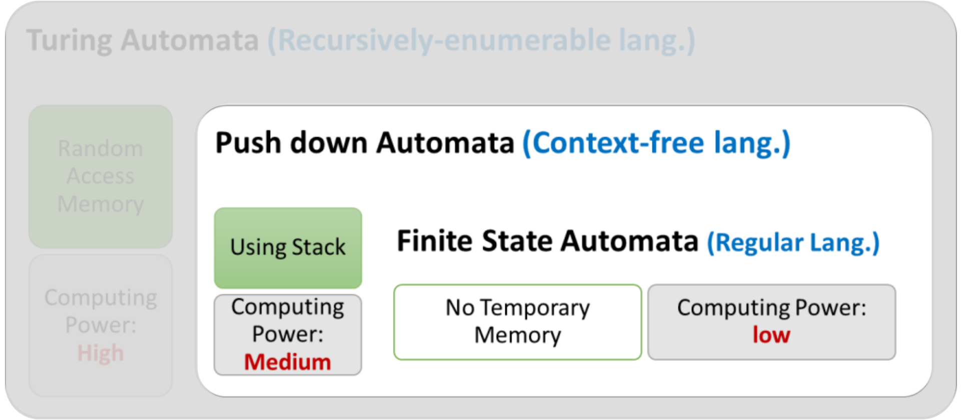
2.2 Pushdown Automata

PUSHDOWN AUTOMATA (PDA)

- PDAs are like nondeterministic finite automata but have an extra component called a stack.
  - provides additional memory beyond the finite amount available in the control.
  - allows pushdown automata to recognize some nonregular languages.



PUSHDOWN AUTOMATA (PDA)

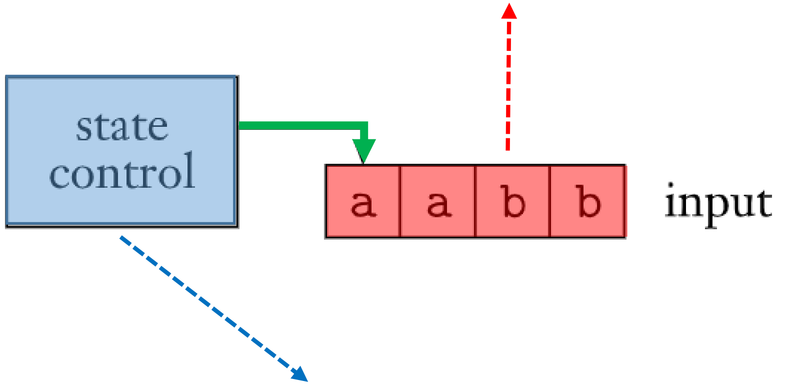
- Pushdown automata are equivalent in power to context-free grammars.
- A language is context free if
  - a **context-free grammar** generating it or
  - a **push-down automaton** recognizing it.

PUSHDOWN AUTOMATA (PDA)

- A PDA can write symbols on the stack and read them back later.
  - Pushing : writing a symbol on the stack
  - Popping : removing a symbol from the stack.
  - stack is a "last in, first out" storage device.

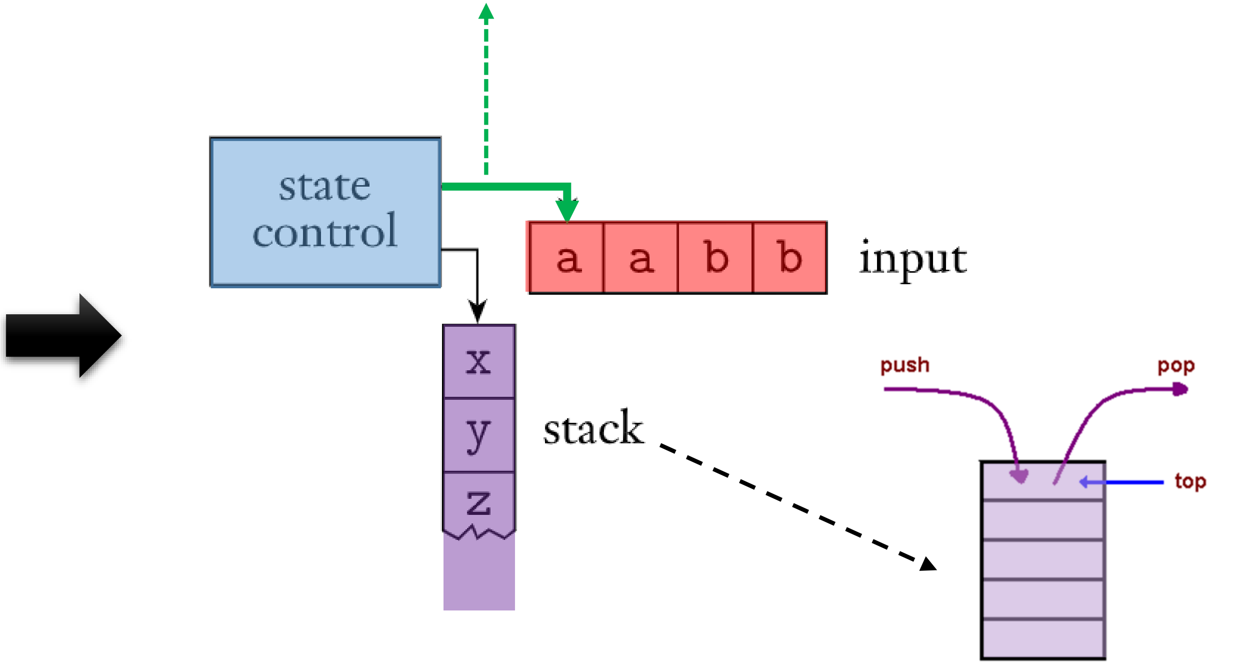
Tape:
contains the input string

Input head:
pointing at the next input symbol to be read



State Control:
the states and transition function

Finite automata

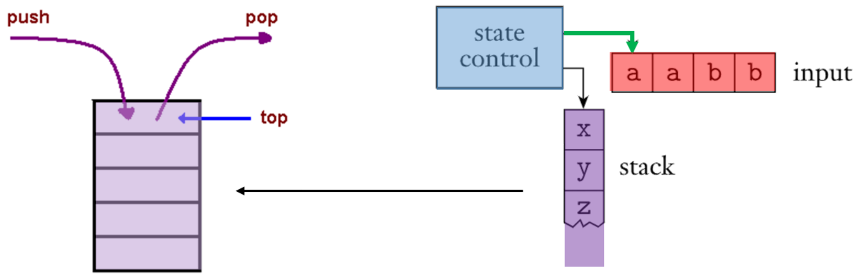


Pushdown automata

Chapter 2

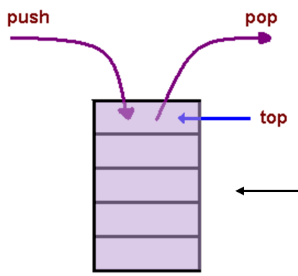
PUSHDOWN AUTOMATA (PDA)

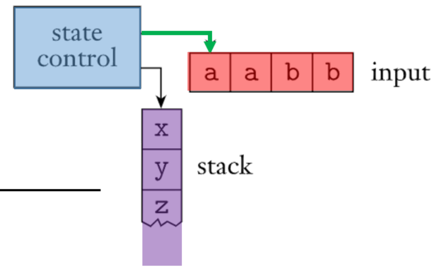
- Stack is of infinite size.
- Stack is a "last in, first out" storage device.
- **Example:** How a PDA recognizes the nonregular language $\{0^n1^n | n \geq 0\}$.



PUSHDOWN AUTOMATA (PDA)

- Stack is of infinite size.
- Stack is a "last in, first out" storage device.
- **Example:** How a PDA recognizes the nonregular language $\{0^n1^n | n \geq 0\}$.
- Can use its stack to store the number of 0's it has seen.

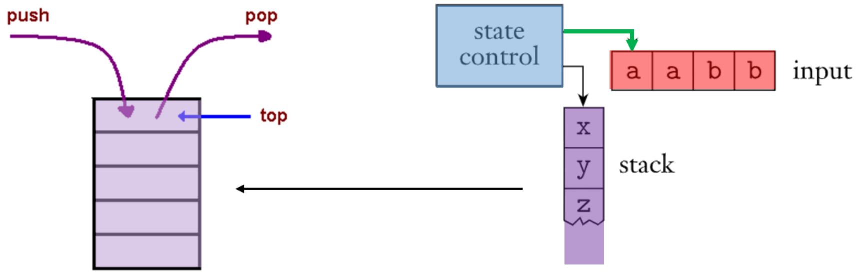




PUSHDOWN AUTOMATA (PDA)

- Stack is of infinite size.
- Stack is a "last in, first out" storage device.
- **Example:** How a PDA recognizes the nonregular language $\{0^n 1^n | n \geq 0\}$.
- Can use its stack to store the number of 0's it has seen.

> - As each 0 is read, push it onto the stack
> - As soon as 1's are read, pop a 0 off the stack
> - If reading the input is finished exactly when the stack is empty, accept the input else reject the input



DETERMINISTIC AND NONDETERMINISTIC (PDA)

- **Deterministic** and **nondeterministic** pushdown automata are not equivalent in power.
  - Nondeterministic pushdown automata recognize certain languages that no deterministic pushdown automata can recognize.

- Note: nondeterministic PDAs are equivalent in power to context-free grammars.

