CS 181 Spring 2018 Homework Week 9

Assigned Thursday 5/31, Due *Tuesday* 6/5 *
* *Note special due date!*

1. Consider the following CFG, G = (V, { a, ";" }, P, S):

   V = { S, L, R }
   P: S -> L S | R
   L -> a ;
   R -> a

   Show that this grammar is not a DCFG by showing that the left-most reduction for some string has a valid string, u, containing an unforced handle. Clearly indicate the unforced handle in u by showing another valid string, u', which begins with the same prefix as u up to and including the handle in u, but such that the handle in u' is different than the handle in u.

2. Let Σ_2 = { #, 0, 1 }. Assume (as mentioned in Lecture) that we can represent any directed graph, G, using strings over an alphabet such as Σ_2. Further assume that we can represent G and two of its nodes using the same alphabet via an encoding such as "g#s#d", where g is a string representing G and s & d are strings representing two nodes in G. Consider the following language, which represents the "Single-Pair Directed Graph Reachability" problem:

   L_2 = { w $\in \Sigma_2^*$ | w = g#s#d is a valid encoding of directed graph, G, and two of its nodes, s & d;
   and there is a directed path in G from s to d }

   Classify L_2 as: Recursive, Recursively Enumerable & Not Recursive, or Non-Recursively Enumerable. *Briefly* justify your answer. (You do not need to provide a proof nor even a detailed description. Just briefly explain why you think your answer is correct.)

For the remaining problems, assume that all of the machines mentioned in the questions are over the alphabet Σ = { a, b }.
Assume the strings representing machines mentioned in the questions are effective encodings over Σ' = { a,b, #,0,1 }.

3. Explain what is wrong with the following "proof" for trying to show that the family of Recursively Enumerable languages are closed under complementation.

   Suppose we are given a Recursively Enumerable language, L, over Σ. Then there must be a Turing Machine (procedure) that recognizes L. Call the TM "M". Since M is a TM, it can be represented as a string, w, over Σ', where w is a valid encoding of M.

   To produce a TM, $\overline{\mathrm{M}}$, that recognizes, $\overline{\mathrm{L}}$, construct $\overline{\mathrm{M}}$ as follows. For any input string, x, over Σ: $\overline{\mathrm{M}}$ first writes the encoding, w, of M onto the work tape and copies its own input string, x, onto the work tape. Then $\overline{\mathrm{M}}$ uses the Universal Turing Machine to simulate M on input string x. If the simulation of M on input x halts and accepts, then $\overline{\mathrm{M}}$ rejects. If the simulation of M on input x halts and rejects, then $\overline{\mathrm{M}}$ accepts.

4. Consider the following language over alphabet (Σ' $\cup$ { \$ }):

   L_4 = { w_1 \$ w_2 | w_1 is a valid encoding of a TM algorithm, M1, and
   w_2 is a valid encoding of a DFA, M_2; and
   there is at least one input string in {a, b}* accepted by both
   M_1 and M_2 }

   Describe (briefly in English) a *procedure* for recognizing L_4.

eof