

# CS3800: Theory of Computation — Summer II '22 — Drew van der Poel

## Homework 3

Due Friday, July 29 at 11:59pm via [Gradescope](#)

Name: Steve Liu

Collaborators:

- Make sure to put your name on the first page. If you are using the  $\text{\LaTeX}$  template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Friday, July 29 at 11:59pm via [Gradescope](#). No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden.

**Problem 1.** *Context-Free Grammars* (7 points)

In the following problems, the alphabet  $\Sigma = \{a, b\}$ . Give a context-free grammar for each of the following languages.

- (a) [3 pts.]  $L_a = \{a^n b^n \mid n > 1 \text{ is not a multiple of } 3\}$

Show how to generate  $aaaabbbb$  with your grammar.

**Solution:**

$A \rightarrow aaaAbbb \mid aaaabbbb \mid aabb$

To generate  $aaaabbbb$ :  $A \rightarrow aaaabbbb$

- (b) [4 pts.]  $L_a = \{w \mid w \text{ has twice as many } a\text{'s as } b\text{'s}\}$

Show how to generate  $aababaaab$  and  $aaaabb$  with your grammar.

**Solution:**

$A \rightarrow AA \mid aAbAa \mid aAaAb \mid bAaAa \mid \varepsilon$

To generate  $aababaaab$ :  $A \rightarrow AA \rightarrow (aAaAb)(A) \rightarrow (aAaAb)(AA) \rightarrow (aAaAb)(aAbAa)(A) \rightarrow (aAaAb)(aAbAa)(AA) \rightarrow (aAaAb)(aAbAa)(aAaAb)(A) \rightarrow$  (going to skip a few steps and replace all of the  $A$ 's with the empty string)  $\rightarrow aababaaab$

To generate  $aaaabb$ :  $A \rightarrow aAaAb \rightarrow aAa(aAaAb)b \rightarrow$  (skipping a few steps, replacing all of the  $A$ 's with the empty string)  $\rightarrow aaaabb$

**Problem 2. CFGs and PDAs (7 points)**

Consider the following context-free grammar  $G$ :

$$S \rightarrow aWb|bWa$$

$$W \rightarrow aW|bW|\epsilon$$

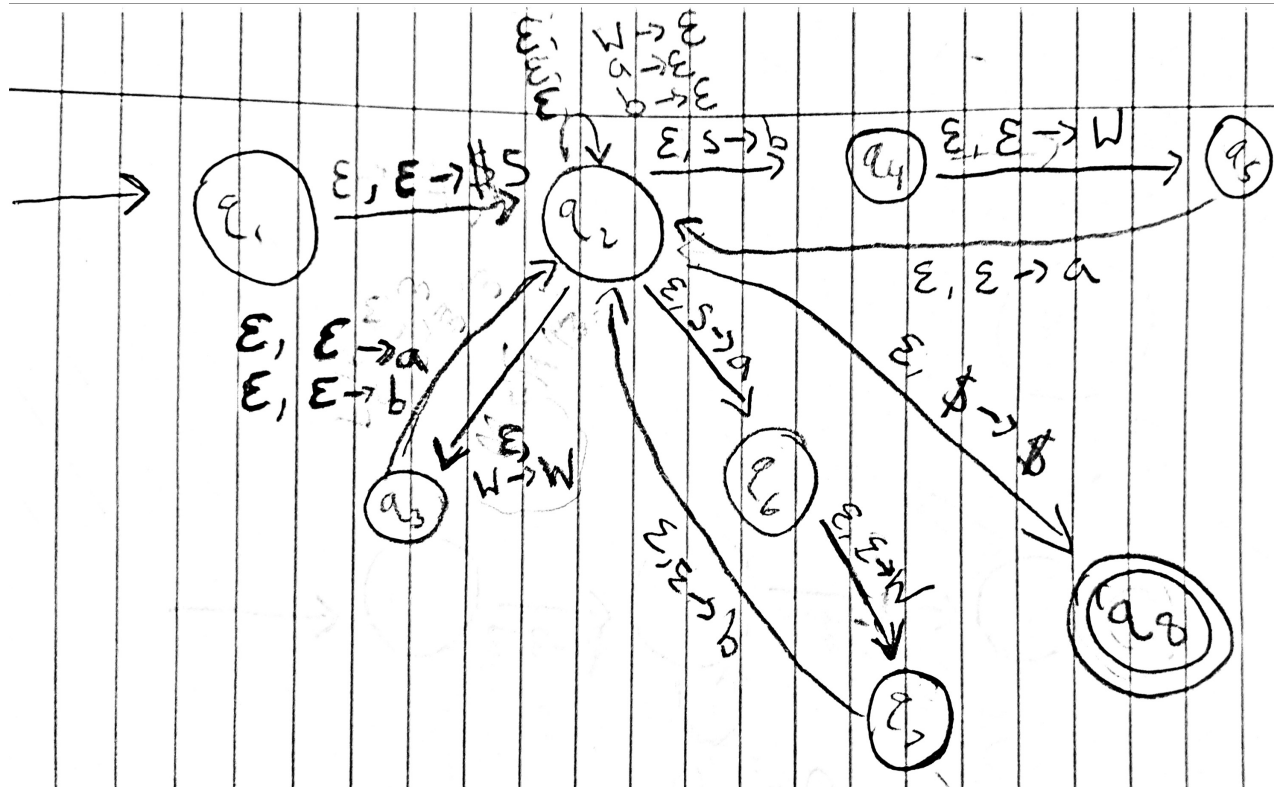
- (a) [2 pts.] Describe the set of strings which can be generated by  $G$ .

