CSE-233 : Section A

Summer 2020

Pushdown Automata (PDA)

Reference:

Book2 Chapter 2

Md. Saidul Hoque Anik

anik@cse.uiu.ac.bd

Finite Automata vs Regular Expression

What's the difference?

Finite Automata vs. Regular Expression

- Regular Expression describes a language (Specification of a language)
- Finite Automata detects if a string is in the language or not (Recognizing mechanism)

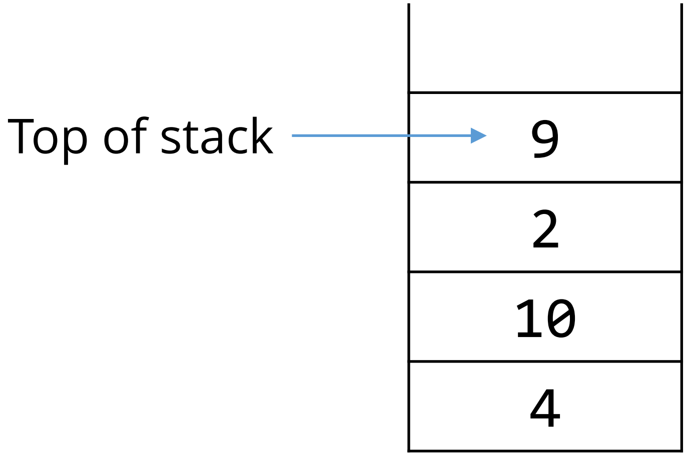
? vs. Context Free Grammar

- CFG describes a regular/non-regular language (Specification of a language)
- ? detects if a string is in the language or not (Recognizing mechanism)

Pushdown Automata vs. Context Free Grammar

- CFG describes a regular/non-regular language (Specification of a language)
- PDA detects if a string is in the language or not (Recognizing mechanism)

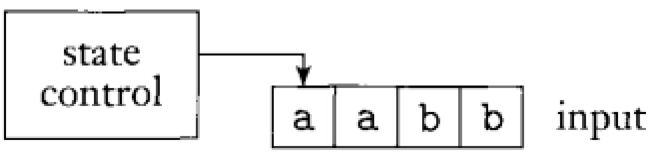
Pushdown Automata is nothing but an NFA with a stack (to push-down or pop data). The stack doesn't have memory limit.



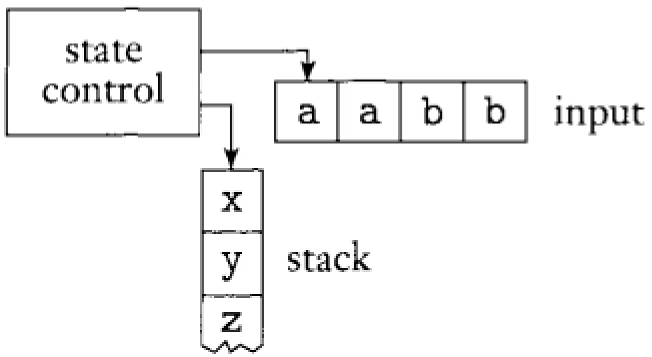
Detecting non-regular language using PDA

How can we keep count of 0 so that we can compare it with 1?

$$L = \{0^n1^n \mid n \geq 0\}$$



FA



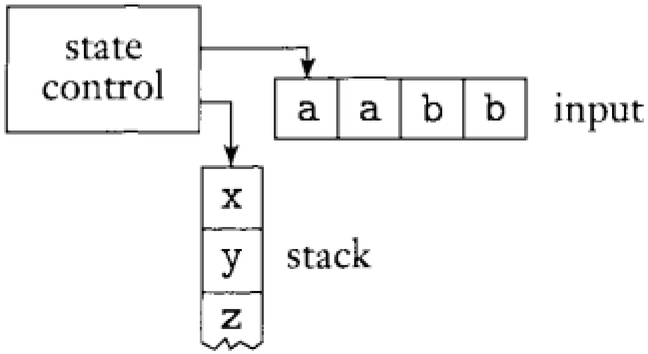
PDA

Detecting non-regular language using PDA

How can we keep count of 0 so that we can compare it with 1?

$$L = \{0^n 1^n \mid n \geq 0\}$$

1. From the start state, take input of 0
2. and push it in stack
3. When we get a 1, we move to the next state and pop the stack to see if it has a 0
4. In the next state while getting 1, we pop 0 from stack.
5. If the stack becomes empty and all input are 1, we reach a final state
6. We don't provide any other transition arrows to final state, so any other case will not reach final state.



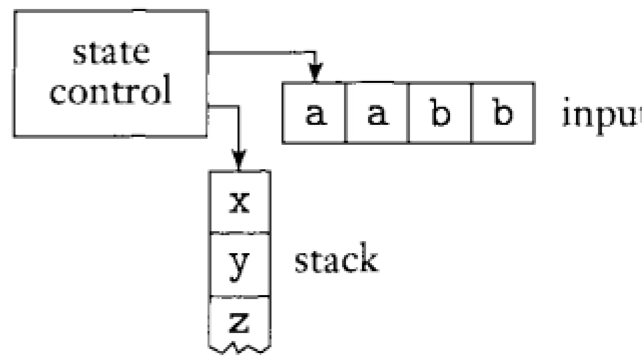
PDA

How do we know if there's more 0s in the stack or not?

How can we keep count of 0 so that we can compare it with 1?

$$L = \{0^n1^n \mid n \geq 0\}$$

1. From the start state, take input of 0
2. and push it in stack
3. When we get a 1, we move to the next state and pop the stack to see if it has a 0
4. In the next state while getting 1, we pop 0 from stack.
5. If the stack becomes empty and all input are 1, we reach a final state
6. We don't provide any other transition arrows to final state, so any other case will not reach final state.



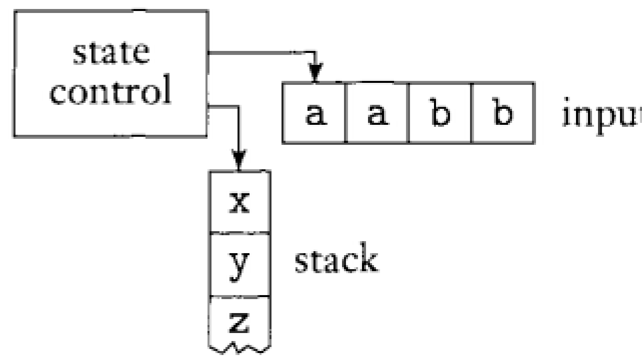
PDA

How do we know if there's more 0s in the stack or not?

