Homework 9

Due midnight, Wednesday, 4/10/2019

You may work on this homework with one or two partners as you like. If you work with partners just turn in a single write-up of the solutions.

Please submit your homework solutions to gradescope as usual.

1. (0) **Academic Honesty Declaration**

2. (20) **Primitive Recursive Functions**

   In lecture 20, we defined partial recursive functions (due to Kleene in 1936). The primitive recursive functions are those functions definable from the primitive functions using composition and primitive recursion. That is, you can use anything but minimization.

   A simple example of a primitive recursive function is the sum of two numbers:

   $plus(0, n) = n$
   $plus(m + 1, n) = S(plus(m, n))$

   where S is the successor function.

   We were a bit sloppy in the above definition. A more careful definition conforming to the rules for primitive recursive functions would have been:

   $plus(0, n) = P_1^1(n)$
   $plus(m + 1, n) = h(m, plus(m, n), n)$

   where $h(x, y, z) = S(P_2^3(x, y, z))$, which is itself defined by composition from S and $(P_2^3$.

   However, I'll allow you to be sloppy as in the first definition of `plus` in your solution to this problem.

   (a) Please give a primitive recursive definition of multiplication. You can use the above definition for *plus* in the definition.

   (b) Give a proof by induction that all primitive recursive functions are total. I.e., if f is an k-ary primitive recursive function and $n_1, \ldots, n_k$ are natural numbers, then $(f(n_1, \ldots, n_k)$ converges to a value.

   Hint: When doing the case for primitive recursion, do another inductive proof on the values of the first argument.

   (c) Explain briefly why the set of all primitive recursive functions is countable. A very informal proof is fine.

3. (5 points) **Lambda Calculus Reduction** This and the following problem refer to Chapter 4 of the text, "Concepts in Programming Languages" by John C. Mitchell. Section 4.2 of that text is on the lambda calculus. You can find that chapter online.

   Please do problem 4.3 from Mitchell, page 83.

4. (10 points) **Symbolic Reduction**

   Please do problem 4.4 from Mitchell, page 83.

5. (10 points) **Lambda Reduction with Sugar**

   Here is a "sugared" lambda-expression using `let` declarations:

$$\begin{array}{l}\texttt{let } \mathit{compose} = \lambda f.\, \lambda g.\, \lambda x.\, f(g\, x) \texttt{ in} \\ \quad \texttt{let } h = \lambda x.\, x + x \texttt{ in} \\ \quad\quad ((\mathit{compose}\, h)\, h)\, 3\end{array}$$

   The "de-sugared" lambda-expression, obtained by replacing each `let` $z = U$ `in` V by $(\lambda z.\, V)\, U$ is

$$\begin{array}{l}(\lambda \mathit{compose}. \\ \quad (\lambda h.\, ((\mathit{compose}\ h)\ h)\ 3)\ \ (\lambda x.\, x + x)) \\ \quad (\lambda f.\, \lambda g.\, \lambda x.\, f(g\ x))\end{array}$$

   This is written using the same variable names as the `let`-form in order to make it easier to compare the expressions.

   Simplify the desugared lambda expression using reduction. Write one or two sentences explaining why the simplified expression is the answer you expected.

6. (20 points) **Defining Terms in Lambda Calculus**

   In class we defined Church numerals and booleans, and showed how to define more complex functions in the pure lambda calculus. Please show how to define the following functions in the pure lambda calculus. (You may use the functions defined in class in defining these new functions.)

   (a) (5 points)

   Define a function Minus such that Minus m n = m - n if m > n and 0 otherwise. Do not use recursion, but instead define it directly. You may assume that you are given the function Pred defined (but not explained) in class that computes the predecessor of a number (where the predecessor of 0 is 0, predecessor of 1 is 0, 2 is 1, etc.). That is, your answer may include the term Pred without providing a definition of it.

   (b) (5 points)

   Define a function LessThan such that LessThan m n = true iff m < n.

   (c) (5 points)

   Define the recursive fibonacci function fib such that fib 0 = 1, fib 1 = 1, and fib n = fib (n-1) + fib (n-2) for n > 1. *This is tricky as you are not allowed to name the function and use the name in the definition. You must use the Y-combinator and emulate what we did in class defining factorial.*

   (d) (5 points)

   Use your definition of fib from above to calculate fib 2. Do it step by step, showing all of your work. You may assume that fib 1 = 1, fib 0 = 1 and that Plus 1 1 = 2 without showing all of the reduction steps. All other steps in the reduction should be shown.