Int 201: Decision Computation and Language
Tutorial 7 Solution

Dr. Chunchuan Lyu

November 21, 2023

Question 1. Draw PDA for language $\{ww^R | w \in \{0,1\}^*\}$, where w^R is reversed of w, and show they are equivalent.

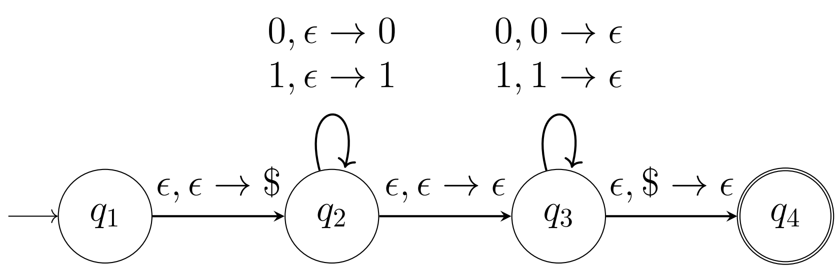

Figure 1: PDA for Q1

Solution 1. Proof: we need to show for all $s = ww^R$, the PDA accepts it and for all s being accepted by PDA, it can be written as $s = ww^R$.

First, for any $s = ww^R$, the non-deterministic PDA has one branch that enters $q3$ when w has been processed. At that time, we denote this as time 0, the $stack_0[i] = w[len(w) - 1 - i]$ where $stack_0[0]$ is the top of stack and $0 \leq i \leq len(w) - 1$. In q_3, it cleans stack elements one by one, when it is at the kth pop $w^R[k] = stack_k[0] = stack_0[k] = w[len(w) - 1 - k]$ always allow the transition to happen. When everything is being consumed, we see the \$ and enter the accepting state.

For string S being accepted by this PDA, it is clear that the acceptance history can be broken into $s = ab$ where a, b are string of equal length. Because to see the \$, the number of symbols processed at q_2 need to equal in number to that of q_3. When q_3 pop kth element, we have $b[k] = a[-k]$ due to the stack pop what's being pushed. In other words, we have $b = a^R$, so it is in $\{ww^R | w \in \{0,1\}^*\}$.

The point is that a language might be accepted by a PDA, but also some other strings not in the language (otherwise, we could have a PDA accepting every string). To exclude this possibility, we need to show all strings accepted by the PDA are in the language.

Question 2. Can you draw PDA for language $\{ww|w \in \{0,1\}^*\}$? Can a PDA with two stacks recognize the language $\{ww|w \in \{0,1\}^*\}$?

Solution 2. No, we can't. Intuitively, the string can be arbitrary long, so we need stack for memory, but stack can only be accessed through a last in first out fashion. It cannot recall the first element without popping out everything in between. We will need pumping lemma (next lecture) to show this is indeed not CFL.

Yes, we can, as we can reverse the stack element with the second stack. Note that with two stacks, the transition becomes a 5-tuple (tape,pop stack1,pop stack2$\rightarrow$ push stack1, push stack2). We use the following PDA with two stacks:

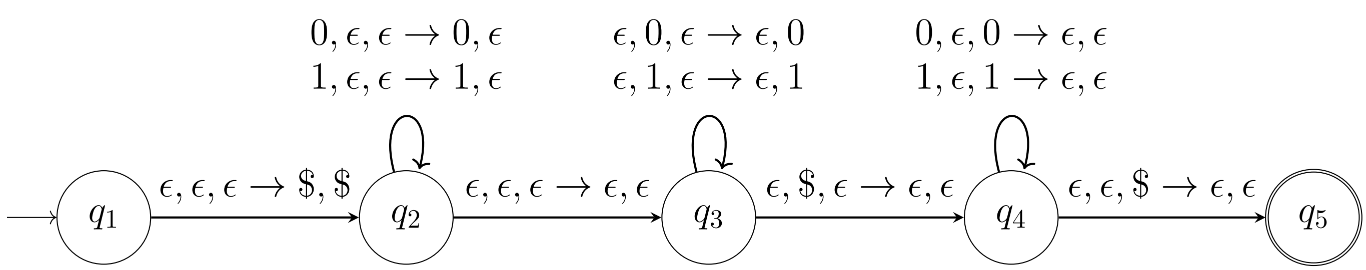

Figure 2: PDA for Q2

Basically, in q_3, we pop all elements from the first stack to the second, essentially simulating a queue with two stacks. This works because the PDA has non-deterministic transition.

Question 3. Draw PDA for language $\{a^i b^j c^k | i, j, k \geq 0, i = j \text{ or } j = k\}$, and show they are equivalent. [1]

