Cornell University
Computer Science 611

Problem Set 1 DUE: Wed., September 7, 5PM

What to turn in

Turn in the written part of the assignment by 5PM on the due date in Upson 4119. The programming part should be submitted using CMS (`http://cms.csuglab.cornell.edu`) by the same time. Most of the assignment is to be done individually, except for the last problem as noted below.

1. Warm up (10 pts.)
   Write the following λ-expressions in their fully-parenthesized, curried forms. Identify the bound variables and indicate which lambda term binds them.

   (a) $\lambda xyz.\, z\ x\ \lambda x.\, x$

   (b) $\lambda xy.\, (\lambda z.\, z\ y)\ \lambda x.\, z\ x$

   (c) $(\lambda x.\, y\ \lambda y.\, x\ y)\ \lambda y.\, x\ y$

   We defined capture-avoiding substitution into a lambda term using the following three rules:

$$\begin{array}{lcll} (\lambda x.\, e_0)\{e_1/x\} & = & \lambda x.\, e_0 & \\ (\lambda y.\, e_0)\{e_1/x\} & = & \lambda y.\, e_0\{e_1/x\} & (\text{where } y \neq x \wedge y \notin FV(e_1)) \\ (\lambda y.\, e_0)\{e_1/x\} & = & (\lambda y'.\, e_0\{y'/y\}\{e_1/x\}) & (\text{where } y' \neq x \wedge y' \notin FV(e_0) \wedge y' \notin FV(e_1)) \end{array}$$

   (d) In these rules, there are a number of conjuncts in the side-conditions whose purpose is perhaps not immediately apparent. Show by counterexample that each of the above conjuncts of the form $x \notin FV(e)$ is independently necessary.

2. Encoding lists (20 pts.)
   In lecture we showed one way to represent the natural numbers in the lambda calculus. In this problem we will encode lists. Consider the following definitions:

$$\begin{array}{rcl} \mathsf{TRUE} & \triangleq & \lambda xy.\, x \\ \mathsf{FALSE} & \triangleq & \lambda xy.\, y \\ \mathsf{NIL} & \triangleq & \lambda x.\, (x\ \mathsf{FALSE}) \\ \mathsf{LIST} & \triangleq & \lambda head.\, \lambda tail.\, \lambda x.\, ((x\ \mathsf{TRUE})\ head)\ tail \end{array}$$

   For convenience, we will write $[x_1, x_2, \ldots, x_n]$ for $(\mathsf{LIST}\ x_1\ (\mathsf{LIST}\ x_2\ (\mathsf{LIST}\ \cdots\ (\mathsf{LIST}\ x_n\ NIL)\ \cdots)))$.

   (a) Show how to write the HEAD and TAIL functions, so that HEAD (LIST *head tail*) = *head* and TAIL (LIST *head tail*) = *tail*. HEAD and TAIL do not need to return anything sensible when applied to NIL. Show that $(\mathsf{HEAD}\ (\mathsf{TAIL}\ [1, 2])) \rightarrow 2$.

   (b) Show how to write a λ-term EMPTY, which returns TRUE if a list is NIL, and FALSE if it is not.

   (c) Write a λ-term MAP which accepts a function and a list and returns a new list containing the results of the function applied to each element of the input list. For example,

$$\mathsf{MAP}\ f\ [1, 2, 3, 4] = [(f\ 1), (f\ 2), (f\ 3), (f\ 4)]$$

   If you get problem 3 working, you can use it to test your solution!

3. Implementing lambda calculus (30 pts.)

   The file lambda.sml contains a partial implementation of some useful lambda calculus mechanisms. In particular, it contains a correct implementation of call-by-value implementation in the function cbv, and you can use it to try out evaluation. The function print_exp can be used to print a human-readable representation of an expression.

   (a) This file also includes most of the implementation of a function nf that reduces a term to $\beta\eta$-normal form, but it doesn't quite work because the substitution function subst is not correct. Fix the implementation of subst and make nf work correctly.

   (b) There is also a very incomplete implementation of a function translate that translates from an extended language to simple lambda calculus. The extended language includes let expressions (like in ML), recursive functions, and pairs. Complete translate so that it faithfully translates extended terms into lambda calculus terms.

   Complete the implementation of lambda.sml and submit the result through CMS. You may do this part of the assignment (and only this part of the assignment) with a partner. Both partners are expected to understand the solution. Make sure to add a comment to lambda.sml indicating who your partner is, if any.