**Solution:**

The set of all strings in which it is an empty string, or the entire string is all  $a$ 's, or the entire string is all  $b$ 's, or the first input is  $a$  and the last input is  $b$  or the first input is  $b$  and the last input is  $a$ .

- (b) [5 pts.] Give a PDA which recognizes the language given by grammar  $G$ . You should show all necessary states for "guessing" rule  $S$ , but can use shorthand otherwise. Specify  $\Sigma$  and  $\Gamma$ .

**Solution:**



$$\Sigma = \{a, b\}$$

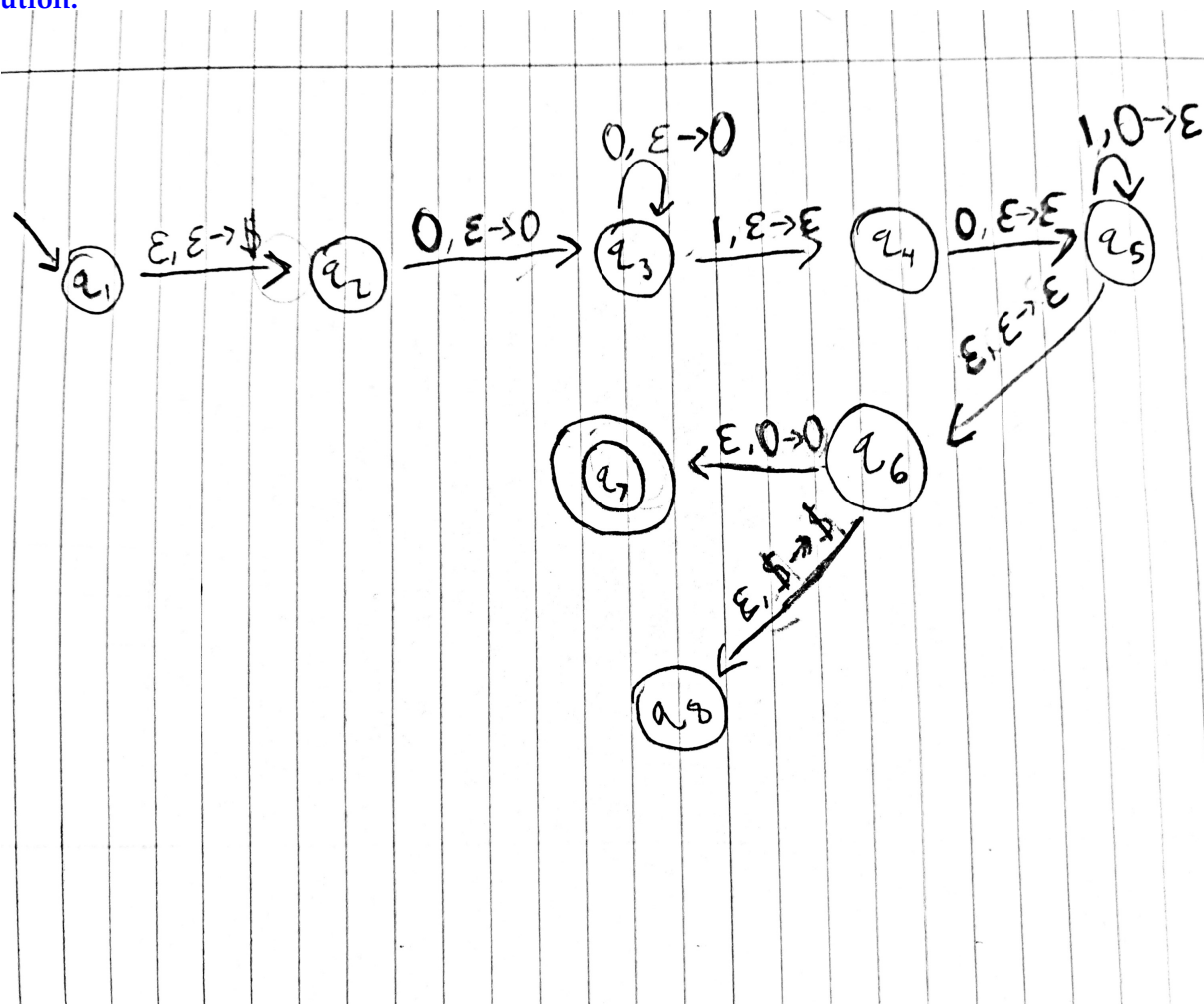
$$\Gamma = \{\$, S, W, a, b\}$$

**Problem 3. PDAs (8 points)**

Consider  $\Sigma = \{0, 1\}$  and language  $L = \{0^n 101^m | n > m; n, m \in \mathbb{N}\}$ .

Show that  $L$  is context-free by giving a PDA which recognizes it. You should give a complete PDA. Your machine should accept strings 00000001011, 010, 00101, and 00000010, but not 10 0010111, or 0101. Explain why it accepts 00101 and why it does not accept 0101.

**Solution:**



00101: We start at  $q_1$ , and epsilon transition while initializing the stack to  $q_2$ . At  $q_2$ , we read the first 0 and push it onto the stack to transition to  $q_3$ . At  $q_3$ , we read the next 0 and push it onto the stack, staying at  $q_3$ . Again, at  $q_3$ , the next input is 1, so we take the 1 transition to  $q_4$  while pushing nothing onto the stack. At  $q_4$ , our next input is 0 so we transition to  $q_5$  and push nothing to the stack. At  $q_5$  we break into two branches, one where read the last input, 1, stay at  $q_5$  and remove a 0 off of the stack or we can take the epsilon transition to  $q_7$ : If we take the epsilon transition, then we would end up at  $q_6$  and then since the top of our stack is a 0, we can take the epsilon transition to  $q_7$ . Since in this case we still have a 1 to read and there are no transitions in  $q_7$  for the 1 then