How can we keep count of 0 so that we can compare it with 1?

$$L = \{0^n1^n \mid n \geq 0\}$$

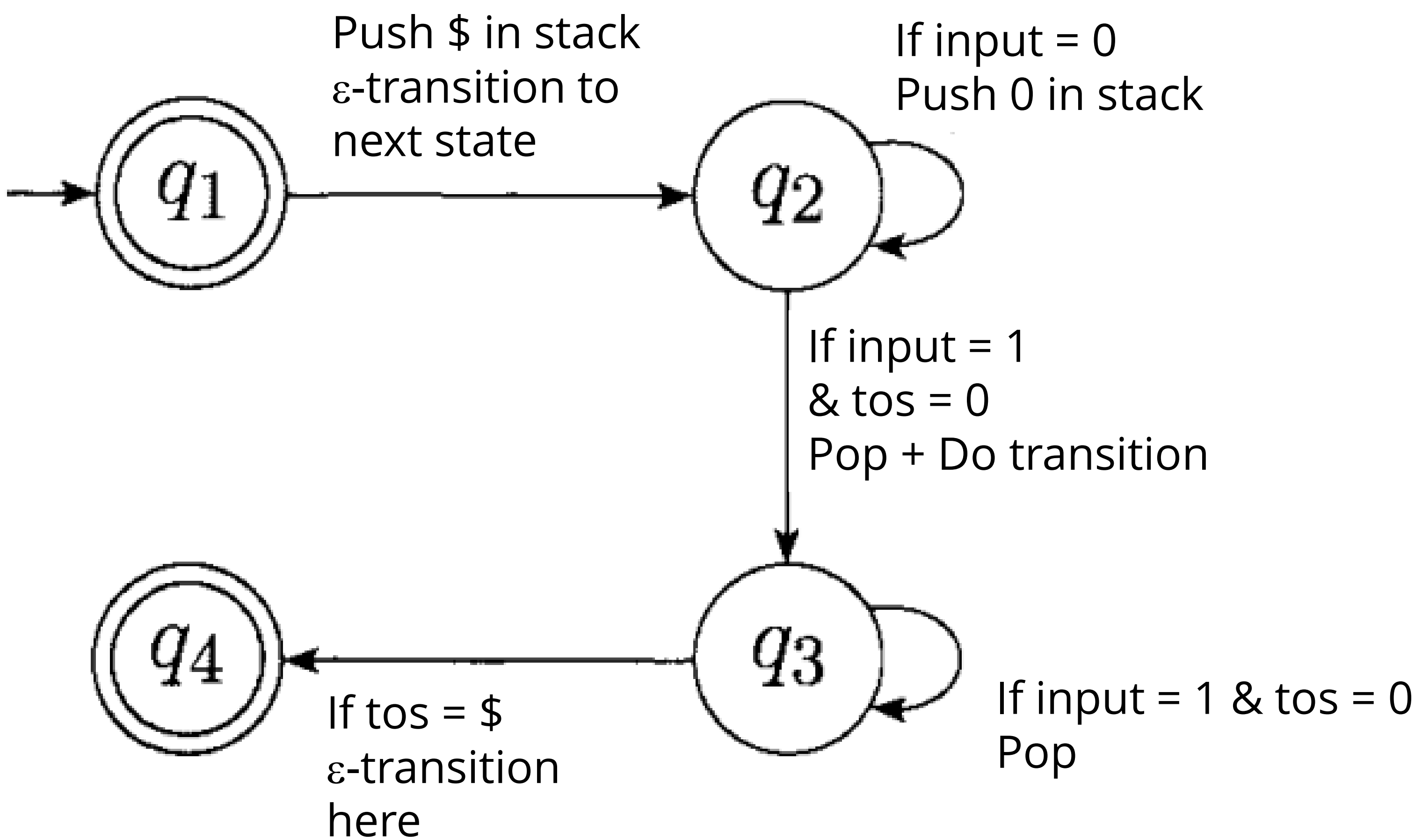
1. From the start state, push an arbitrary symbol \$ denoting the top of stack and take input of 0
2. and push it in stack
3. When we get a 1, we move to the next state and pop the stack to see if it has a 0
4. In the next state while getting 1, we pop 0 from stack.
5. If the stack top is \$ and all input are 1, we reach a final state
6. We don't provide any other transition arrows to final state, so any other case will not reach final state.



PDA

Informal PDA

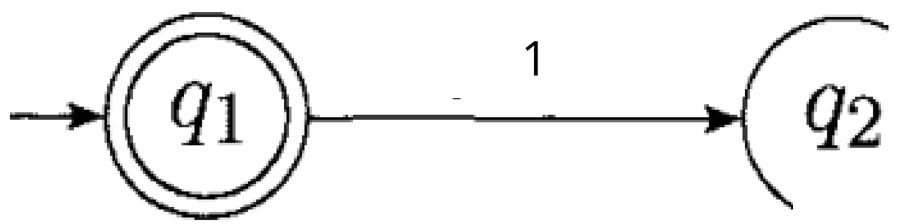
How can we keep count of 0 so that we can compare it with 1?



*tos = Top of stack

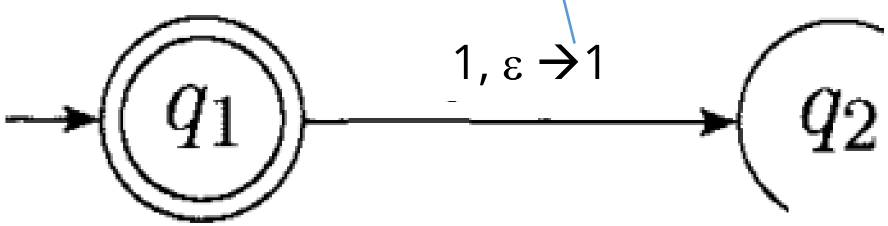
Writing Convention

For NFA we used to write like this



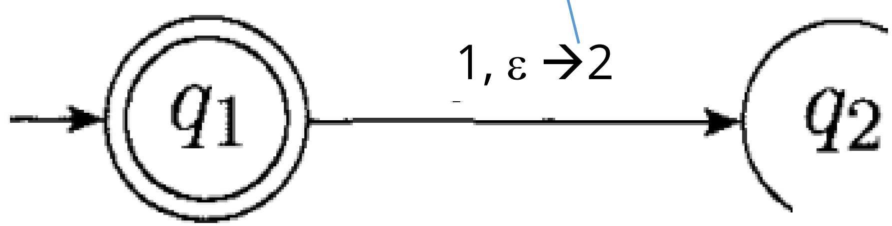
Writing Convention

If input == 1, Push 1 in stack and go to q2



Writing Convention

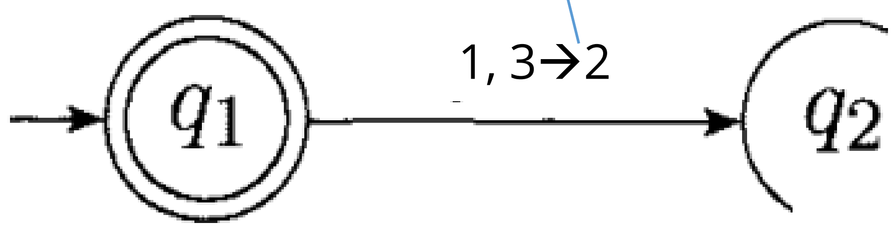
If input == 1, Push 2 in stack and go to q2



Writing Convention

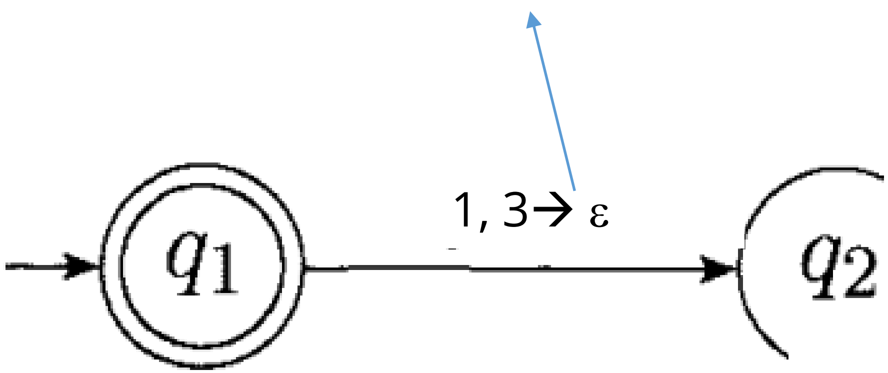
If input == 1 & tos == 3,

Pop 3, Push 2 in stack and go to q2



Writing Convention

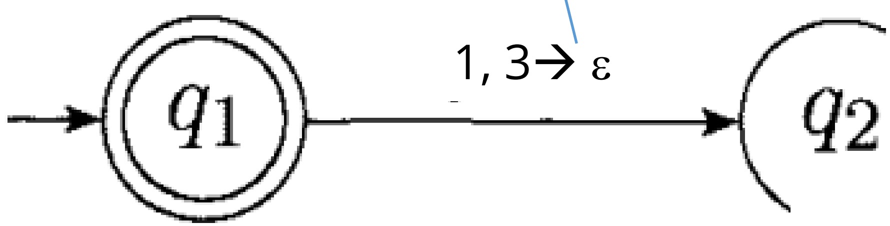
What will this mean?



