Homework 13

Due Wednesday, 5/8/2019 at 4 p.m.

Please submit your homework solutions on Gradescope as usual. If you have more than one file to be turned in, please put it in a folder and zip it up before turning it in.

1. (0 points) **Academic Honesty**
2. (10 points) **Type 0 Grammars** Please do problem 23.1f on page 524 of Rice.
3. (15 points) **μ-recursive functions are effectively computable**

   Recall that the primitive recursive functions are defined on non-negative numbers.

   (a) Please show that all primitive recursive functions are effectively computable by showing that for every primitive recursive function $f(x_1, ..., x_n)$ there is a Turing machine that computes it. For simplicity, you may assume that all numbers are written in unary for TM programs. Moreover if a function takes n arguments, you may assume the input is written on the TM tape as $w_1\#w_2\#...\#w_n$ where $w_1, ..., w_n$ are the encodings of the n arguments.

   Hint: Show the base functions can be computed by Turing machines and then show by induction that compositions of primitive recursive functions can be effectively computed and that functions defined by primitive recursion can be effectively computed. This proof structure is similar to the proof that all primitive recursive functions are total – but a bit simpler as you don't need an induction within an induction.

   The Turing machines used in your proof can be described informally (and may use multiple tapes if you like). I'm looking for the broad strokes of what the machines would do rather than detailed transition function (or the kind of diagrams we used to specify Turing machines in class). It should be clear what they do, but you can wave your hands over the details.

   (b) We proved that the primitive recursive functions do not include all total effectively computable functions (essentially because they are all total), we can add one more kind of function construction to get all effectively computable (partial) functions:

   Let $f(x, y_1, \ldots, y_n)$ be a primitive recursive function. Then define

   $$g(y_1, \ldots, y_n) = \mu x.(f(x, y_1, \ldots, y_n) = 1)$$

   where the notation $\mu x.(f(x, y_1, \ldots, y_n) = 1)$ stands for the smallest natural number x such that $f(x, y_1, \ldots, y_n) = 1$ (you can read μ as standing for "minimal"). We say g is the minimization of f on x.

   Show that if f is primitive recursive (and hence total) then $\mu x.(f(x, y_1, \ldots, y_n) = 1)$ is an effectively computable partial function. Show this by explaining how to effectively compute it. The results of part a of this problem allow you to assume that f is effectively computable.

   While I will not ask you to prove it, Turing computable functions on natural numbers are also definable by μ recursive functions.