that it is undecidable whether a term has a head normal form.

6. (5 points) 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 $\lambda x_1...x_n.x_i M_1...M_r$ 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*).