Writing Convention

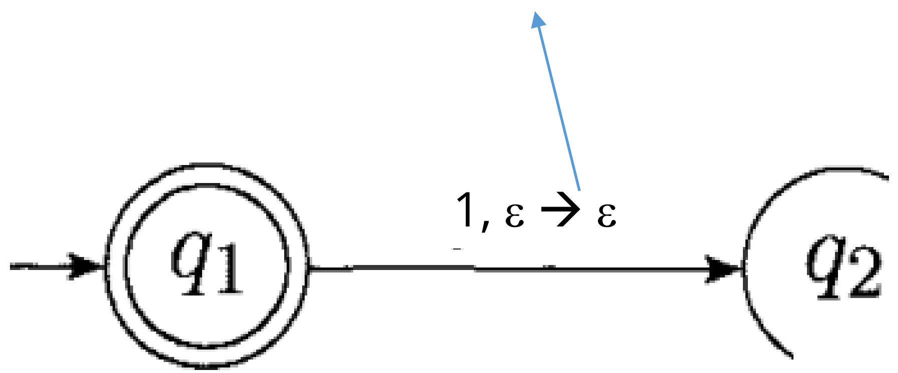
If input == 1 & tos == 3,

Pop 3 from stack, Push nothing, and go to q2



Writing Convention

What does it mean?

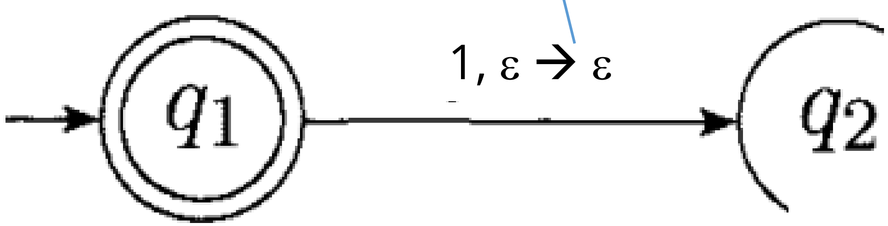


Writing Convention

If input == 1

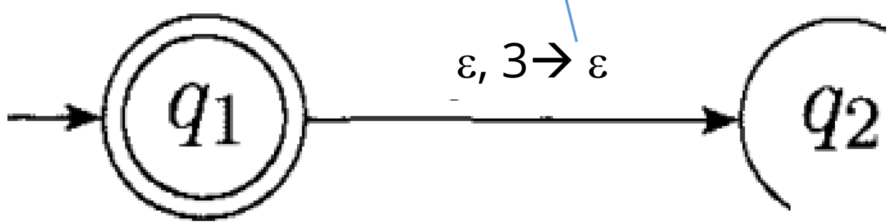
Then go to q2

No need to check stack



Writing Convention

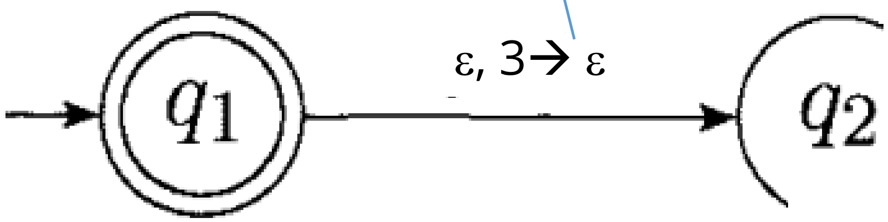
What does this mean?



