Int 201: Decision Computation and Language
Tutorial 8 Solution

Dr. Chunchuan Lyu

November 13, 2025

Question 1. Show that all languages recognized by a regular PDA can be recognized by a PDA that has only one accepting state.

Solution 1. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA with multiple accepting states. We construct a new PDA $M' = (Q \cup q_f, \Sigma, \Gamma, \delta', q_0, q_f)$ where q_f is a new state and:

δ' contains all transitions in δ For each state $q \in F$, add transition $\delta'(q, \epsilon, \epsilon) = (q_f, \epsilon)$ Proof of equivalence:

$L(M) \subseteq L(M')$: If $w \in L(M)$, then M reaches some accepting state $q \in F$ after reading w. In M', after reaching q, we can use the added ϵ-transition to reach q_f. Thus $w \in L(M')$.

$L(M') \subseteq L(M)$: If $w \in L(M')$, then M' reaches q_f after reading w. By construction, this is only possible by first reaching some state in F and then using an ϵ-transition. Therefore, $w \in L(M)$.

Question 2. Show that all languages recognized by a regular PDA can be recognized by a PDA that accepts when both its' stack is empty and its' in an accepting state.

Solution 2. With the previous lemma, without loss of generality, we assume that there is only one terminal start q_0.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F = \{q_a\})$ be a PDA accepting by final state. We construct a new PDA $M' = (Q', \Sigma, \Gamma \cup \$, \delta', q_{00}, \{q_{a2}\})$ where:

$Q' = Q \cup \{q_{00}, q_{a1}, q_{a2}\}$

The transitions δ' include:

Initial transition: $\delta'(q_{00}, \epsilon, \epsilon) = (q_0, \$)$ All transitions from M For each $f \in F$: $\delta'(f, \epsilon, \epsilon) = (q_{a1}, \epsilon)$ Clean-up transitions: $\delta'(q_{a1}, \epsilon, \gamma) = (q_{a1}, \epsilon)$ for all $\gamma \in \Gamma$ $\delta'(q_{a1}, \epsilon, \$) = (q_{a2}, \epsilon)$ Proof of equivalence:

$L(M) \subseteq L(M')$: Let $w \in L(M)$ In M', we first push \$ The computation proceeds as in M (not touching \$) Upon reaching an accepting state in F, we can move to q_{a1} In q_{a1}, pop all symbols until \$ Finally, pop \$ and enter q_{a2} with empty stack

$L(M') \subseteq L(M)$: Let $w \in L(M')$ To reach q_{a2} with empty stack, the computation must: Start with pushing \$ Reach some $f \in F$ without popping \$ Pop everything including \$ This means w is accepted by M through state f

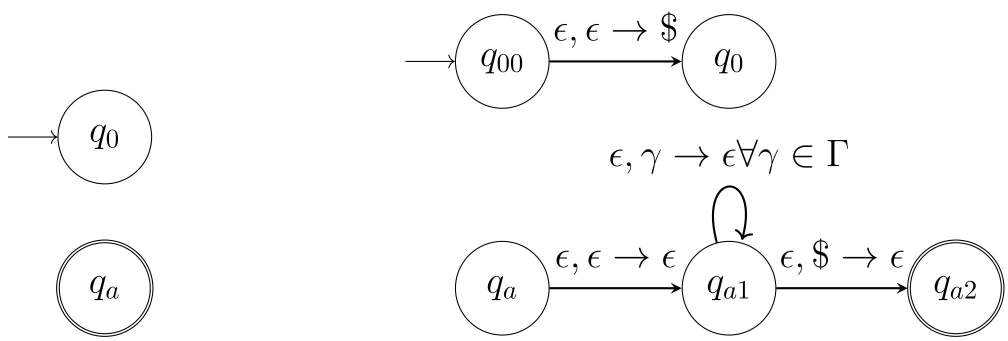

Figure 1: Left side old PDA M; right side new PDA M' accepting in empty stack

Question 3. Show that all languages recognized by a regular PDA can be recognized by a PDA where all transitions either push or pop the stack but not do both.

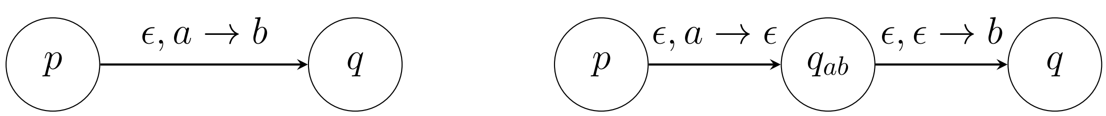

Figure 2: Left side old PDA; right side PDA that don't push and pop at the same time

Solution 3. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. We construct $M' = (Q', \Sigma, \Gamma, \delta', q_0, F)$ where:

$Q' = Q \cup q_{ab} | \exists p, q \in Q,$ transition from p to q that both pops and pushes

For each transition in δ that both pops and pushes:

If $\delta(p, x, a) = (q, b)$ (pops a and pushes b), replace with: $\delta'(p, x, a) = (q_{ab}, \epsilon)$ (pop only) $\delta'(q_{ab}, \epsilon, \epsilon) = (q, b)$ (push only) All other transitions remain unchanged Proof of equivalence:

Each computation step in M that both pops and pushes is replaced by two steps in M' The intermediate state q_{ab} has exactly one outgoing transition At any point, the stack contents in M' match those in M except during intermediate states Therefore, $L(M) = L(M')$

Question 4. Show that the language $\{ww|w \in \{0,1\}^*\}$ is not context-free by using the pumping lemma.