FORMAL DEFINITION OF A PUSHDOWN AUTOMATON

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

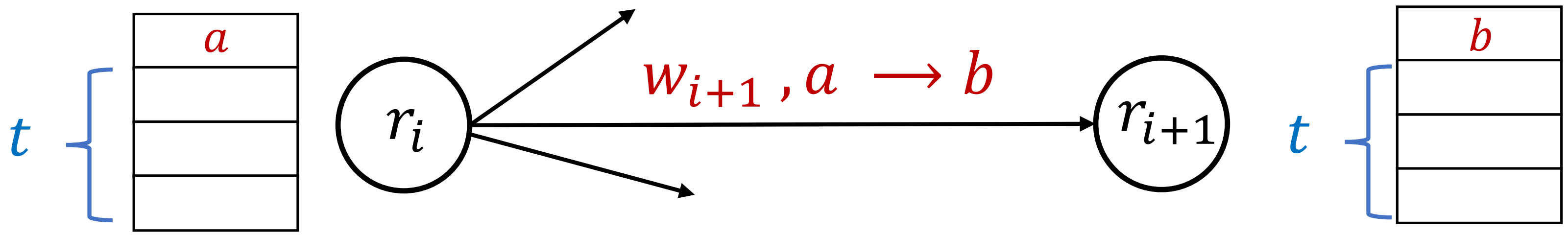
1. Q is the set of states,
2. Σ is the input alphabet,
3. Γ is the stack alphabet,
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the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

Where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

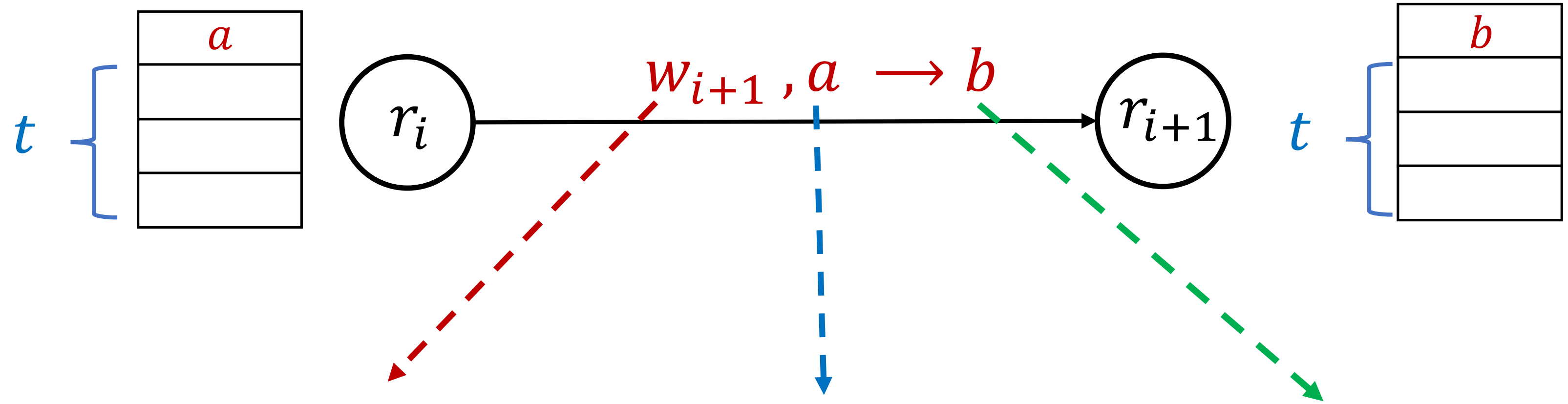
COMPUTAIONS OF A PUSHDOWN AUTOMATON

- A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.
- It accepts input $w = w_1 w_2 \cdots w_m$, where each $w_i \in \Sigma_\varepsilon$ and sequences of states $r_0, r_1, \cdots, r_m \in Q$ and strings $s_0, s_1, \cdots, s_m \in \Gamma^*$ exist that satisfy the following conditions.

1. $r_0 = q_0$ and $s_0 = \varepsilon$.
2. For $i = 0, \ldots, m-1$, we have $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$. This condition states that M moves properly according to the state, stack, and next input symbol.
3. $r_m \in F$. This condition states that an accept state occurs at the input end.



COMPUTAIONS OF A PUSHDOWN AUTOMATON



Pop: a
Push : b

Input symbol
Maybe ε

Symbol on top of the stack.
This symbol is popped.
Maybe ε : means the stack is Neither read nor popped

This symbol is pushed onto the stack.
This symbol is popped.
Maybe ε : means nothing is pushed.

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 1:** The formal description of the PDA that recognizes the language

$$\{0^n 1^n | n \geq 0\}.$$

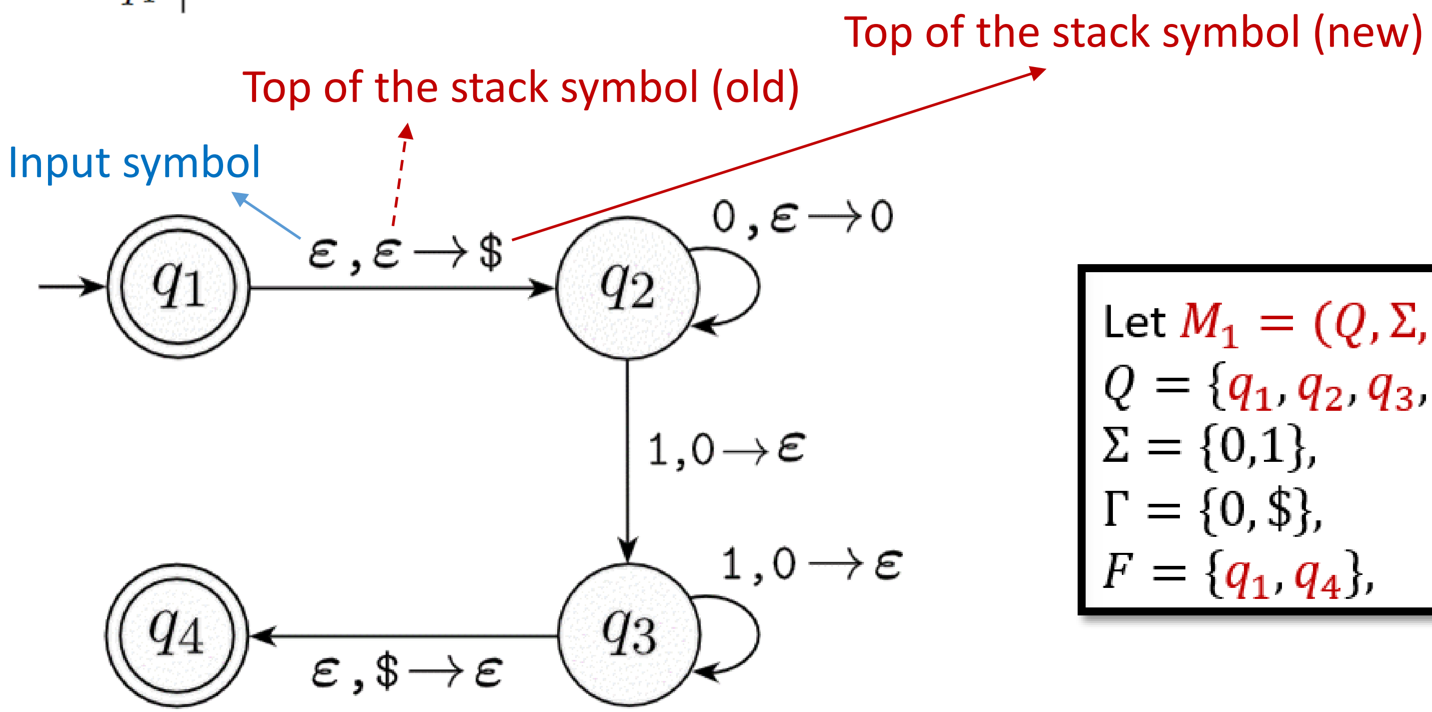
- Let $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, F)$, where
- $Q = \{q_1, q_2, q_3, q_4\}$,
- $\Sigma = \{0,1\}$,
- $\Gamma = \{0, \$\}$,
- $F = \{q_1, q_4\}$,
- And δ is given by the following table, wherein blank entries signify ∅.

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									

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 1:** (cont.)

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									

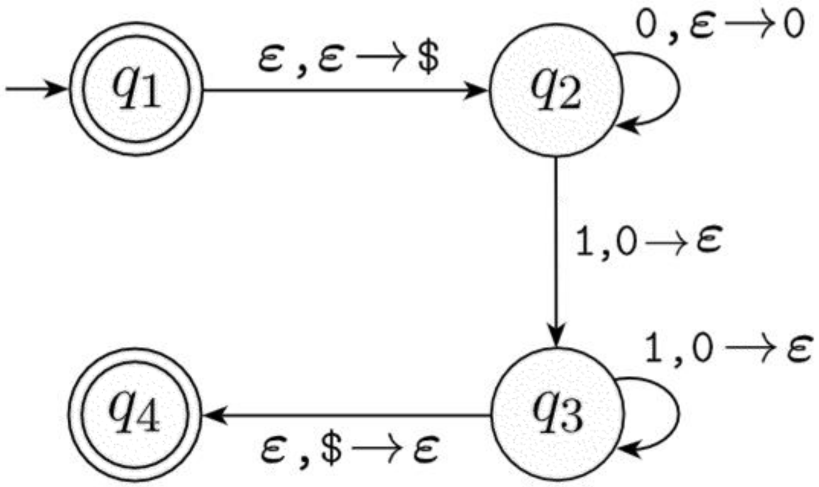


Let $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, F)$, where
$Q = \{q_1, q_2, q_3, q_4\}$,
$\Sigma = \{0,1\}$,
$\Gamma = \{0, \$\}$,
$F = \{q_1, q_4\}$,

Chapter 2

EXAMPLES OF PUSHDOWN AUTOMATA

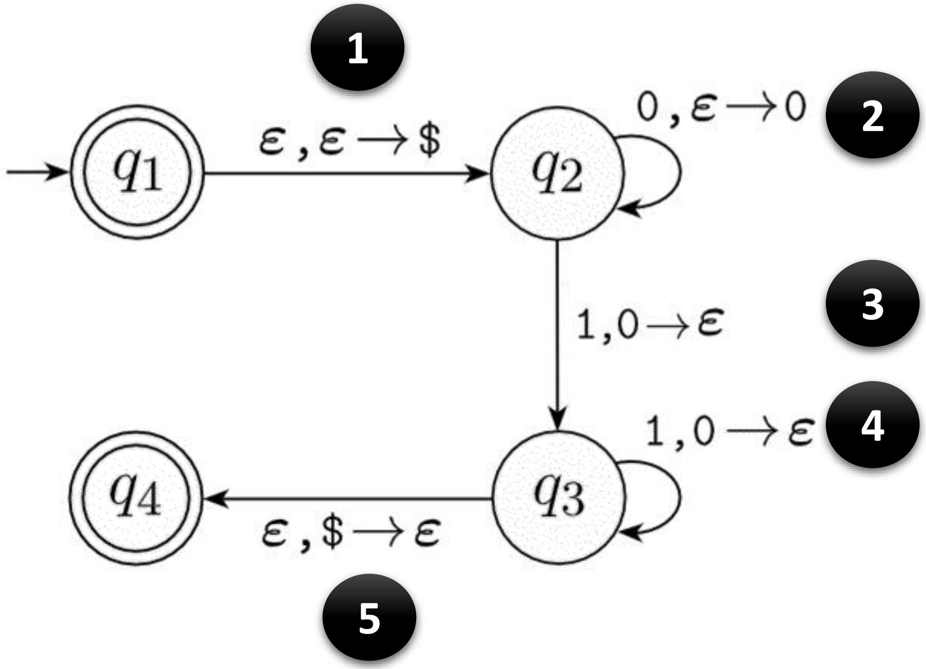
- **Example 1:** (cont.)
- "$a, b \rightarrow c$" : to signify that when the machine is reading an a from the input, it may replace the symbol b on the top of the stack with a c.
- $a = \varepsilon$: the machine may make this transition without reading any symbol from the input.
- $b = \varepsilon$: the machine may make this transition without reading and popping any symbol from the stack.
- $c = \varepsilon$: the machine does not write any symbol on the stack when going along this transition.