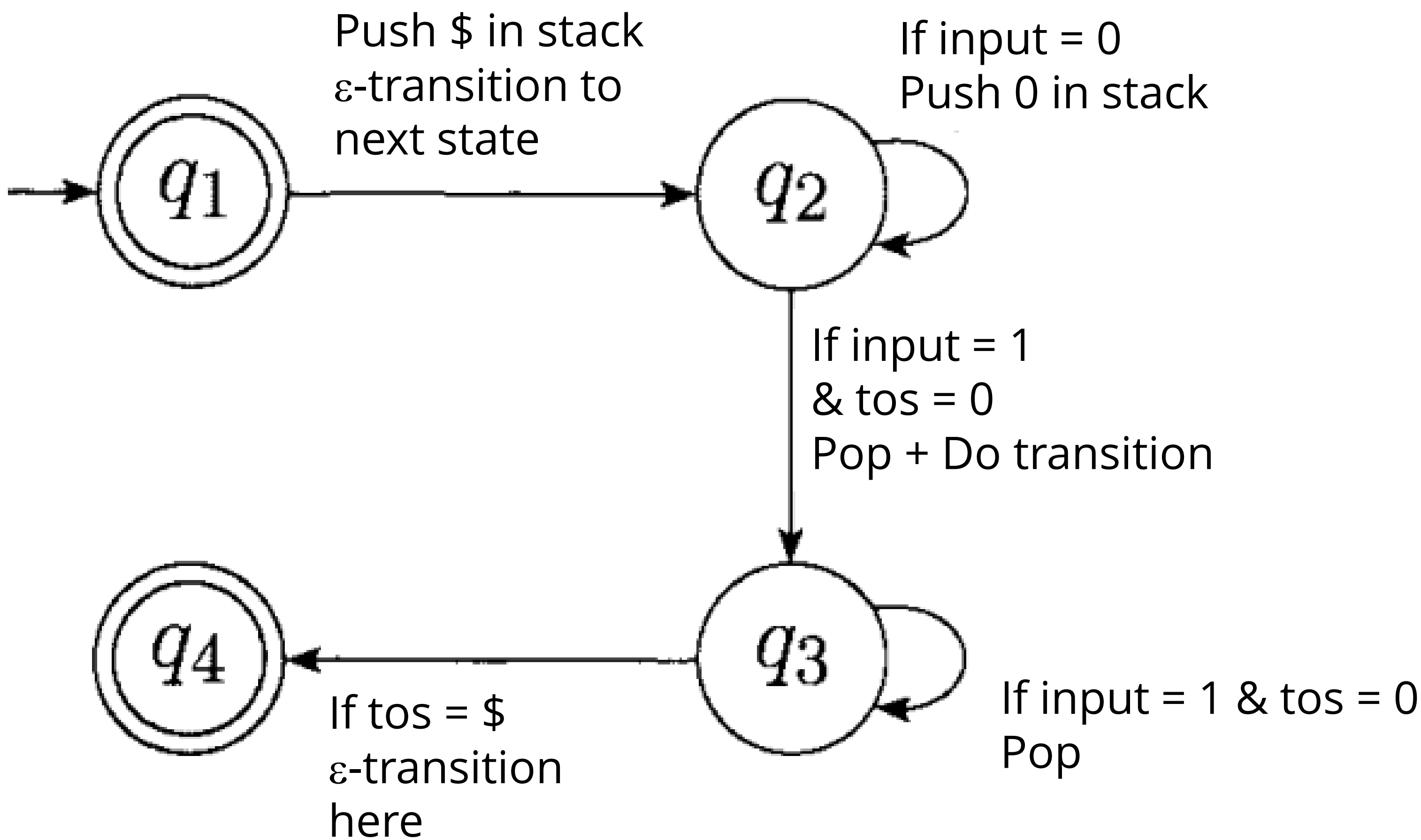
Writing Convention

If you reach q1 & tos == 3,

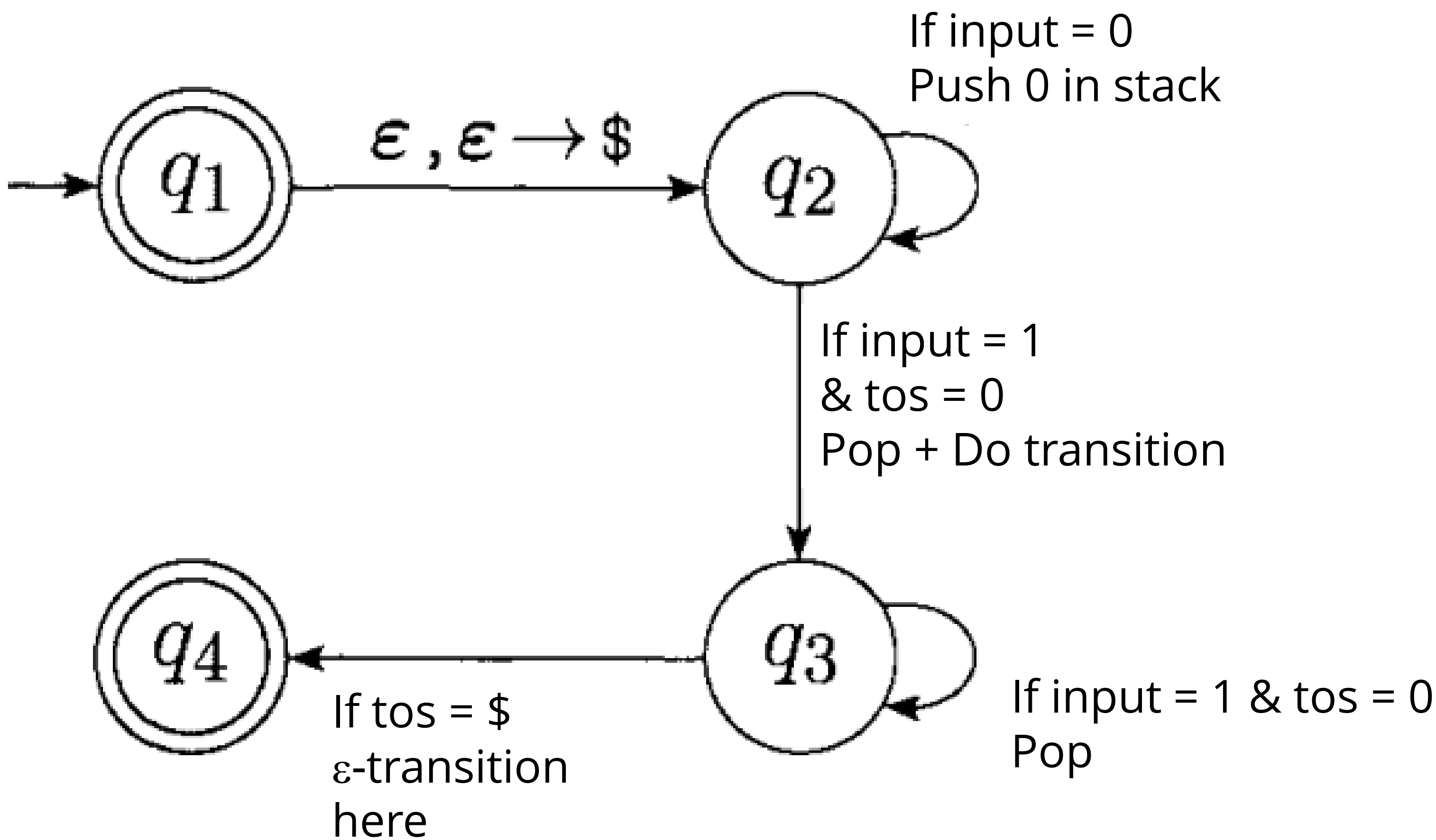
Pop 3 from stack, Push nothing, and go to q2



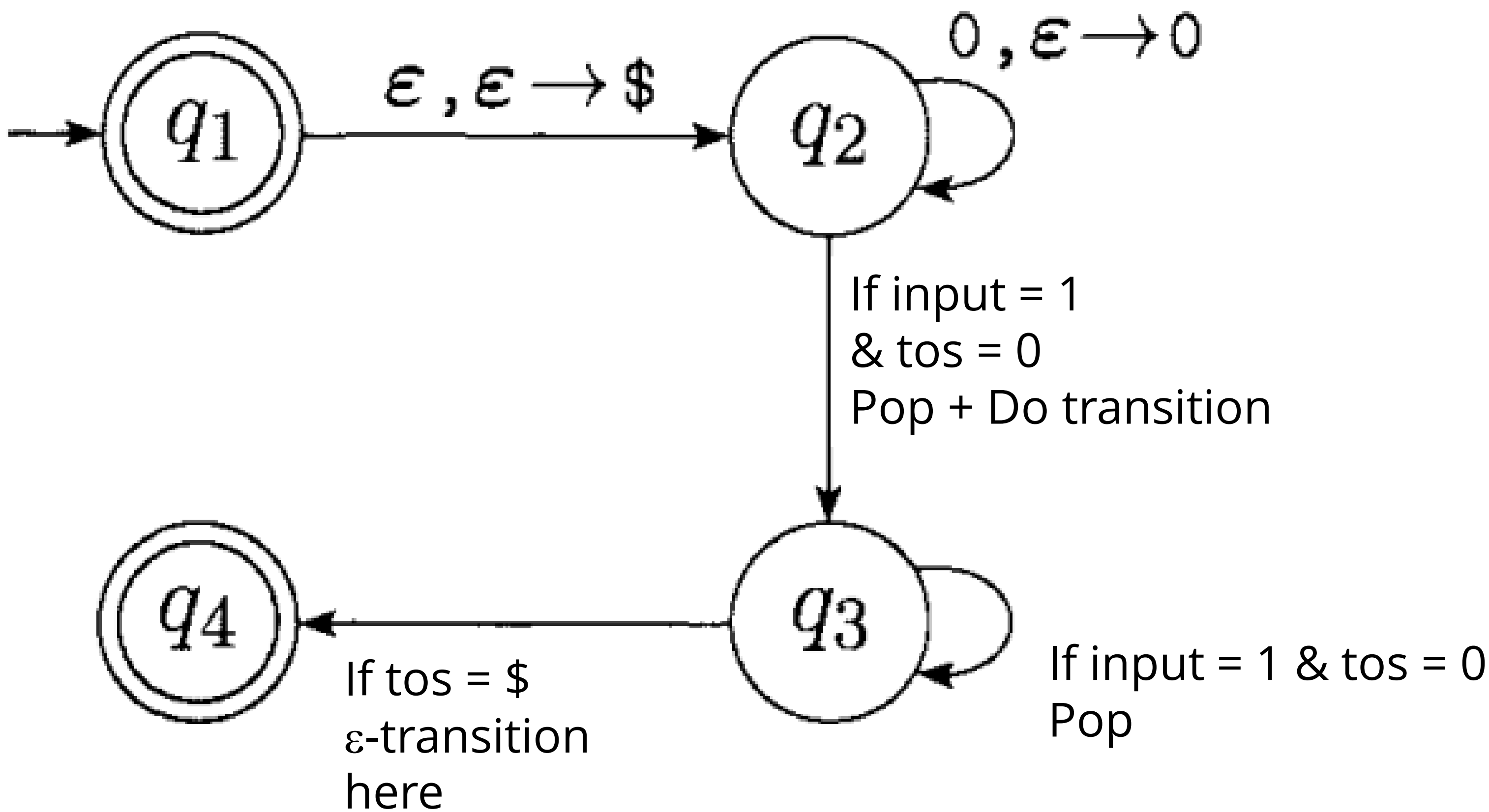
Can you write the following PDA formally?



