CSE 105
Individual Homework 4
Due: Tuesday May 8, 2018 at 11:00PM (on Gradescope)

INSTRUCTIONS
This **Individual HW4** must be completed without any collaboration with other students in this class. The only allowed sources of help for this homework are the class textbook, notes, and podcast, and the instructional team.

Your homework **must be typed**. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

READING Sipser Sections 2.1, 2.2, 2.3.

KEY CONCEPTS Pushdown automata, context-free grammars, context-free languages, derivations, parse trees, ambiguous grammars, leftmost derivations.

---

1. (**10 points**) Consider the context-free grammars

$$G_1 = (\{S, T\}, \{a, b, c\}, R_1, S) \qquad\qquad G_2 = (\{S, T, X\}, \{a, b, c\}, R_2, S)$$

where R_1 is the set of rules containing

$$\begin{aligned} S &\to aSc \mid T \\ T &\to bTc \mid \varepsilon \end{aligned}$$

and R_2 is the set of rules containing

$$\begin{aligned} S &\to TbTSXbX \mid \varepsilon \\ T &\to aT \mid \varepsilon \\ X &\to cX \mid \varepsilon \end{aligned}$$

To show that a string is in $L(G_i)$, we give a derivation of the string using the rules in R_i. For example, the derivation

$$S \implies aSc \implies aTc \implies abTcc \implies abcc$$

proves that $abcc \in L(G_1)$.

a. Is the empty string in $L(G_1)$? Is the empty string in $L(G_2)$?

b. Show that the string $abbccc$ is in $L(G_1)$ by giving a derivation of it using the rules in R_1.

c. Show that the string $abbccc$ is in $L(G_2)$ by giving a derivation of it using the rules in R_2.

d. Is $L(G_1)$ infinite? Is $L(G_2)$ infinite?

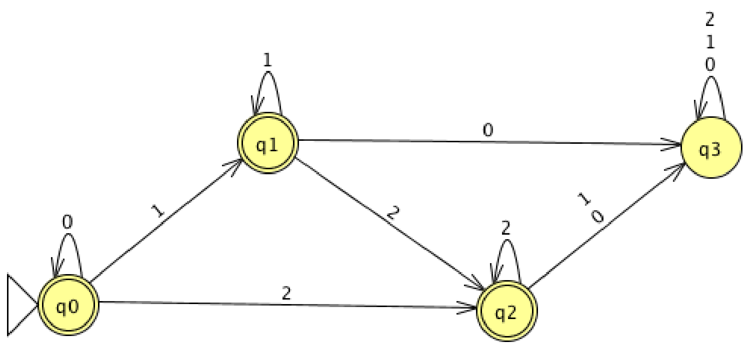
e. Is $L(G_1) \subseteq L(a^*b^*c^*)$? Is $L(G_2) \subseteq L(a^*b^*c^*)$?

No justifications required for credit for this question; but, as always, they're a good idea for your own benefit.

2. (**10 points**). In class (Monday April 29), we proved that every regular set is PDA-recognizable by modifying the state diagram of an NFA to get a PDA. In the book on page 107, the top paragraph describes a procedure for converting DFA to CFGs:

> You can convert any DFA $M = (\{q_0, \ldots, q_n\}, \Sigma, \delta, q_0, F)$ into an equivalent CFG as follows. Make a variable R_i for each state q_i of the DFA. Add the rule $R_i \to aR_j$ to the CFG if $\delta((q_i, a)) = q_j$ is a transition in the DFA. Add the rule $R_i \to \varepsilon$ if q_i is an accept state of the DFA. Make R_0 the start variable of the grammar.

Consider the state diagram of a DFA recognizing the language $A = L(0^*1^*2^*)$.



a. What would the **input alphabet** of a PDA recognizing the **complement** of A be?

   What would the **stack alphabet** of a PDA recognizing the **complement** of A be?

   Draw the state diagram of this PDA recognizing the **complement** of A.

b. Use the process above to build a CFG G generating the **complement** of A. Specifically, fill in the blanks below: $G = (V, \Sigma, R, S)$ where

   $V =$ ____________________

   $\Sigma =$ ____________________

   $R =$ ____________________

   $S =$ ____________________

No justifications required for credit for this question; but, as always, they're a good idea for your own benefit.

3. (**10 points**)

a. Suppose $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is a PDA. We can define a new PDA N so that $L(M) = L(N)$ and N is guaranteed to have an empty stack at the end of any accepting computation. Informally, the construction is as follows: Add three new states q_1', q_2', q_3' and one new stack symbol #.

- One of the new states q_1' will be the new **start** state and it has a spontaneous transition to the old start state q_0 which pushes the new stack symbol # to the stack.
- The transitions between the old states are all the same.
- From each of the old accept states, **add** a spontaneous transition (that doesn't modify the stack) to the second new state q_2'.
- In this state q_2', pop all old stack symbols from the stack without reading any input.
- When the new stack symbol # is on the top of the stack, transition to the third new state q_3' and accept.

Complete the formalization of this description by filling in the blanks in the transition function below.

$$N = (Q \cup \{q_1', q_2', q_3'\}, \Sigma, \Gamma \cup \{\#\}, \delta', q_1', \{q_3'\})$$

where we assume $\{q_1', q_2', q_3'\} \cap Q = \emptyset$ and $\# \notin \Gamma$, and

$\delta'(\ (q, x, y)\ ) = \delta(\ (q, x, y)\ )$	if $q \in Q, x \in \Sigma, y \in \Gamma_\varepsilon$
$\delta'(\ (q, x, y)\ ) = \delta(\ (q, x, y)\ )$	if $q \in Q, q \notin F, x = \varepsilon, y \in \Gamma_\varepsilon$
$\delta'(\ (q, x, y)\ ) =$ ________________	if $q \in F, x = \varepsilon, y \in \Gamma$
$\delta'(\ (q, x, y)\ ) =$ ________________	if $q \in F, x = \varepsilon, y = \varepsilon$
$\delta'(\ (q_1', \varepsilon, \varepsilon)\ ) =$ ________________	
$\delta'(\ (q_2', \varepsilon, x)\ ) =$ ________________	if $x \in \Gamma$
$\delta'(\ (q_2', \varepsilon, \#)\ ) =$ ________________	
$\delta'(\ (q, x, y)\ ) = \emptyset$	otherwise

Some hints:

- *The transition function of the PDA N has domain*

$$Q \cup \{q_1', q_2', q_3'\} \times \Sigma_\varepsilon \times (\Gamma \cup \{\#\})_\varepsilon$$

and codomain

$$\mathcal{P}(\ Q \cup \{q_1', q_2', q_3'\} \times (\Gamma \cup \{\#\})_\varepsilon\ ).$$

Make sure the outputs you specify when you fill in the blank are **sets** *of* **ordered pairs** *of the right type.*

- *The new machine N has only one accept state: the new state q_3' (where do you see this in the formal definition?).*

b. Suppose $G = (V, \Sigma, R, S)$ and define the new grammar $G' = (V, \Sigma, R \cup \{S \to SS\}, S)$.

i **True** or **False**: $L(G') = L(G) \circ L(G)$ *for all grammars* G.

ii **True** or **False**: $L(G') = (L(G))^*$ *for all grammars* G.

iii **True** or **False**: $L(G') = \mathit{DOUBLE}(L(G))$ *for all grammars* G.

No justifications required for credit for this question; but, as always, they're a good idea for your own benefit.