Chapter 2

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 1:** (cont.)



- no explicit mechanism to allow the PDA to test for an empty stack.
- This PDA is able to get the same effect by initially placing a special **symbol $** on the stack.

- $b = \varepsilon$: the machine may make this transition

witho

the st

- $c =$

on the

1. Creating an empty stack
2. If input is 0, then push a 0 onto the stack

3,4. If input is 1, then pop a 0 from the stack and no push

5. The last input symbol and stack is empty → accept the string

EXAMPLES OF PUSHDOWN AUTOMATA

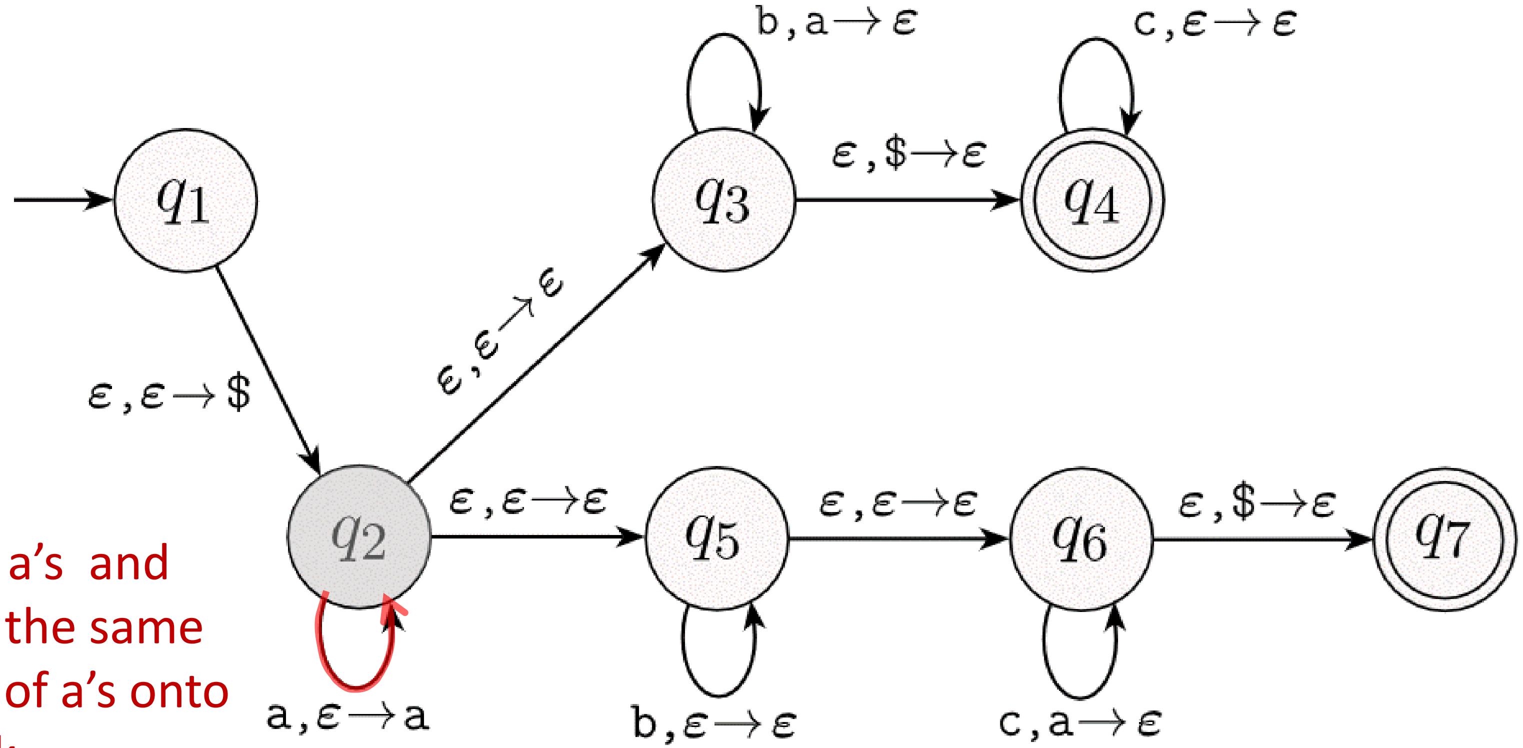
- **Example 3:** Find the PDA that recognizes the language

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

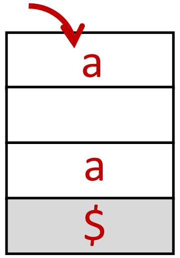
EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 2:** Find the PDA that recognizes the language

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



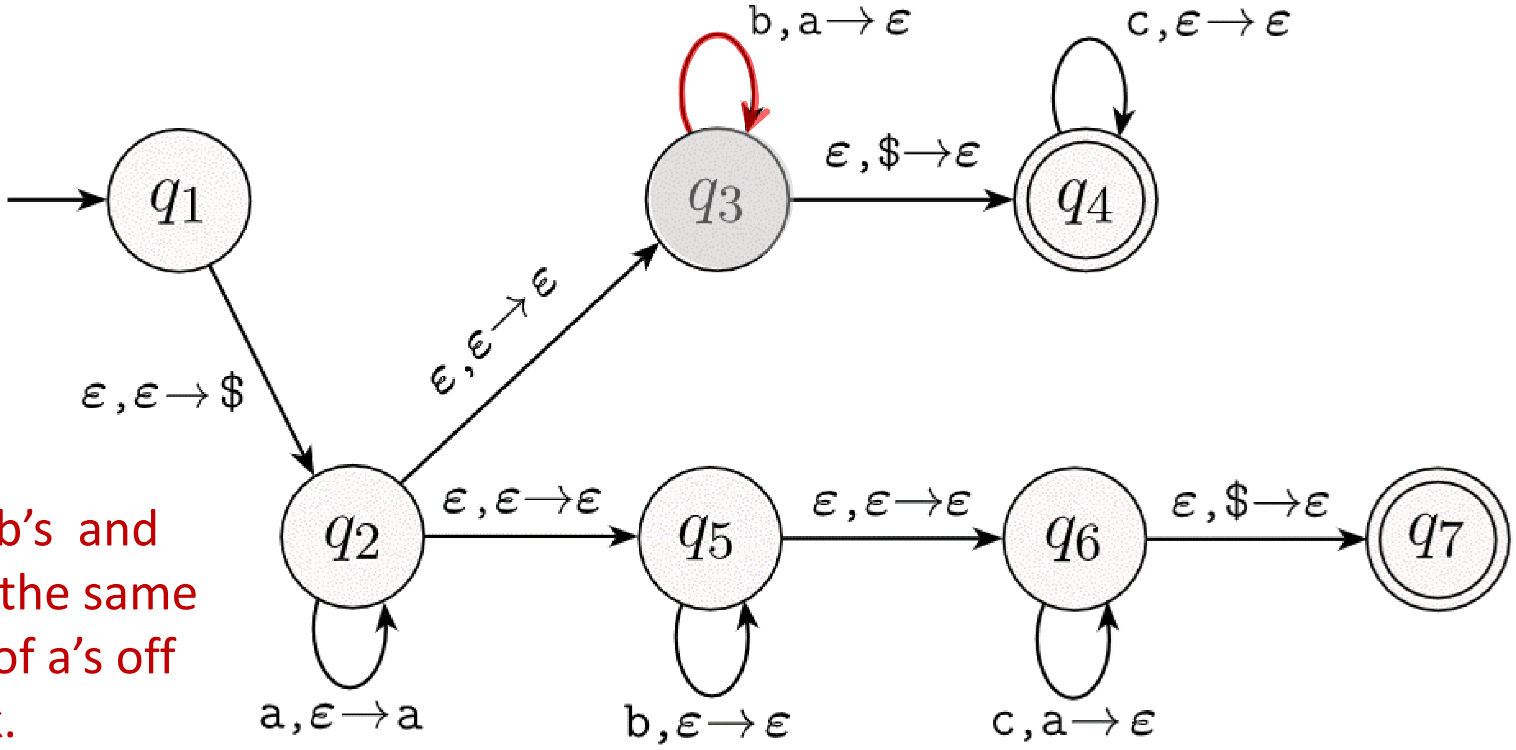
- Reading a's and pushing the same amount of a's onto the stack.



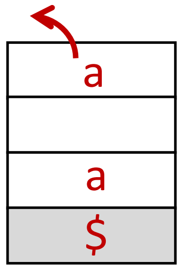
EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 2:** Find the PDA that recognizes the language