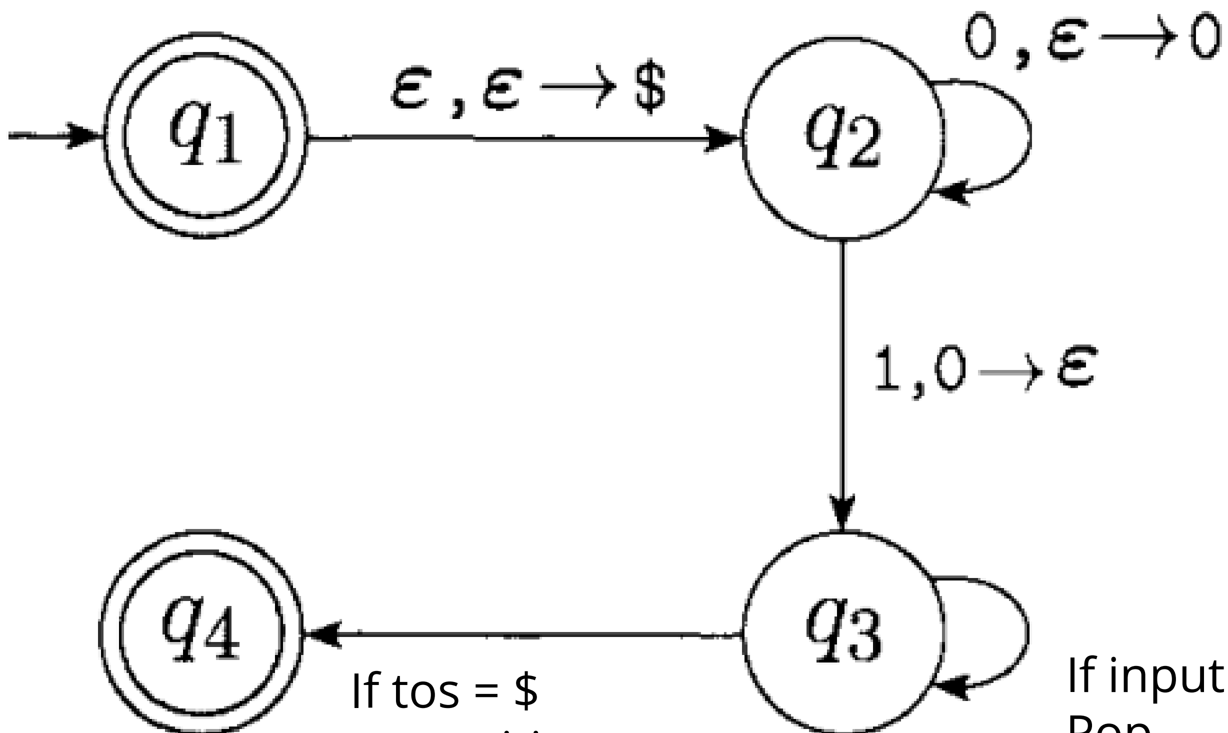
Can you write the following PDA formally?



Can you write the following PDA formally?



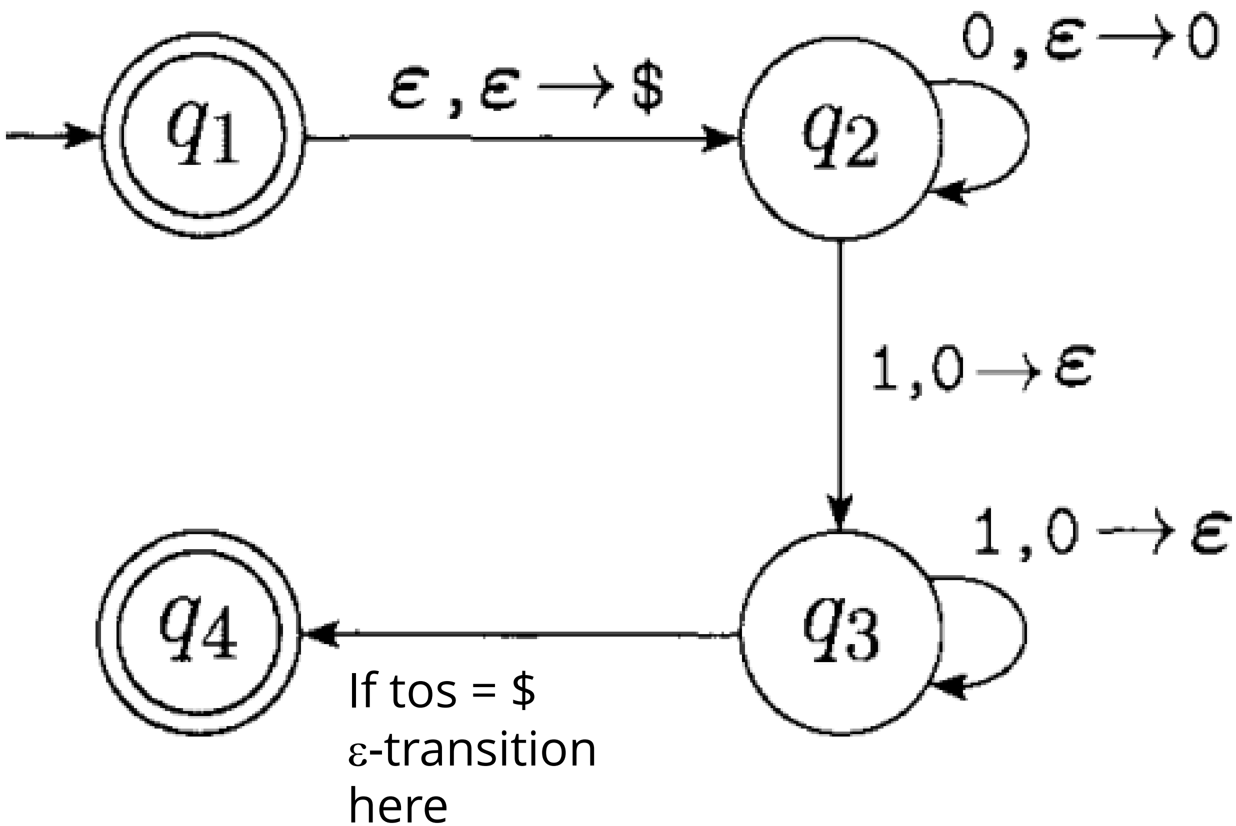
Can you write the following PDA formally?