this branch fails. If we take the  $q5$  transition and remove a 0 off of the stack, then we can take the transition  $q6$  and then since the top of our stack is still a 0, we can take the epsilon transition to  $q7$ . At  $q7$ , since we are finished with reading the string and we are in an accept state, the string is accepted.

0101: We start at  $q1$ , and epsilon transition while initializing the stack to  $q2$ . At  $q2$ , we read the first 0 and push it onto the stack to transition to  $q3$ . At  $q3$  we read the 1 to transition to  $q4$  and then the 0 to transition to  $q5$ , pushing nothing onto the stack in both transitions. At  $q5$ , we once again break into two branches: one where we take the epsilon branch and one where we stay at  $q5$ , read the 1, and pop a 0 off of the stack. For the same reason, taking the epsilon transition would result in the branch rejecting at  $q7$ . In the other branch, after we read the 1 and popped the top 0 off of the stack, then we can take the epsilon transition to  $q6$ . At  $q6$ , since our stack is now empty, we take the transition to  $q8$  where the string will be rejected.

**Problem 4. Non Context-Free Languages (4+4=8 points)**

- (a) Prove that language  $L = \{w | w \in \{a, b, c\}^* \text{ and the number of } a\text{'s is equal to the number of } b\text{'s and the number of } a\text{'s is greater than the number of } c\text{'s}\}$  is not context-free.

