Solutions to Supplementary Problems

S7.1 Show, using the pumping lemma for context-free languages, that the language

$$\{ww \mid w \in \{a,b\}^*\}$$

is not context-free. (*Hint:* Consider strings of the form $a^n b^n a^n b^n$.)

Solution. See Example 2.38 in the Sipser book.

S7.2 Prove that the class of context-free languages is not closed under intersections and complements. (*Hint:* Represent the language $\{a^k b^k c^k \mid k \geq 0\}$ as the intersection of two context-free languages.)

Solution. The language $L = \{a^k b^k c^k \mid k \geq 0\}$ was shown to be not context-free as an example in Section 7.6 of the lectures (also Example 2.36 in the Sipser book). To prove that the class of context-free languages is not closed under intersections, it now suffices to find two context-free languages L_1 and L_2 such that $L = L_1 \cap L_2$. Languages $L_1 = \{a^i b^k c^k \mid i,k \geq 0\}$ and $L_2 = \{a^k b^k c^i \mid i,k \geq 0\}$ satisfy this condition.

It then follows that context-free languages cannot be closed under complementations either, because they are closed under unions, and by De Morgan's Laws intersections can be represented in terms of complements and unions as $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

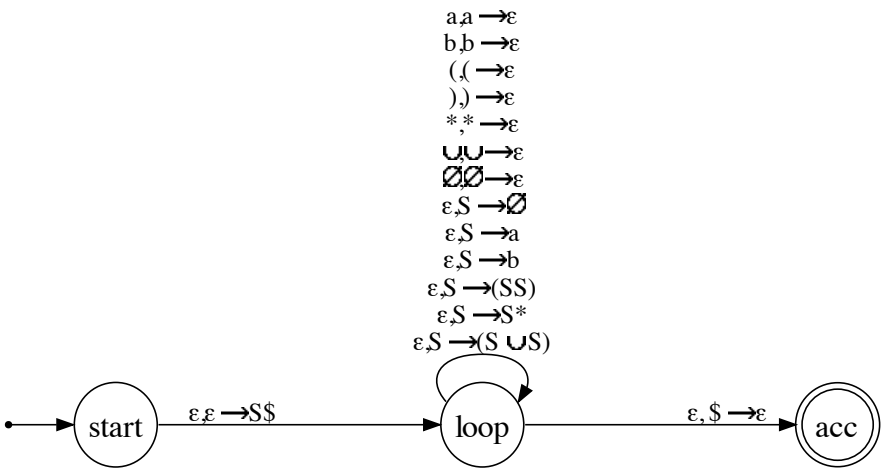
To still validate that the languages L_1 and L_2 indeed are context-free, let us present context-free grammars generating them. Language L_1 is generated by grammar $G_1 = (\{S,A,B\}, \{a,b,c\}, R_1, S)$, where $R_1 = \{S \to AB, A \to aA \mid \varepsilon, B \to bBc \mid \varepsilon\}$. Similarly, language L_2 is generated by grammar $G_2 = (\{S,A,B\}, \{a,b,c\}, R_2, S)$, where $R_2 = \{S \to AB, A \to aAb \mid \varepsilon, B \to cB \mid \varepsilon\}$.

S7.3 Design a pushdown automaton corresponding to the grammar

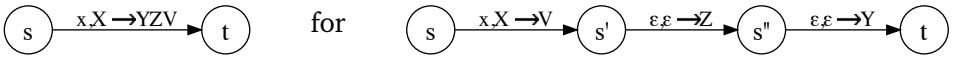
$$S \quad \to \quad (SS) \mid S^* \mid (S \cup S) \mid \emptyset \mid a \mid b$$

where the set of terminal symbols is $\Sigma = \{(,), \cup, *, \emptyset, a, b\}$.

Solution. Let us denote the example grammar as G. Following the construction in Theorem 7.4 of Lecture 7 (Lemma 2.21 in the Sipser book), we can design the following pushdown automaton (presented here in Sipser's notation) that recognises the language $L(G)$:



Here the stack symbol \$ is used to mark the bottom of the stack. We also use the rather standard abbreviation



that lets us represent pushing multiple symbols on the stack by a single transition arc.

The idea is that the automaton can generate a leftmost derivation for any word in $L(G)$ by applying transitions of form $\varepsilon, S \to X_1 ... X_n$ for each production $S \to X_1 ... X_n$ in G. To validate that the terminal string generated on the stack matches the given input, the automaton has transitions of form $x, x \to \varepsilon$ for each terminal symbol x of G. Applying these transitions matches the generated string to the input symbol by symbol, from left to right, and at the same time pops the generated string off the stack.

Here is how the automaton processes input string $(a \cup b^*)$:

State	Input	Stack	Next transition
q_{start}	$(a \cup b^*)$	ε	$\varepsilon, \varepsilon \to S\$$
q_{loop}	$(a \cup b^*)$	$S\$$	$\varepsilon, S \to (S \cup S)$
q_{loop}	$(a \cup b^*)$	$(S \cup S)\$$	$(,(\to \varepsilon$
q_{loop}	$a \cup b^*)$	$S \cup S)\$$	$\varepsilon, S \to a$
q_{loop}	$a \cup b^*)$	$a \cup S)\$$	$a, a \to \varepsilon$
q_{loop}	$\cup b^*)$	$\cup S)\$$	$\cup, \cup \to \varepsilon$
q_{loop}	$b^*)$	$S)\$$	$\varepsilon, S \to S*$
q_{loop}	$b^*)$	$S^*)\$$	$\varepsilon, S \to b$
q_{loop}	$b^*)$	$b^*)\$$	$b, b \to \varepsilon$
q_{loop}	$^*)$	$^*)\$$	$^*, ^* \to \varepsilon$
q_{loop}	$)$	$)\$$	$),) \to \varepsilon$
q_{loop}	ε	$\$$	$\varepsilon, \$ \to \varepsilon$
q_{acc}	ε	ε	

This corresponds to the leftmost derivation $\underline{S} \underset{\text{lm}}{\Rightarrow} (\underline{S} \cup S) \underset{\text{lm}}{\Rightarrow} (a \cup \underline{S}) \underset{\text{lm}}{\Rightarrow} (a \cup \underline{S^*}) \underset{\text{lm}}{\Rightarrow} (a \cup b^*)$.

Note: The language $L(G)$ generated by the grammar is of course that of all syntactically well-formed regular expressions over the alphabet $\{a, b\}$.