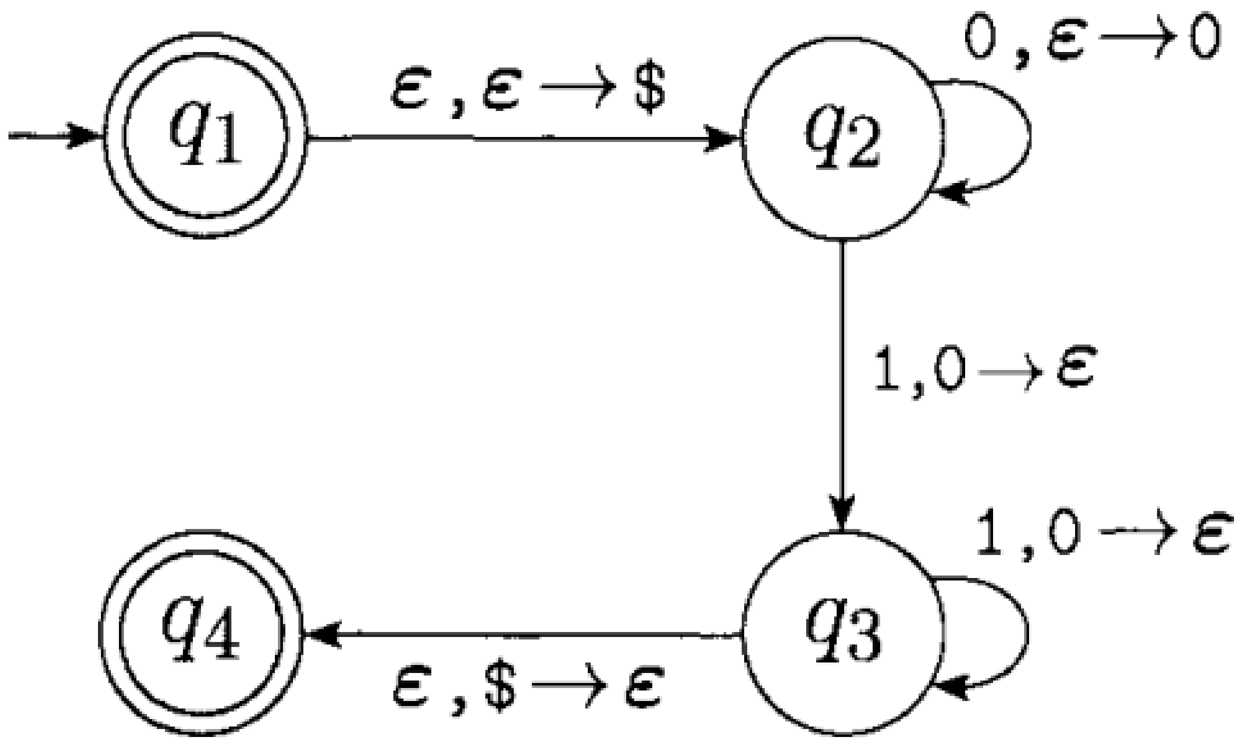
If tos = \$
ε-transition
here

If input = 1 & tos = 0
Pop

Can you write the following PDA formally?



Can you write the following PDA formally?

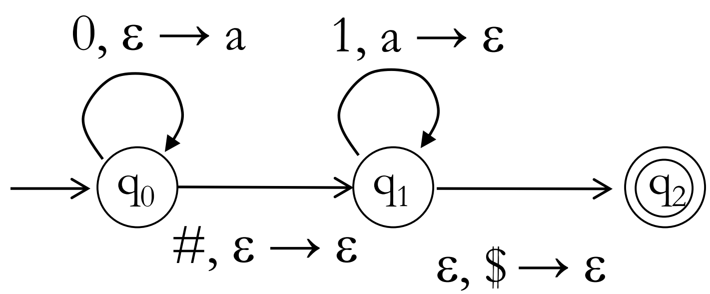


Task

Design Pushdown Automata for the following languages-

1. $a^n \# b^n$; $n \geq 1$
2. $a^n b^n c^m \mid n, m \geq 1$
3. $a^n b^m c^n \mid n, m \geq 1$
4. $a^{m+n} b^m c^n \mid n, m \geq 1$

What does this PDA do?



Formal Definition of PDA

The definition can slightly vary depending on context.

A ***pushdown automaton*** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q, Σ, Γ, and F are all finite sets, and

1. Q is the set of states,
2. Σ is the input alphabet,
3. Γ is the stack alphabet,
4. $\delta \colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

Example

The following is the formal description of the PDA (page 110) that recognizes the language $\{0^n1^n \mid n \geq 0\}$. Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q =$

$\Sigma =$

$\Gamma =$

$F =$

Example

The following is the formal description of the PDA (page 110) that recognizes the language $\{0^n1^n | n \geq 0\}$. Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$,

$\Sigma =$

$\Gamma =$

$F =$

Example

The following is the formal description of the PDA (page 110) that recognizes the language $\{0^n1^n | n \geq 0\}$. Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$$Q = \{q_1, q_2, q_3, q_4\},$$

$$\Sigma = \{0,1\},$$

$$\Gamma =$$

$$F =$$

Example

The following is the formal description of the PDA (page 110) that recognizes the language $\{0^n1^n | n \geq 0\}$. Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$,

$\Sigma = \{0,1\}$,

$\Gamma = \{0, \$\}$,

$F =$

Example

The following is the formal description of the PDA (page 110) that recognizes the language $\{0^n 1^n | n \geq 0\}$. Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

