CPSC 313 — Winter 2020
Assignment 3 — Non-Context-Free Grammars and Turing Machines

Due Friday, April 17, at 11:55 pm on Gradescope

Prior to submission, be sure to familiarize yourself with the **Policies and Guidelines** as well as the **Submission Procedure** as detailed on the assignments course webpage

http://people.ucalgary.ca/~rscheidl/313/assignments.html.

Assignments that don't follow these instructions will incur penalties, possibly even a zero score.

1. **Pumping Lemma for context-free languages**

   Problem 2.31, page 157 (3rd ed) and page 131 (2nd ed) of Sipser. For convenience, the question is reproduced here. Recall that a *palindrome* is a string that reads the same forwards as backwards. Let L be the language of all palindromes over the alphabet $\{0, 1\}$ that contain an equal number of 0's and 1's. For example, 0110 and 010110011010 belong to L, but 0101 and 010 do not. Use the Pumping Lemma for context-free languages to prove that L is not context-free.

2. **Closure properties of Turing recognizable languages**

   Prove that the set of Turing recognizable languages is closed under the operations of concatenation and Kleene closure.

   Proceed as follows. Let L_1 and L_2 be Turing-recognizable languages over some alphabet Σ, and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be Turing machines that recognize L_1 and L_2, respectively, so $L(\mathcal{M}_1) = L_1$ and $L(\mathcal{M}_2) = L_2$. Construct Turing machines $\mathcal{M}$ and $\mathcal{M}^*$ that recognize L_1L_2 and L_1^*, respectively. It is easiest to construct these as a *non-deterministic* Turing machine, so you can non-deterministically split up your input string as a concatenation (of two strings, i.e. $w = w_1w_2$, for the concatenation and of an arbitrary number of strings, i.e. $w = w_1w_2 \dots w_n$, for the Kleene closure).

   For both operations, prove your construction correct. That is, prove that $L(\mathcal{M}) = L_1L_2$, $L(\mathcal{M}^*) = L_1^*$, and acceptance for both $\mathcal{M}$ and $\mathcal{M}^*$ happens in a finite number of steps.

3. **Turing machine design**

   Consider the language $L_{n\text{-steps}}$ consisting of all encodings $\langle \mathcal{M}, w, n \rangle$ where

   - $\mathcal{M}$ is Turing machine,
   - w is a string,
   - n is a positive integer and
   - $\mathcal{M}$ halts on input w in exactly n steps (i.e. in n steps but n no fewer that n steps).

   Prove that $L_{n\text{-steps}}$ is decidable by constructing a decider $\mathcal{M}_{n\text{-steps}}$ for $L_{n\text{-steps}}$. Prove your decider correct.