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



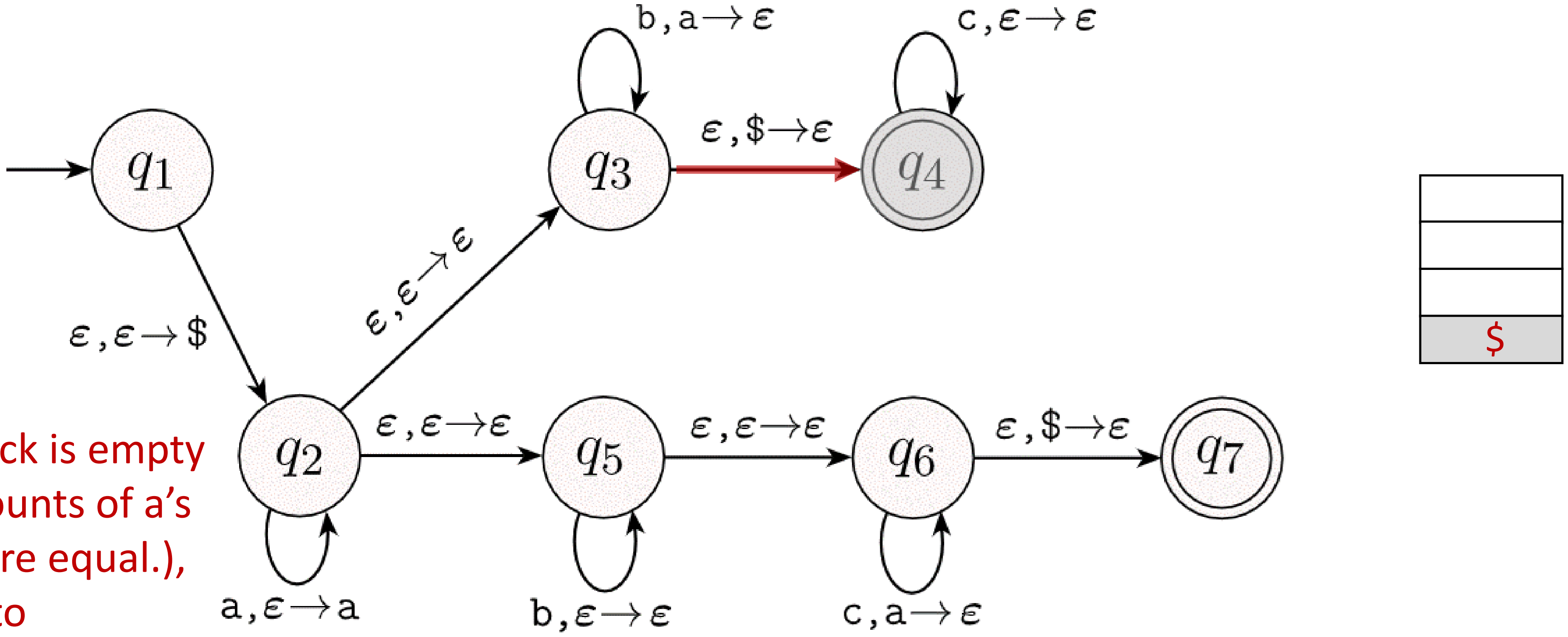
- Reading b's and popping the same amount of a's off the stack.



EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 2:** Find the PDA that recognizes the language

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

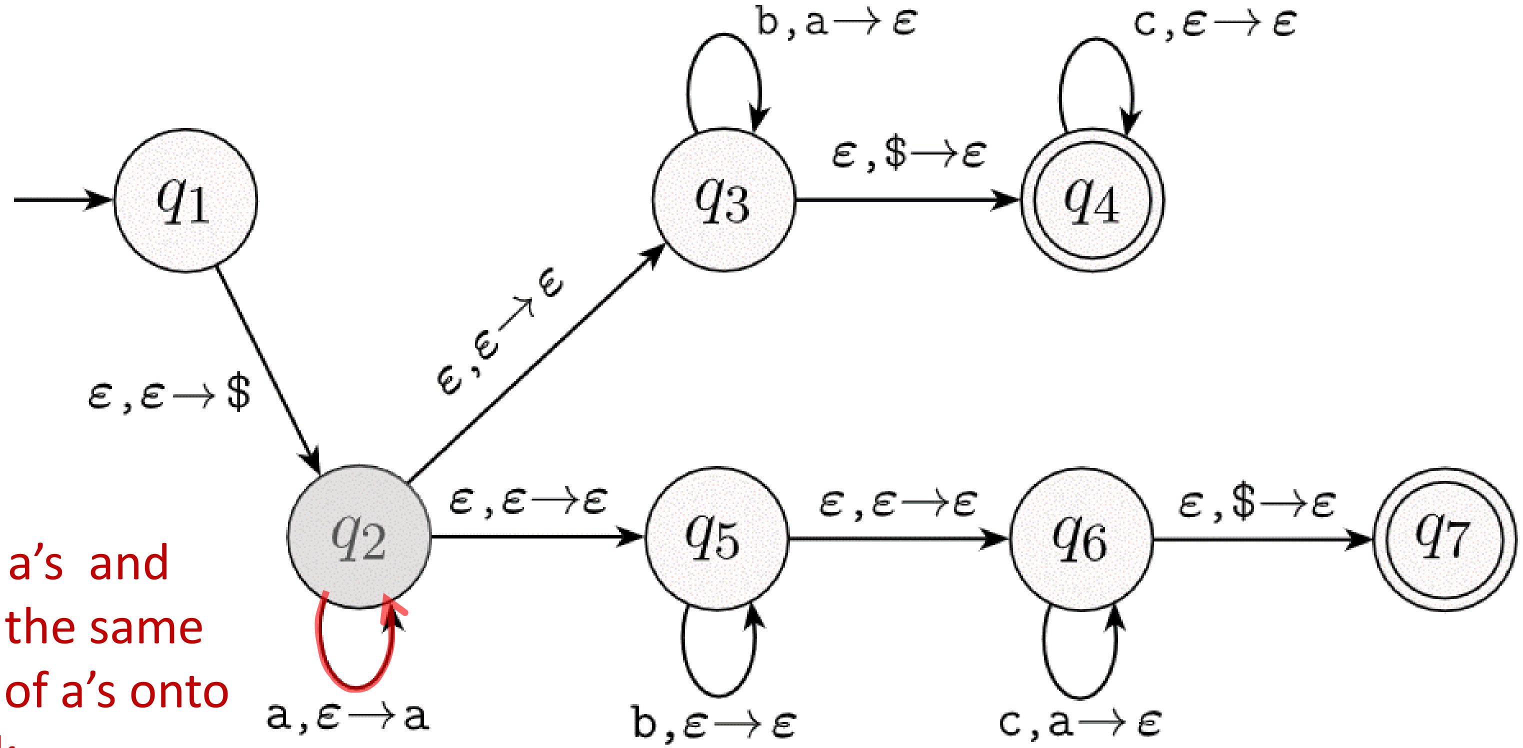


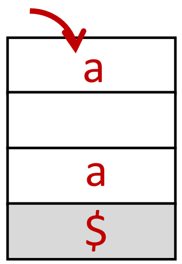
- If the stack is empty (the amounts of a's and b's are equal.), then go to the accept state.

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 2:** Find the PDA that recognizes the language

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



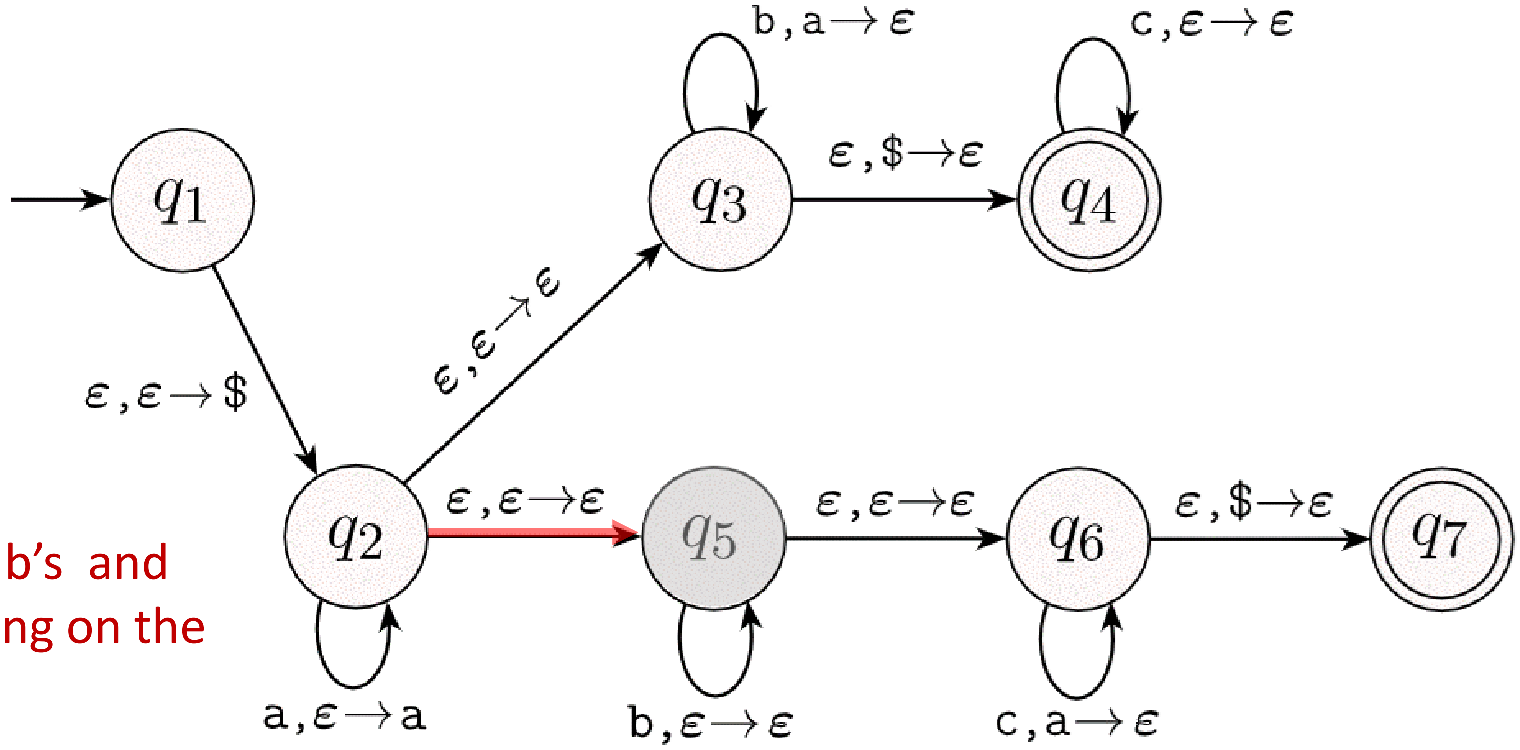


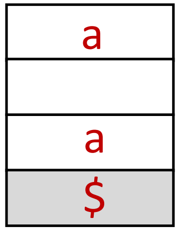
- Reading a's and pushing the same amount of a's onto the stack.