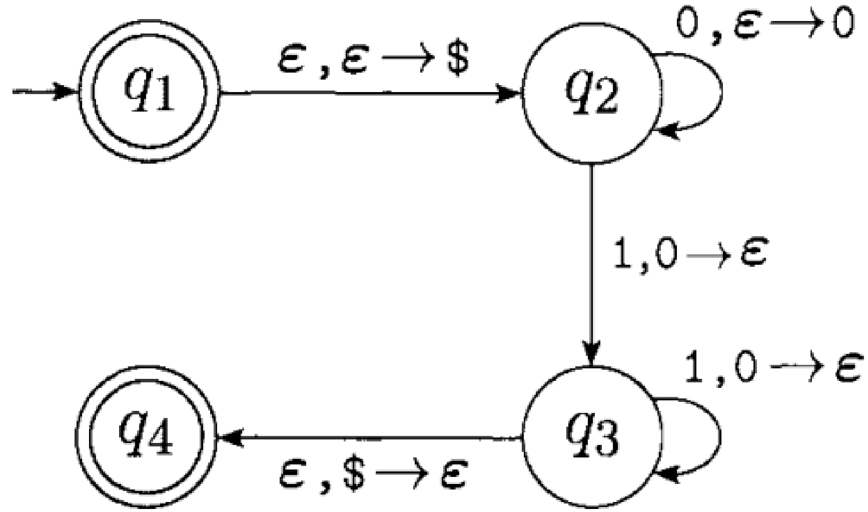
$Q = \{q_1, q_2, q_3, q_4\}$,

$\Sigma = \{0,1\}$,

$\Gamma = \{0, \$\}$,

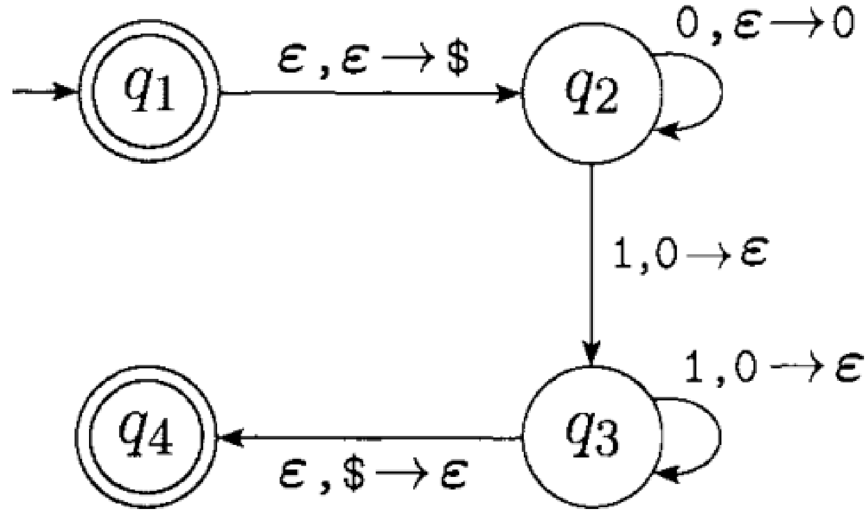
$F = \{q_1, q_4\}$, and

Example



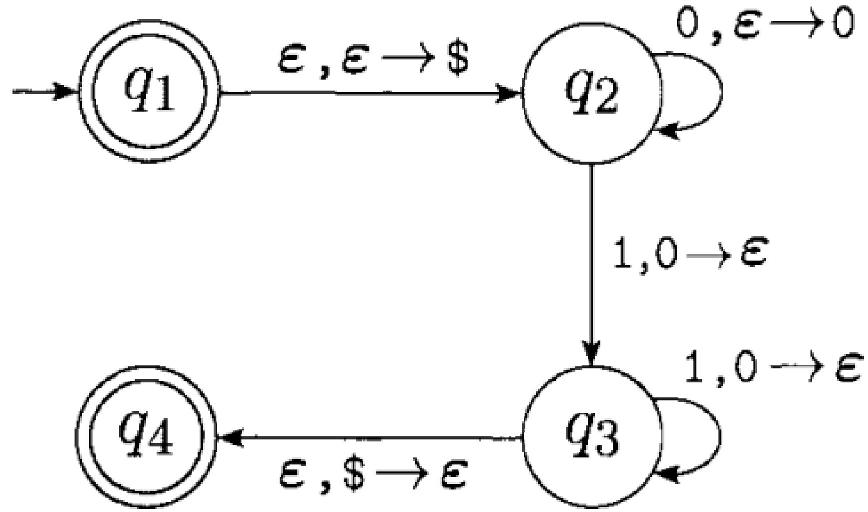
Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									
q_2									
q_3									
q_4									

Example



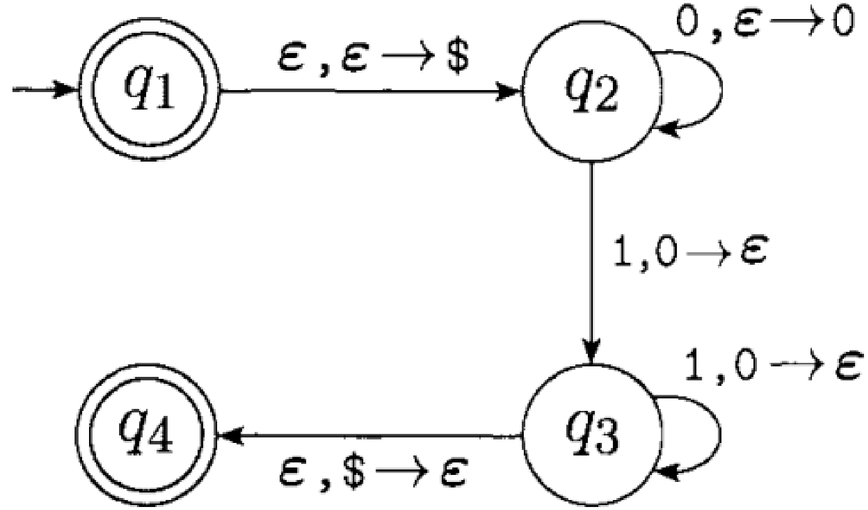
Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2									
q_3									
q_4									

Example



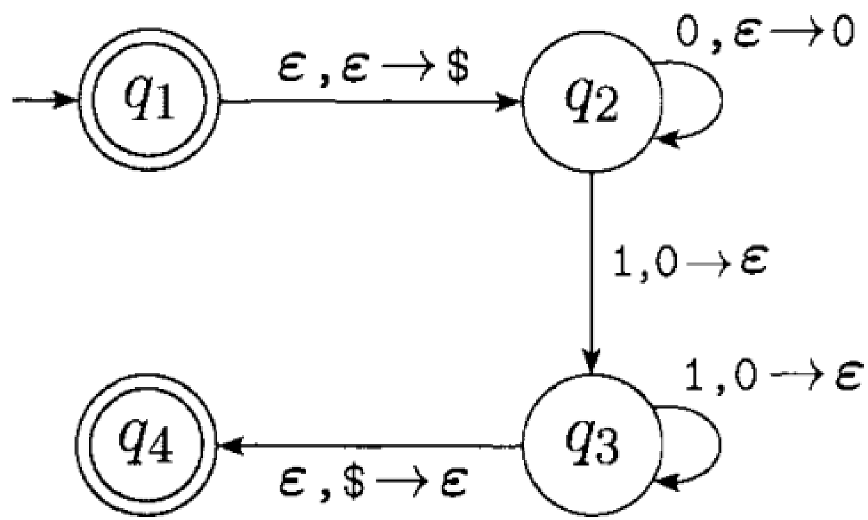
Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3									
q_4									

Example



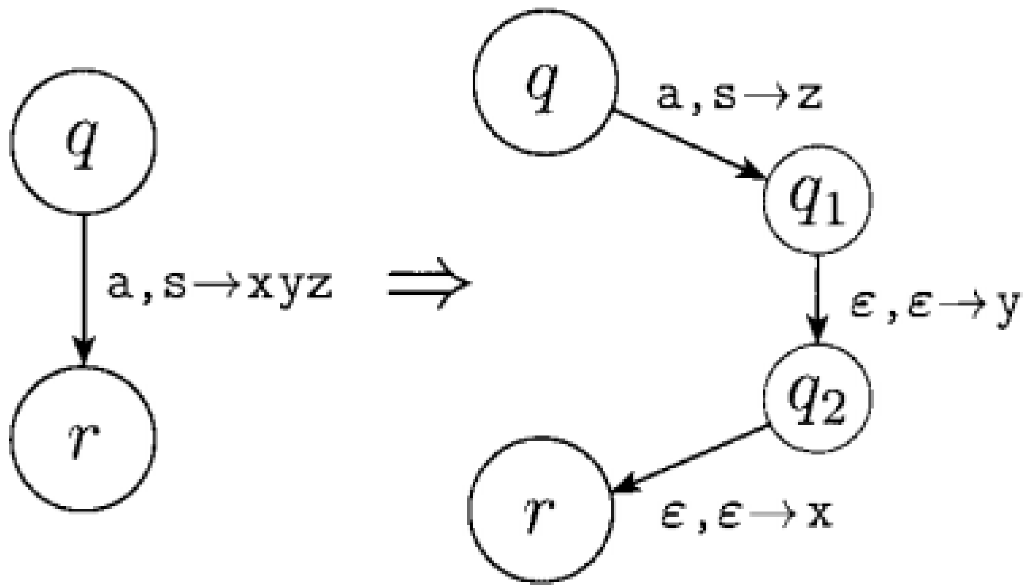
Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

Example



Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

Shorthand Notation



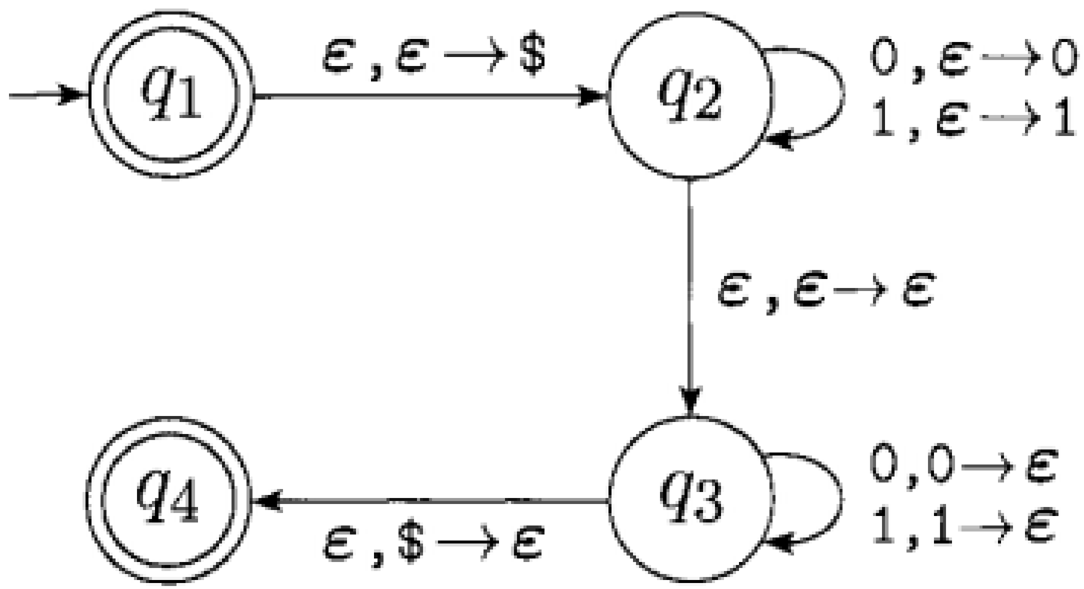
Implementing the shorthand $(r, xyz) \in \delta(q, a, s)$

