CSE 105
Homework 5
Due: Monday May 15, 2017

INSTRUCTIONS

Upload a **single file** to Gradescope for each group. All group members' names and PIDs should be on **each** page of the submission.

Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

READING Sipser Sections 2.1, 2.2, examples from 2.3, 3.1

KEY CONCEPTS Non regular languages, Pumping Lemma, context-free grammars, context-free languages, derivations, parse trees, ambiguous grammars, leftmost derivations, Turing machines, deciders.

---

1. (**9 points**) For any string $w \in \Sigma^*$, the expression w^R denotes the reverse of w, i.e., w written backward. The reversal of a language L is $L^R = \{w^R \mid w \in L\}$ and is obtained by reversing every word of L.

[[**Warmup**: Prove that the class of regular languages is closed under reversal.
(*This part will not be graded but is recommended practice.*)]]

(a) Let the language $D = \{w \mid w$ is a string over $\{0,1\}$ that ends with $01\}$. Design a CFG describing D.

For full credit, you must correctly specify the grammar and informally justify why it describes the language.

(b) Describe D^R in set builder notation. Design a CFG describing D^R.

For full credit, you must correctly specify the grammar and informally justify why it describes the language D^R (include a description of this language, D^R, as part of your justification).

(c) Prove that the class of context-free languages is closed under reverse, i.e., prove that if L is context-free then L^R is also context-free.

For full credit, you must set up the proof so that all variables you use have been defined, then clearly and correctly give the construction, and then informally justify why the construction works.

Hint: We have implemented this construction in Haskell. The file `CFG.hs` defines the CFG data type as well as the `reverseCFG` function, which takes as input a CFG and returns as output a CFG generating the reverse of the language generated by the input grammar. Load the file into the haskell interpreter by running the command `ghci CFG.hs` at the command line prompt. Then, at the ghci prompt, execute `cfg1` to see the rules of a sample input grammar (*can you determine the language generated by this grammar?*). Execute `reverseCFG cfg1` to see the output of the reversal construction for this example. Open the `CFG.hs` file in any text editor to look at the definition of the `reverseCFG` function. Can you extrapolate from it the formal definition you need to give in your proof?

2. (**6 points**) Let $G = (\{S, C, B, A\}, \Sigma, R, S)$ be a CFG where

$$\Sigma = \{\text{if}, \text{bool}, \text{then}, \text{else}, \text{assnt}, ' \ '\}$$

(where the last terminal symbol is the space character) and the set of rules is given by

$$\begin{aligned} S &\to A \mid C \mid B \\ C &\to \text{if bool then } S \\ B &\to \text{if bool then } S \text{ else } S \\ A &\to \text{assnt} \end{aligned}$$

This grammar is intended to parse nested conditional statements. However, it is ambiguous. Prove that this grammar is ambiguous by giving a string that is generated by this grammar but has two different leftmost derivations (or parse trees). Include these leftmost derivations as part of your proof.

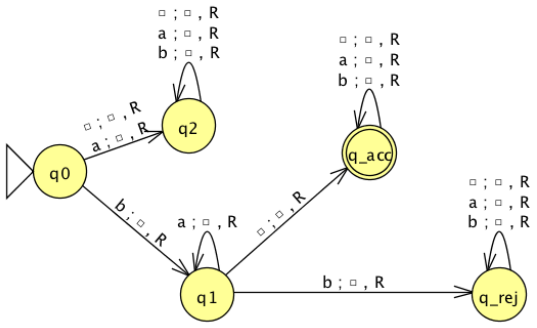
3. (**8 points**) Let A be the following language over the alphabet $\{0, 1, 2\}$

$$A = \{1^n 0^m 2^k \mid n, m, k \geq 0 \text{ and } k \leq n + m\}$$

In JFLAP, draw the state diagram of a PDA M such that M recognizes A and M has at most five states. Along with the JFLAP diagram, include a brief description (in English) of why each state is included and why you chose to make each state either accepting or non-accepting.

Hint: You can use JFLAP to test out your PDA design on some sample string. Open JFLAP, choose "New > Pushdown Automaton" and, if given the choice, select "Single character input" for the "Type of PDA Input". Once you create your state diagram, choose "Multiple Run" from the "Input" menu. Pick your test suite of strings and run to debug your design.

4. (**12 points**) Consider the Turing machine, M, determined by the following state diagram.



(a) Give a string of length 3 that is accepted by M. Explain why it is accepted by tracing through the computation of M on this string.

(b) Give a string of length 3 that is not accepted by M. Explain why it is not accepted by tracing through the computation of M on this string.

(c) Is M a decider? Prove your answer.

(d) Is the language of M regular? Is it context-free? Prove your answers.