**Solution:**

Assume that  $L$  is a context-free grammar, and so the context-free grammar pumping lemma applies. Let  $P$  be the pumping length. Let  $S = a^{2P}c^Pb^{2P}$ ,  $S \subseteq L$ , and  $|S| \geq P$ . Because we know that  $|vxy| \leq P$ ,  $vxy$  cannot have both  $a$ 's and  $b$ 's. We can then break the string of  $vy$  into 2 cases:

Case 1:  $vy$  contains all of the same letters. If  $vy$  consists of all  $a$ 's or all  $b$ 's then if we let  $i = 2$ ,  $uvvxyyz$ , then the string is no longer in the language since either  $|a| > |b|$  or  $|b| > |a|$  depending on the scenario. If  $vy$  consists of only  $c$ 's then if we let  $i = P$ ,  $uv^Pxy^Pz$ , then we know that  $|c| > |a|$  which means that  $S$  is no longer in the language.

Case 2:  $vy$  contains some mix of  $a$ 's and  $c$ 's or  $b$ 's and  $c$ 's. If this is the case then if we let  $i = 2$ ,  $uvvxyyz$ , we will have an unequal number of  $a$ 's and  $c$ 's depending on the scenario. If  $vy$  is some mixture of  $a$ 's and  $c$ 's then we will have more  $a$ 's than  $c$ 's after pumping and if  $vy$  consists of only  $b$ 's and  $c$ 's then we will have more  $c$ 's than  $a$ 's after pumping.

Thus, by contradiction, the string  $S$  cannot uphold the pumping lemma despite being in the grammar,  $L$ , which means that  $L$  is not context-free.

- (b) Prove that language  $L = \{a^l b^{l^2} | l \in \mathbb{N}\}$  is not context-free.

**Solution:**

Assume that  $L$  is a context-free grammar, and so the context-free pumping lemma applies. Let  $P$  be the pumping length. Let  $S = a^P b^{P^2}$ ,  $S \subseteq L$ , and  $|S| \geq P$ . Because we know that  $|vxy| \leq P$ ,  $vxy$  either can be all  $a$ 's, all  $b$ 's or, a mixture of  $a$ 's and  $b$ 's.

Case 1:  $vy$  consists of all  $a$ 's. We know that  $vy$  is non-empty ( $|vy| > 0$ ), so if we let  $i = 2$ ,  $uvvxyyz$ , must mean that the number of  $a$ 's must increase. Since the number of  $a$ 's increased by some arbitrary amount and the number of  $b$ 's has not changed, then  $|a|$  is no longer equal to  $\sqrt{|b|}$  which means  $uvvxyyz$  is not in  $L$ . This means that  $S$  does not uphold the pumping lemma for this case.

Case 2:  $vy$  consists of all  $b$ 's. We know that  $vy$  is non-empty ( $|vy| > 0$ ), so if we let  $i = 2$ ,  $uvvxyyz$ , must mean that the number of  $b$ 's must increase. Since the number of  $b$ 's increased by some arbitrary amount and the number of  $a$ 's has not changed, then  $|b|$  has to now be greater than the  $|a|^2$  which means  $uvvxyyz$  is not in  $L$ . This means that  $S$  does not uphold the pumping lemma for this case.

Case 3:  $vy$  consists of some mix of  $a$ 's and  $b$ 's. If we let,  $i = 2$ , we know that the number of  $a$ 's and  $b$ 's must increase by some arbitrary amount because  $|vy| > 0$ . Let's assume that there are

$k$  number of a's in  $vy$  we then know that the number of b's is  $|vy| - k$ . Since  $a = P$  and  $|b| = P^2$  in the original  $S$ , the new number of a's after pumping  $i = 2$  times is  $(P - k) + 2k = P + k$  and the new number of b's is  $P^2 - (|vy| - k) + 2(|vy| - k) = P^2 + |vy| - k$ . If we square the new number of a's,  $(P + k)^2$ , this value should equal to the number of new b's. Putting it all together we get the equation:  $(P + k)^2 = P^2 + |vy| - k$ .

Simplification:

$$(P + k)^2 = P^2 + |vy| - k$$

$$P^2 + 2Pk + k^2 = P^2 + |vy| - k$$

$$2Pk + k^2 = |vy| - k$$

$$2Pk + k^2 + k = |vy|$$

However, since we know that  $k > 0$ , this means that  $2Pk + k^2 + k$  must be greater than  $P$  which violates the condition that  $|vxy| \leq P$ .