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 2:** Find the PDA that recognizes the language

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



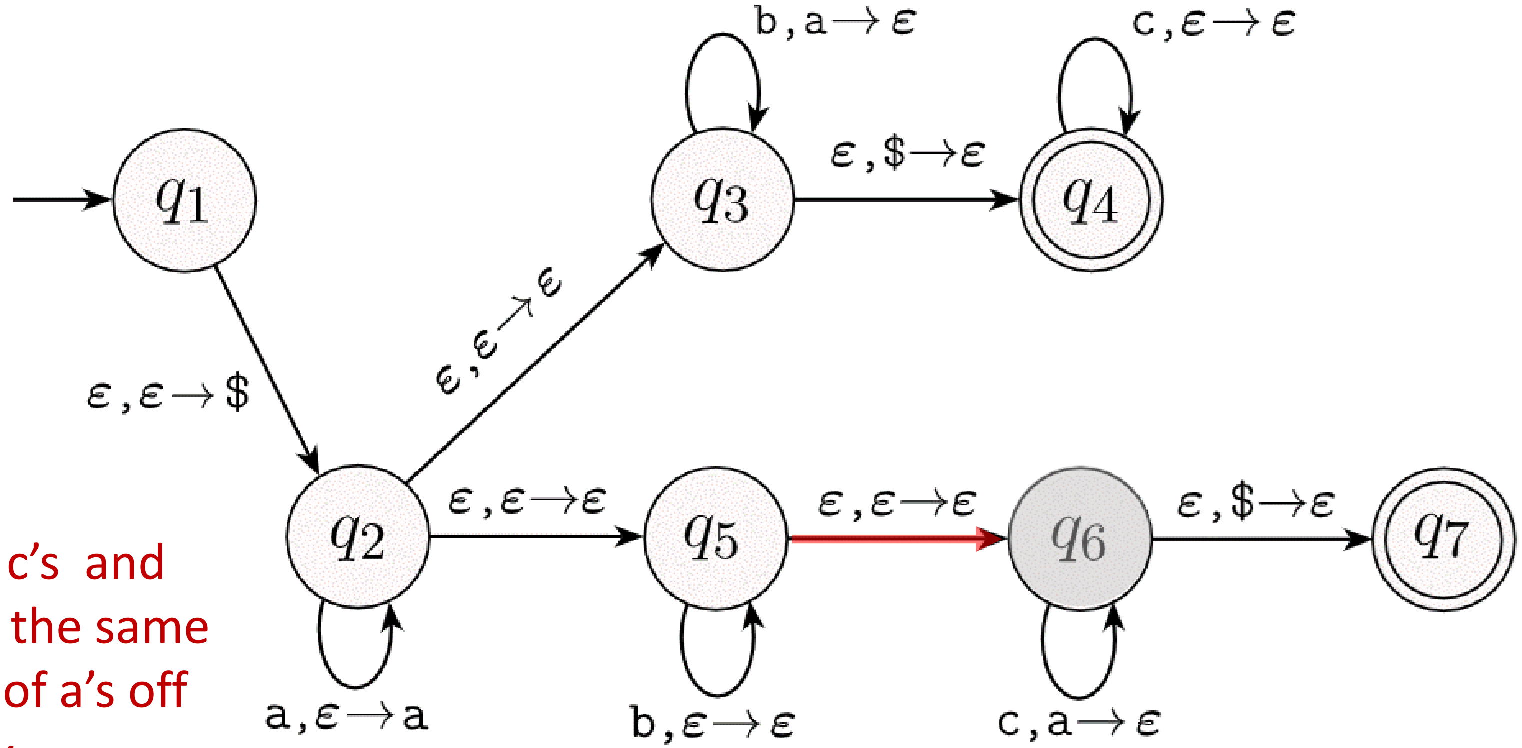


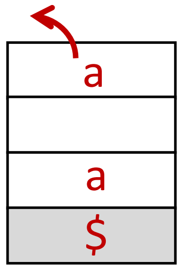
- Reading b's and do nothing on the stack.

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 2:** Find the PDA that recognizes the language

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



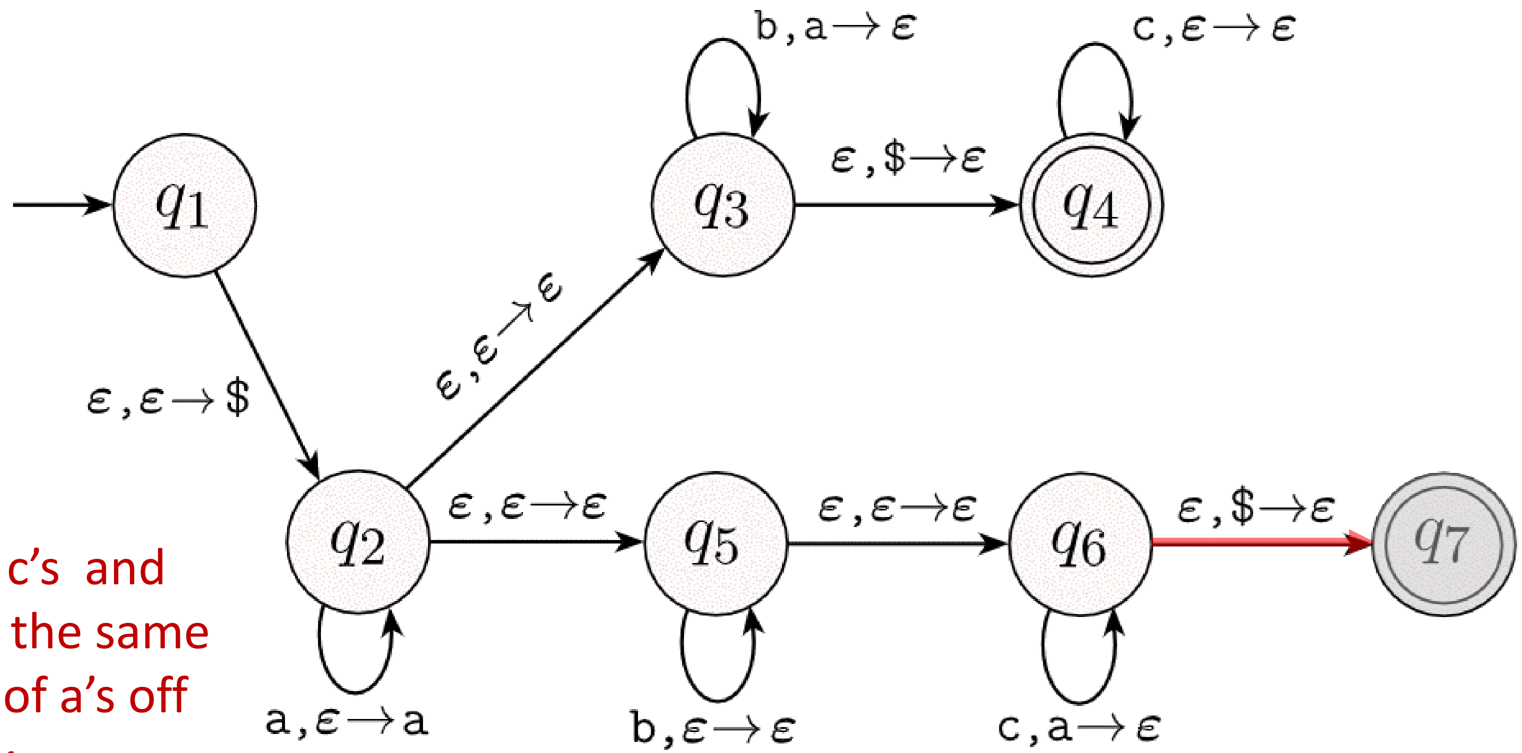


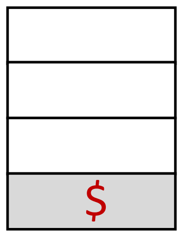
- Reading c's and popping the same amount of a's off the stack.

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 2:** Find the PDA that recognizes the language

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



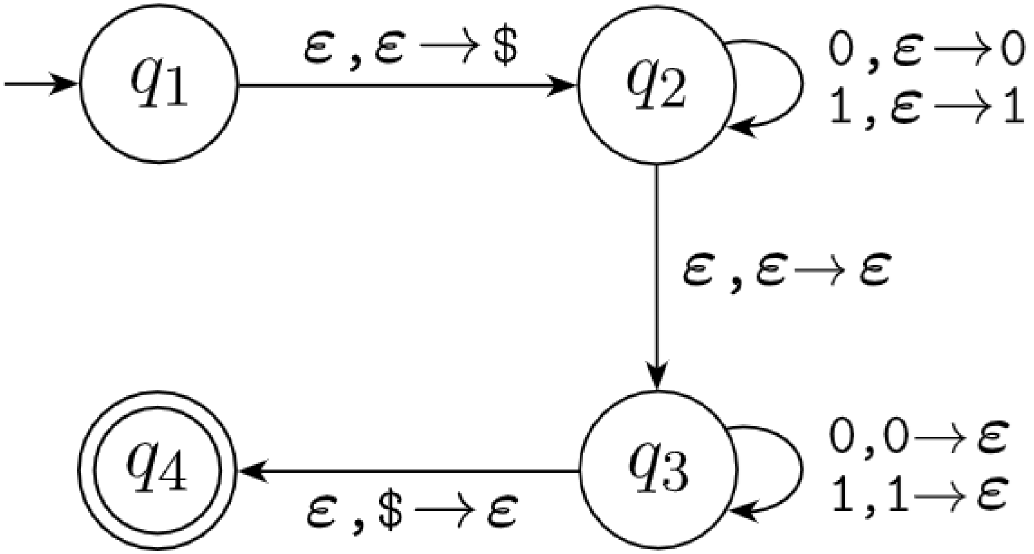


- Reading c's and popping the same amount of a's off the stack.

EXAMPLES OF PUSHDOWN AUTOMATA

- **Example 3:** Find the PDA that recognizes the language

$$\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$$



EQUIVALENCE OF PDA WITH CFG

- THEOREM: A language is context free if and only if some pushdown automaton recognizes it.

- Part1: If a language is context free, then some pushdown automaton recognizes it.

- Part2: If a pushdown automaton recognizes some language, then it is context free.

EQUIVALENCE OF PDA WITH CFG

- Part1: If a language is context free, then some pushdown automaton recognizes it.

- PROOF IDEA:
- Let A be a CFL, Then A has a CFG **G**, generating it.
- Using the following example we show how to convert **G** into an equivalent PDA, **P**.

- **Example:** Grammar **G**

$$S \rightarrow BS|A \qquad : R1, R2$$
$$A \rightarrow 2A|\varepsilon \qquad : R3, R4$$
$$B \rightarrow BB1|0 \qquad : R5, R6$$

Using **left-most derivation** for generating **001** will be :

$$\underset{R1}{S \Rightarrow} \underset{R5}{BS \Rightarrow} \underset{R6}{BB1S \Rightarrow} \underset{R6}{0B1S \Rightarrow} \underset{R2}{001S \Rightarrow} \underset{R4}{001A \Rightarrow} 001$$

EQUIVALENCE OF PDA WITH CFG

- PROOF IDEA: (cont.)