Solution 4. Proof by contradiction: Assume $\{ww|w \in \{0,1\}^*\}$ is CFL, pumping lemma says that there is a pumping length p. Take $s = 0^p1^p0^p1^p$, clearly s is in the language, and $|s| > p$. The pumping lemma which states $\exists$ decomposition $s = uvxyz$ satisfying:

- $|vxy| \leq p$
- $|vy| \geq 1$
- $\forall i \geq 0$, $uv^ixy^iz \in L$

Let us show none of the decomposition of $s = uvxyz$ will work. The key is to rely on $|vxy| \leq p$.

Case 1: if vxy lies entirely on one contiguous segments of 1s or 0s, the pumping will destroy the symmetry.

Case 2: if

- $u = 0^{p-m}$
- $vxy = 0^m1^n$
- $z = 1^{p-n}0^p1^p$

for $m + n \leq p$.

If we are going to pump down to uxz, we either lose some 1 or lose some 0 on the first half, and we lost the symmetry.

Case 3: we must have it on the middle:

- $u = 0^p1^{p-m}$

- $vxy = 1^m 0^n$
- $z = 0^{p-n} 1^p$

for $m + n \leq p$.

If we are going to pump down to uxz, for the middle part, we either lose some 1 or lose some 0, as $|vz| \geq 1$. In other words, we end up with $uxz = 0^p 1^i 0^j 1^p$, where either $i < p$ or $j < p$. So, we must either mismatch the 0s or 1s, and hence break the symmetry. So, none of the splitting works and we get a contradiction.

Question 5 (Optional*)**.** A deterministic PDA have deterministic transition function instead of relations. Can the palindrome language $\{ww^R | w \in \{0,1\}^*\}$ be recognized by a deterministic PDA? (no need for a proof)

Solution 5. No, the language $ww^R | w \in 0, 1^*$ cannot be recognized by a deterministic PDA.

Key insight: To verify ww^R, a PDA must:

Store w in the stack Determine the middle point of the input Compare the remainder of input with stack contents A DPDA cannot reliably determine the middle point since it has only one possible action for each configuration. In contrast, an NPDA can non-deterministically guess the middle point and verify correctness through multiple computation paths.

Reference: A proof can be found in Martin's "Introduction to Languages and the Theory of Computation" (Theorem 5.16).

Question 6 (Optional*)**.** Convert this PDA to CFG.

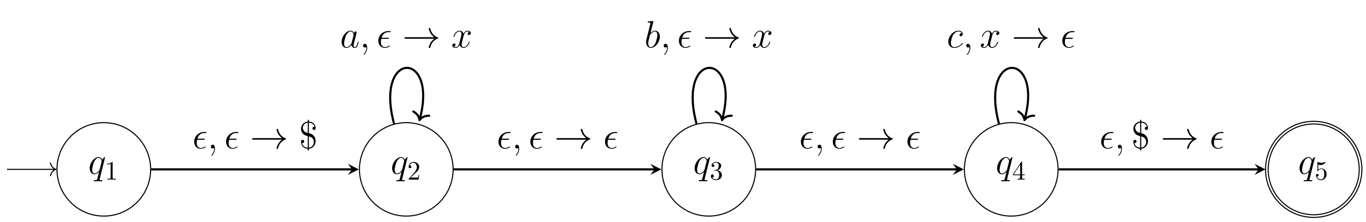

Figure 3: PDA for Q6

Solution 6. In theory, run the conversion algorithm in full will get the job done. However, many of the transition A_{ij} can only generate ϵ. So here, we list only the symbols that are useful.$S = A_{15}$

- $\forall i, A_{ii} \to \epsilon$
- $A_{15} \to A_{24}$ Matching $\epsilon, \epsilon \to \$$ and $\epsilon, \$ \to \epsilon$
- $A_{24} \to aA_{24}c$ Matching $a, \epsilon \to x$ and $c, x \to \epsilon$
- $A_{24} \to A_{23}A_{34}$ Composition Rule
- $A_{23} \to A_{33}$ All strings that can take PDA from 3 to 3 with empty stack can take it from 2 to 3. (Also, consider that there is always ϵ transition in each node , so apply matching rule)
- $A_{34} \to bA_{34}c|A_{44}$ For the same reason as above.

Or we understand this PDA generates $\{a^n b^m c^{m+n} | m, n \geq 0\}$, and write

- $S \to aSc|B$
- $B \to bBc|\epsilon$

or for better comparison, we write $S = A_{24}$

- $A_{24} \to aA_{24}c|A_{34}$
- $A_{34} \to bA_{34}c|\epsilon$

Question 7 (Optional*)**.** Have fun with the notebook. https://github.com/ND-CSE-30151/spring-2024/blob/main/notes

13, 15, 16