HW6 – Lambda Calculus

CS 476, Fall 2021

1 Instructions

Begin by downloading the file hw6-base.ml from the course website and renaming it to hw6.ml. Then fill in your answers to the problems, adding or modifying definitions as you see fit. Some questions will ask you to solve a problem and put your answer in the comments, while others will ask you to write some code. Submit your completed hw6.ml via Gradescope. As always, please don’t hesitate to ask for help on Piazza (https://piazza.com/class/ksknvqg6ogb2kc).

2 Problems

1. (3 points) What do each of the following lambda-terms evaluate to? Write your answers in the comments in the space provided.

   (a) \(\lambda x. x\) \(y\)
   
   (b) \(\lambda x. (\lambda y. x)\) \(z\)
   
   (c) \(\lambda x. (\lambda x. x)\) \(x\) \((\lambda y. y)\)

2. (4 points) Line 10 of the provided code defines an OCaml type exp representing lambda-terms. Define a variable \(omega : exp\) that represents the lambda-term \((\lambda x. x x)\) \((\lambda x. x x)\). It may help to refer to the provided definition of \(lam1\), which represents the lambda-term \(\lambda x. (\lambda y. x y)\).

3. (5 points) Define a function \(vars\) that returns a list of all the variables that appear in a lambda-term. This function will be used to generate fresh variable names in the definition of capture-avoiding substitution (already provided). Remember that you can write \(x :: l\) to add an element \(x\) to a list \(l\), and \(l1 @ l2\) to combine two lists \(l1\) and \(l2\).

   Once you have completed this problem, \(eval (App (lam1, Var "y"))\) should return \(Some (Lam ("z", App (Var "y", Var "z")))\).

4. (3 points) Use the \(eval\) function to check your answers to problems 1 and 2 (recall that \(omega\) should evaluate to itself). If the results you get are different from what you wrote, see if you can figure out why. In the space provided in the comments, write the commands you ran to check your answers, and any differences you noticed.
5. (for graduate students) Does the provided definition of \texttt{eval} implement call-by-name or call-by-value semantics? How would you change it to implement the other kind of semantics? You can either write your answer in text in a comment, or define a function \texttt{eval2} that demonstrates it.