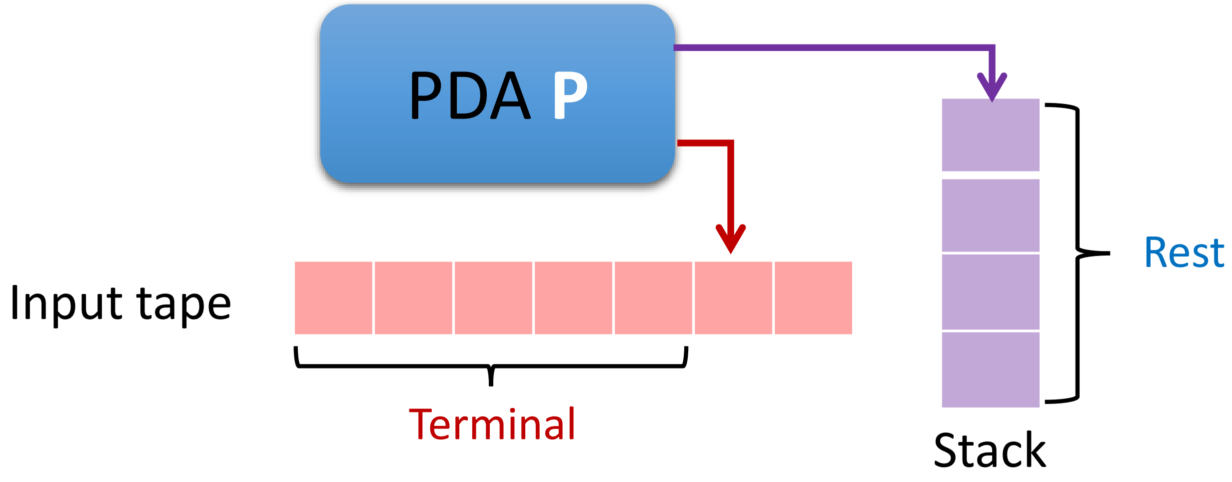
- In general form all of intermediate string in the derivation for generating a string will be in form

$$000110010101SA01SSA00SSAA0$$

Terminal Rest

(contains terminals and nonterminals)

- The **PDA** P will work by accepting its input w, if G generates that input,
  - by determining whether there is a derivation for w. (left-most derv.)

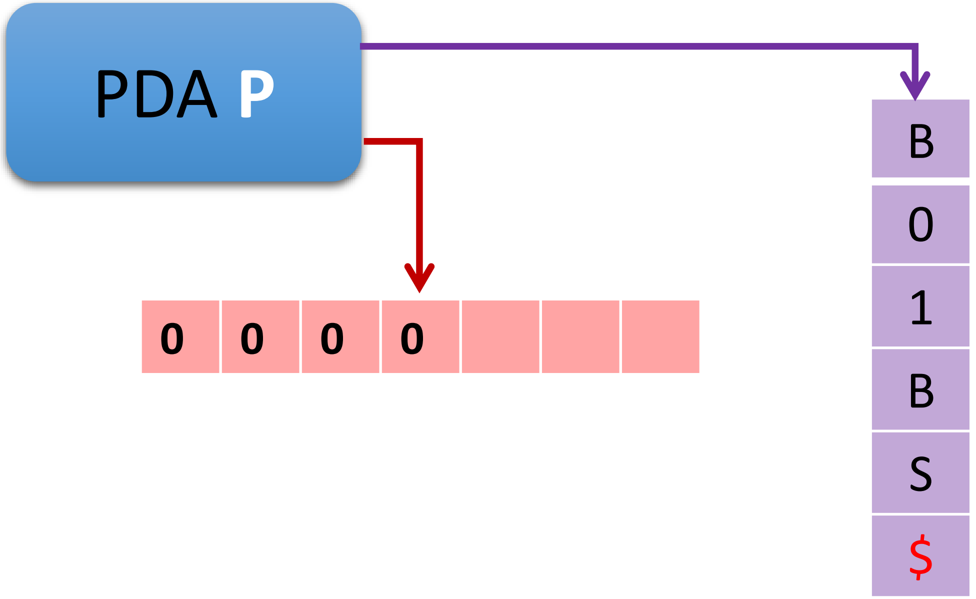


EQUIVALENCE OF PDA WITH CFG

- PROOF IDEA: (cont.)

- Let's see how we can store a an **intermediate string** (sentential form) in the derivation in a pushdown automaton.

$$S \overset{*}{\Rightarrow} 0000B01BS \Rightarrow \cdots$$



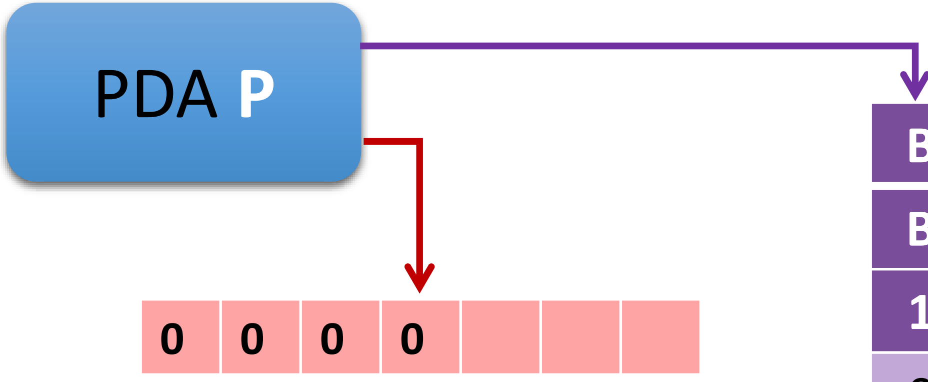
EQUIVALENCE OF PDA WITH CFG

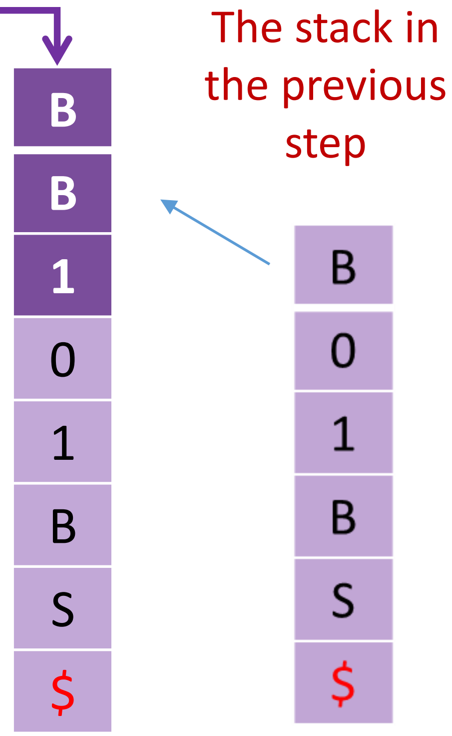
- PROOF IDEA: (cont.)

- At each step of the derivation, we expand the leftmost nonterminal using the rules.

  $S \overset{*}{\Rightarrow} 0000B01BS \Rightarrow \cdots$

  Rule: $\mathrm{B} \longrightarrow \mathrm{BB1}$





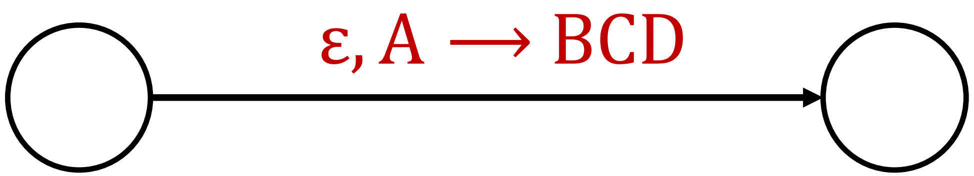
- so the steps that we follow in this computation will be
  - We match the stack top to a rule
  - Pop the top of the stacks and
  - push the left side of the rule (backward)

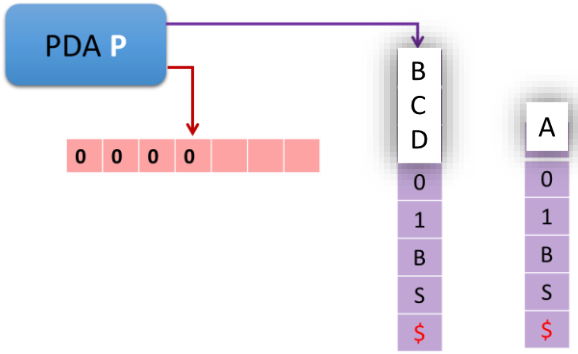
- Since PDA is nondeterministic, all rules at the same time will be examined.

EQUIVALENCE OF PDA WITH CFG

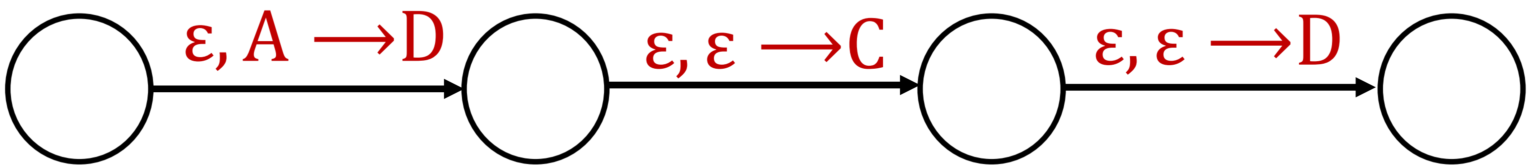
- PROOF IDEA: (cont.)

- How can we add the following rule in a PDA?
- **Rule:** A ⟶ BCD where A,B,C, and D are nonterminals





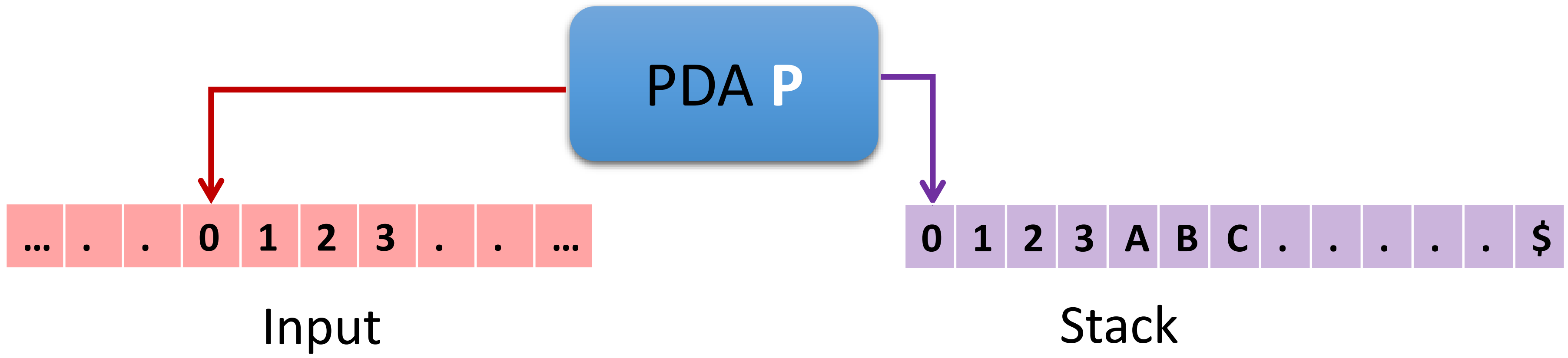
- We are not allowed to push multiple symbols.