Taking Advantage of Non-determinism

PDA M_3 that recognizes $\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$

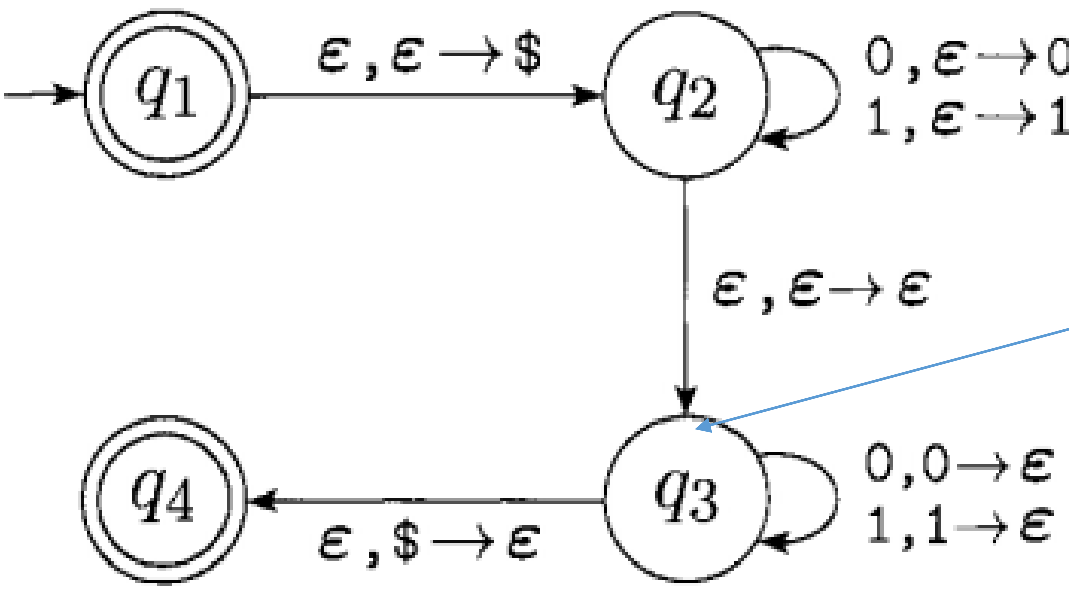
Taking Advantage of Non-determinism

PDA M_3 that recognizes $\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$



Important Note

PDA M_3 that recognizes $\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$



In every epsilon transition, the stack is also being copied along with the states.

This means that the popping of 0 and 1 in q3 has no effect in the stack of q2 as it's done in a copied dedicated stack for q3

Another Example

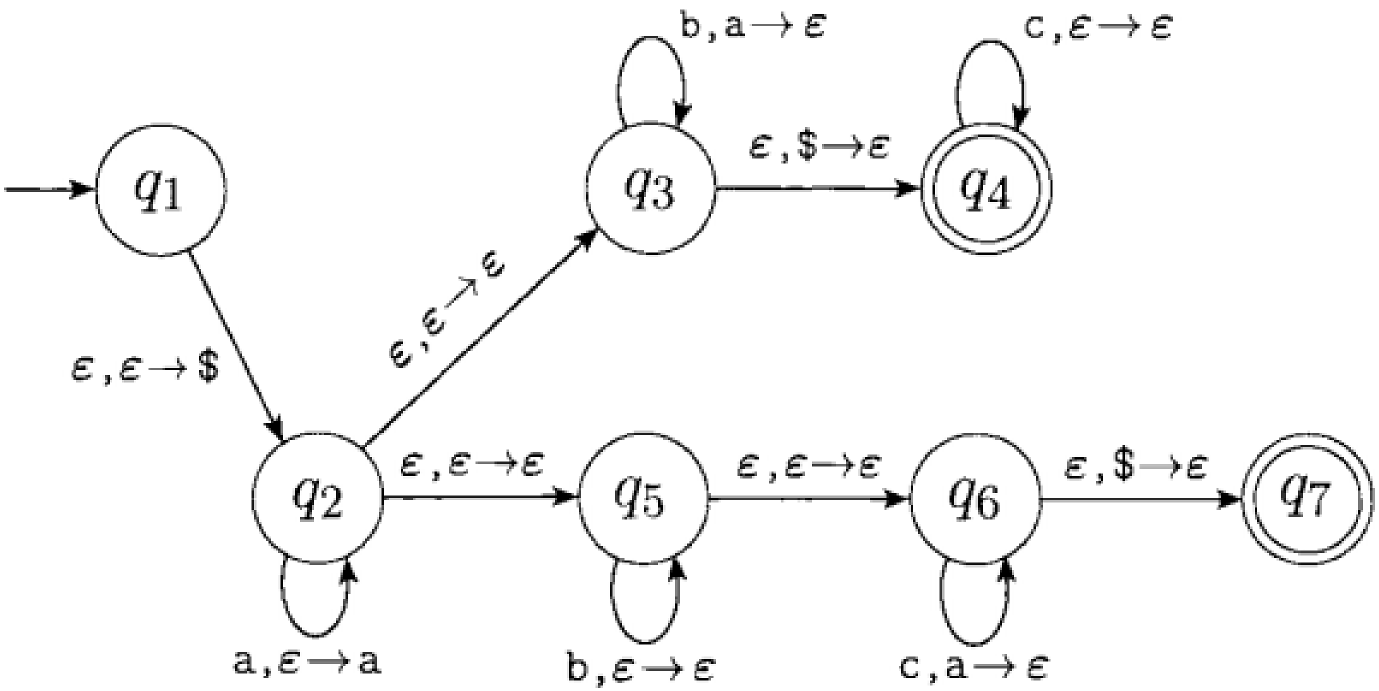
Design a PDA that recognizes the following language

$$\{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}.$$

Another Example

Design a PDA that recognizes the following language

$$\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}.$$



Practice

Describe PDAs for the following languages:

- $L = \{w\#w^R : w \in \{0, 1\}^*\}, \Sigma = \{0, 1, \#\}$
- $L = \{ww^R : w \in \Sigma^*\}, \Sigma = \{0, 1\}$
- $L = \{w : w$ has same number of 0s and 1s$\}, \Sigma = \{0, 1\}$
- $L = \{0^i 1^j : i \leq j \leq 2i\}, \Sigma = \{0, 1\}$

Practice

Draw the schematic diagram of-

- PDA for $a^n b^{2n} | \, n, m \geq 1.$
- PDA for odd palindrome*.
- PDA for $a^n b^{n+m} c^m | \, n, m \geq 1$ OR $a^n b^n b^m c^m | \, n, m \geq 1.$
- PDA for $a^n b^m c^{n+m} | \, n, m \geq 1$

*Palindrome of odd length (e.g. 1110111, abcdcba, etc)