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

\[ G_1 = (\{S, T\}, \{a, b, c\}, R_1, S) \quad G_2 = (\{S, T, X\}, \{a, b, c\}, R_2, S) \]

where \( R_1 \) is the set of rules containing

\[
S \rightarrow aSc \quad T \\
T \rightarrow bTc \quad \varepsilon
\]

and \( R_2 \) is the set of rules containing

\[
S \rightarrow TbTSXbX \quad \varepsilon \\
T \rightarrow aT \quad \varepsilon \\
X \rightarrow cX \quad \varepsilon
\]

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

\[
S \Rightarrow aSc \Rightarrow aTc \Rightarrow abTcc \Rightarrow abcc
\]

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 \( abbecc \) is in \( L(G_1) \) by using the rules in \( R_1 \).

c. Show that the string \( abbecc \) 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 $R_i \rightarrow aR_j$ to the CFG if $\delta((q_i, a)) = q_j$ is a transition in the DFA. Add the rule $R_i \rightarrow \varepsilon$ 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 $A = L(0^*1^*2^*)$.

![State Diagram](image)

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

\[
V = \underline{\text{__________________________}} \\
\Sigma = \underline{\text{__________________________}} \\
R = \underline{\text{__________________________}} \\
S = \underline{\text{__________________________}}
\]

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

\[
\begin{align*}
\delta'( (q, x, y) ) &= \delta( (q, x, y) ) & &\text{if } q \in Q, x \in \Sigma, y \in \Gamma \\
\delta'( (q, x, y) ) &= \delta( (q, x, y) ) & &\text{if } q \in Q, q \notin F, x = \varepsilon, y \in \Gamma \\
\delta'( (q, x, y) ) &= & &\text{if } q \in F, x = \varepsilon, y \in \Gamma \\
\delta'( (q'_1, \varepsilon, \varepsilon) ) &= & &\text{if } x \notin \Gamma \\
\delta'( (q'_2, \varepsilon, \#) ) &= & &\text{if } x \in \Gamma \\
\delta'( (q, x, y) ) &= \emptyset & &\text{otherwise}
\end{align*}
\]

Some hints:

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

$$Q \cup \{q'_1, q'_2, q'_3\} \times \Sigma \times (\Gamma \cup \{\#\})$$

and codomain

$$\mathcal{P}( Q \cup \{q'_1, q'_2, q'_3\} \times (\Gamma \cup \{\#\}) ).$$

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 $G' = (V, \Sigma, R \cup \{S \rightarrow SS\}, S)$.

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: $L(G') = DOUBLE(L(G))$ for all grammars $G$.

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