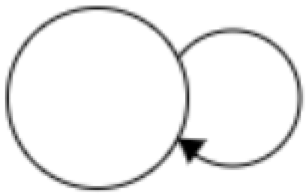


EQUIVALENCE OF PDA WITH CFG

- PROOF IDEA: (cont.)

- What will be the **transition** if there is a **terminal** on top of the stack?





$0,0 \longrightarrow \varepsilon$

$1,1 \longrightarrow \varepsilon$

$2,2 \longrightarrow \varepsilon$

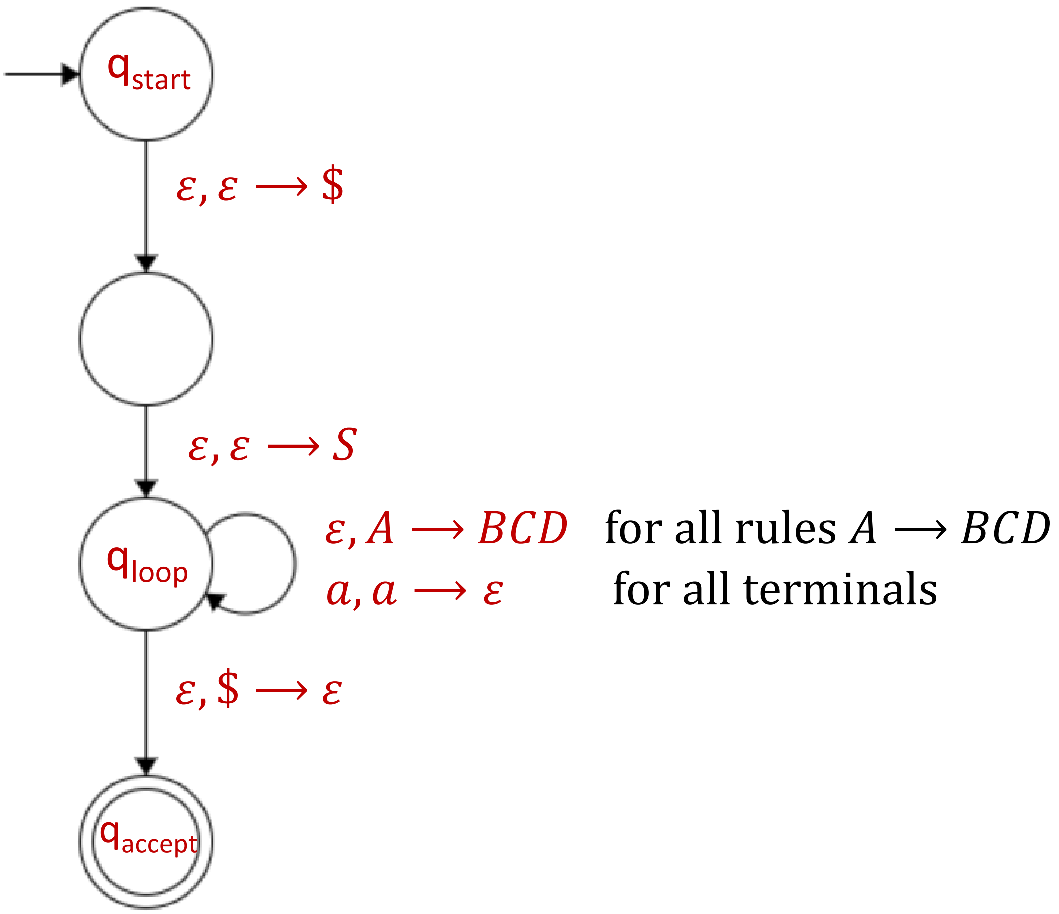
$3,3 \longrightarrow \varepsilon$

For all symbols in alphabet.
(nonterminals)

EQUIVALENCE OF PDA WITH CFG

- PROOF IDEA: (cont.)

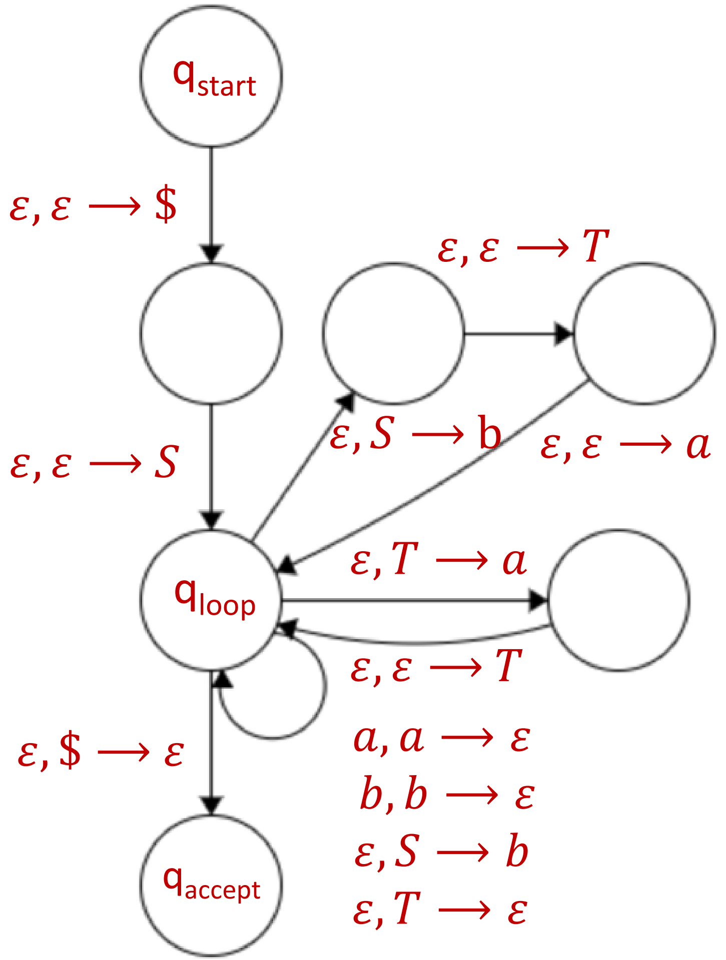
- The Final PDA



EQUIVALENCE OF PDA WITH CFG

- Example: construct aPDA P1 from the following CFG G.

$$S \rightarrow aTb|b$$
$$T \rightarrow Ta|\varepsilon$$



Chapter 2

EQUIVALENCE OF PDA WITH CFG

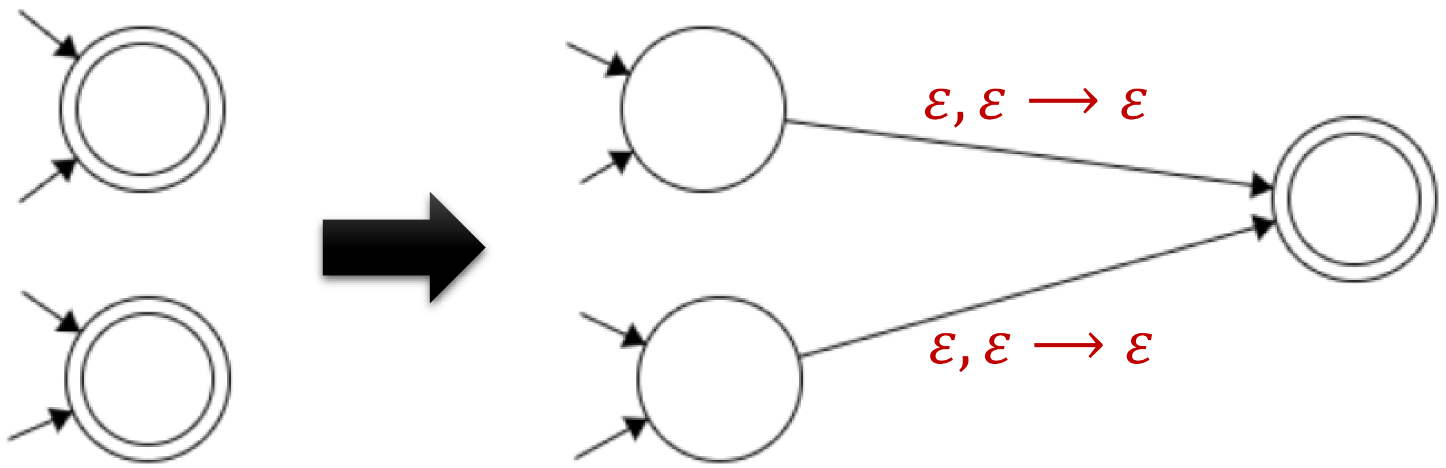
- Part2: If a pushdown automaton recognizes some language, then it is context free.

- Proof idea:
- We have a PDA P, and we want to make a CFG G that generates all the strings that P accepts.
- G should **generate a string** if that string causes the PDA to go from its start state to an accept state.

- **Steps:**
  - **Step 1:** simplify the PDA
  - **Step2:** Building CFG

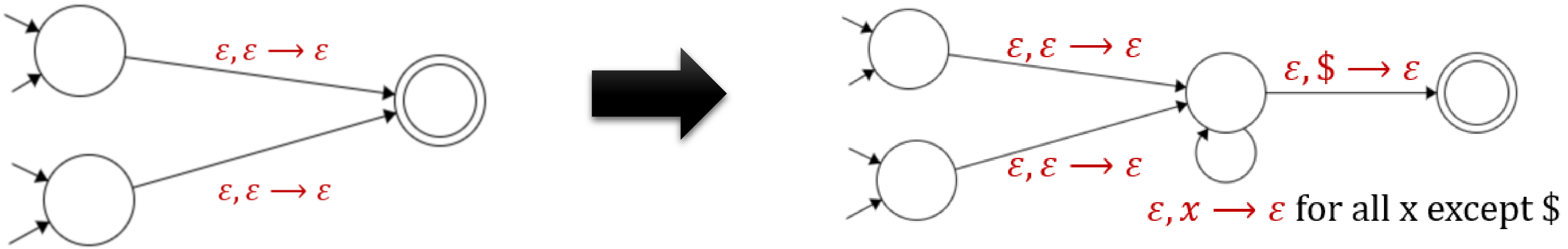
EQUIVALENCE OF PDA WITH CFG

- Proof idea: (cont.)
- **Step1:** Simplification by modifying P as below to give it the following three features:
  1. It has a single accept state, q_{accept}.