[1] Note that this is different from the pda in the lecture

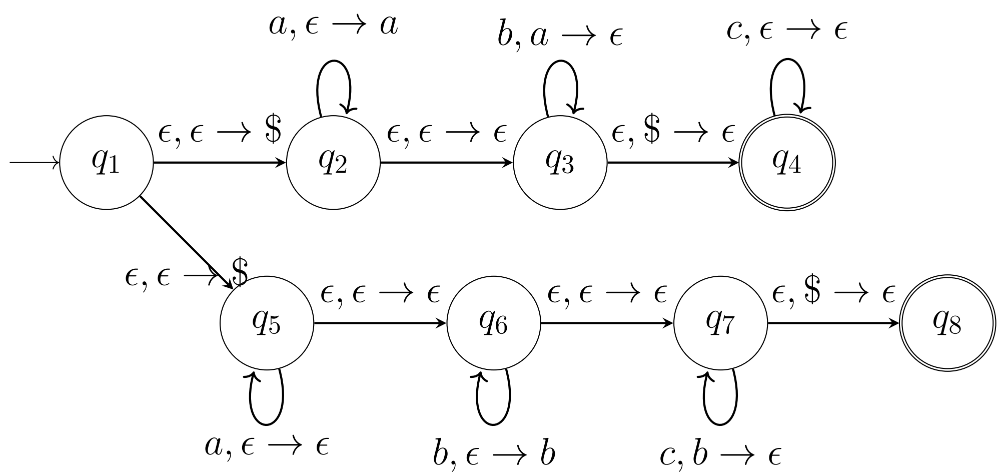

Figure 3: PDA for Q3

Solution 3. For every string in $\{a^ib^jc^k | i,j,k \geq 0, i = j \text{ or } j = k\}$, it is either in $\{a^ib^jc^k | i,j,k \geq 0, i = j\}$ or in $\{a^ib^jc^k | i,j,k \geq 0, j = k\}$. If it is the case $i = j$, it will pass through q_1, q_2, q_3, q_4. Start at q_1 ,ϵ transition moves it to q_2 where it counts number of a, and move to q_3 as $i = j$, it see b with equal frequency, and apply $\epsilon, \$ \to \epsilon$ to move to q_4, the only symbols left are c, so it stays at q_4 being accepted. If it is the case $j = k$, the machine moves to q_5, and process a without changing the stack, then in q_6 it counts b, and see the same number of c as b in q_7, and move to q_8 the accepting state.

For every string being accepted by this PDA, it either enters q_4 or q_8. If it enters q_4, after pop the \$ it only sees c. Then at q_3 say it invoked $\{b, a \to \epsilon\}$ j times corresponding to number of b, at q_2, $\{a, \epsilon \to a\}$ has to be invoked at least j times. However, as q_3 sees \$ after j bs, $\{a, \epsilon \to a\}$ has to be invoked exactly j times. So, we have equal number of as and bs. Therefore, the string would be in $\{a^ib^jc^k | i,j,k \geq 0, i = j\}$.

Similarly, if it enters q_8, it would be in $\{a^ib^jc^k | i,j,k \geq 0, j = k\}$. In all, if it is recognized by this PDA, it has to be in $\{a^ib^jc^k | i,j,k \geq 0, i = j \text{ or } j = k\}$.

Question 4 (Optional)**.** Given the CFG $G = (V, \Sigma, R, S)$:

- V = {S, NP,VP,Det,Nominal,Noun,PP,Preposition,Verb}
- Σ = The, spy, saw, cop, with, a , telescope

- Rules

  S $\rightarrow$ NP VP

  NP $\rightarrow$ Det Nominal

  Nominal$\rightarrow$ Noun $\|$ Nominal PP

  VP$\rightarrow$ VP PP$\|$ Verb NP

  PP $\rightarrow$ Preposition NP

  Det $\rightarrow$ The $\|$ a

  Noun $\rightarrow$ spy $\|$ cop$\|$telescope

  Verb $\rightarrow$ saw

  Preposition $\rightarrow$ with

Is there a third derivation other than what we found in the lecture for The spy saw a cop with a telescope ?

Solution 4. No, there isn't. The interesting thing is that one might interpret the sentence with the meaning where The spy is with a telescope at hand, but he did not use the telescope to see the cop. However, the restricted grammar structure of CFG will not allow this interpretation. Let's stop here before we sink into the rabbit hole of syntaxsemantics interface. To prove there is no other parse requires some understanding of CFG parsing being solvable by dynamic programming. One can refer to the NLP textbook that explains the CYK algorithm. Here is some code that find all parses.

Question 5. Complete the proof for the Kleene closure property of CFL.

Solution 5. For L_1 being context-free, there exists a corresponding $G_1 = (V_1, \Sigma_1, R_1, S_1)$. Let $G_2 = (V_1, \Sigma_1, R_1 \cup \{S \rightarrow S_1 S | \epsilon\}, S)$. We claim $L(G_2) = L_1^*$, as G_2 is context free, we will have the Kleene closure property of CFL. For any $w \in L(G_2)$, except ϵ which is in Kleene star, all strings are multiple consecutive realization of S_1 through the original production rule. Therefore, $w \in L_1^*$. If it is in Kleene star of L_1, say it consists of k realization of strings from L_1, we apply $S \rightarrow S_1 S$ k times and for all copies there will be production rules that matches all fragments. $k = 0$ corresponds to $S \rightarrow \epsilon$ production.

Question 6. Show that context free languages are closed under intersection if they are closed under complement.

Solution 6. Assume complement closure, that is $\forall L, \overline{L} \in CFL$. For any $L_1, L_2 \in CFL$, we have

$$L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}} \quad (1)$$

$$= \overline{\overline{L_1} \cup \overline{L_2}} \quad (2)$$

Due to the complement closure, we have $\overline{L_1}, \overline{L_2} \in CFL$. With the union closure we proved in the lecture $\overline{L_1} \cup \overline{L_2} \in CFL$, and we apply complement closure again, we have $L_1 \cap L_2 \in CFL$.

Based on what I know, a direct proof cannot be given for the other way around, and we have to use the pumping lemma (next lecture) to show that they are both false.