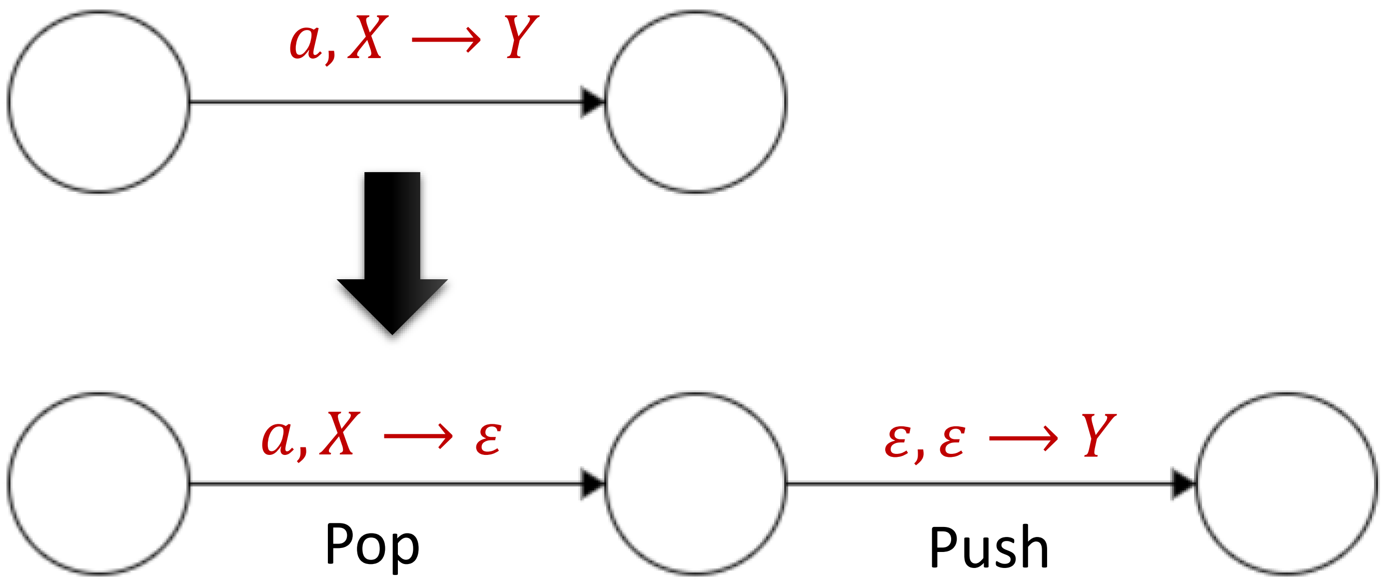
EQUIVALENCE OF PDA WITH CFG

- Proof idea: (cont.)
- **Step1:** Simplification by modifying P as below to give it the following three features:
  2. It empties its stack before accepting.



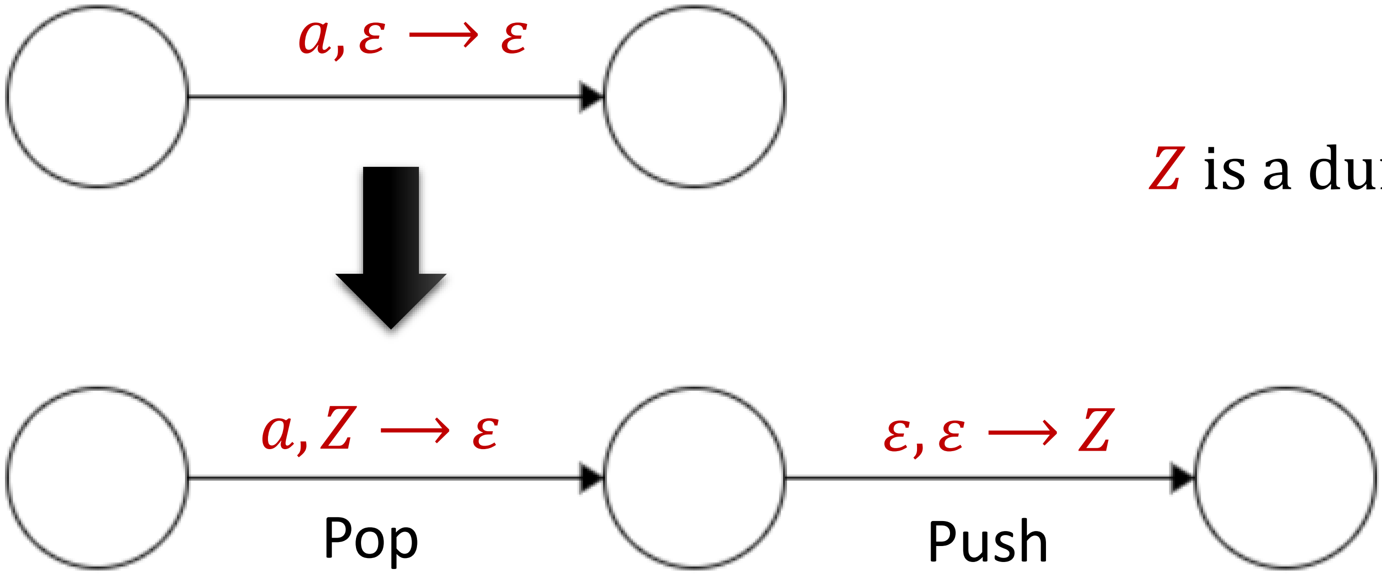
EQUIVALENCE OF PDA WITH CFG

- Proof idea: (cont.)
- **Step1:** Simplification by modifying P as below to give it the following three features:
  3. Each transition either pushes a symbol onto the stack (a push move) or pops one off the stack (a pop move), but it does not do both at the same time.



EQUIVALENCE OF PDA WITH CFG

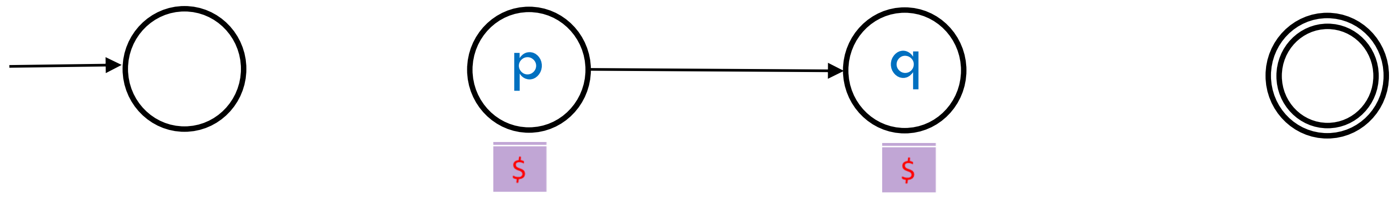
- Proof idea: (cont.)
- **Step1:** Simplification by modifying P as below to give it the following three features:
  3. Each transition either pushes a symbol onto the stack (a push move) or pops one off the stack (a pop move), but it does not do both at the same time.



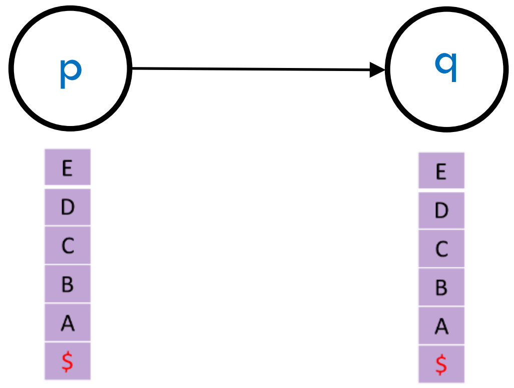
Z is a dummy symbol.

EQUIVALENCE OF PDA WITH CFG

- Proof idea: (Cont.)
- **Step2: Building CFG**
- For **each pair of states p and q** in P, the grammar will have a **variable** A_{pq}.
  - This variable **generates all the strings** that can take P from p with an empty stack to q with an empty stack.



  - such strings can also take P from p to q, regardless of the stack contents at p, leaving the stack at q in the same condition as it was at p.



EQUIVALENCE OF PDA WITH CFG

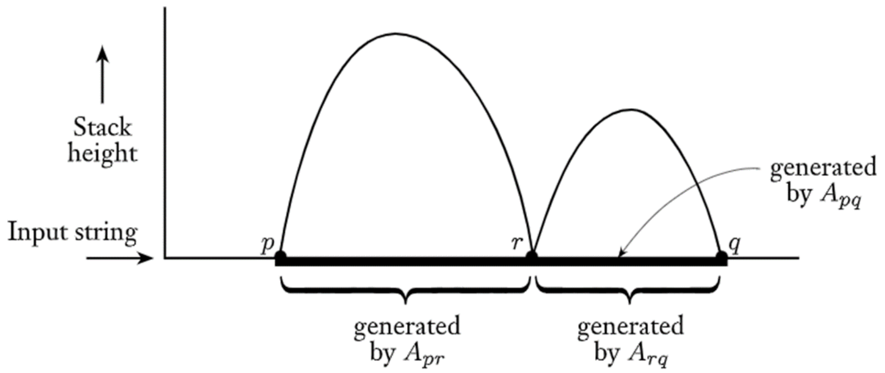
- Proof idea: (Cont.)
- **Step2:**

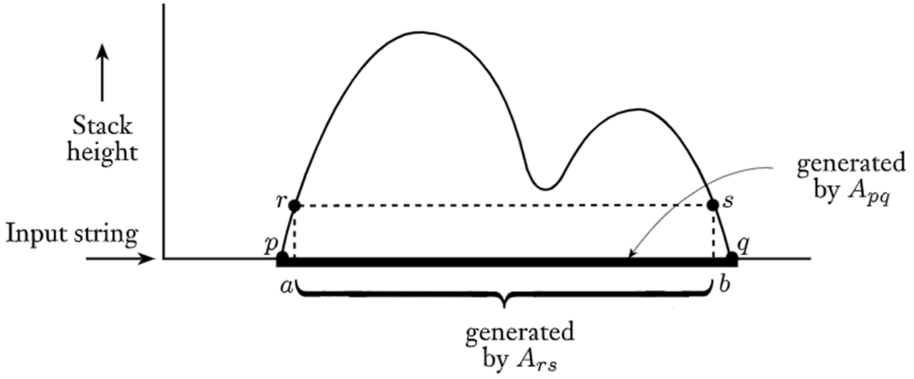
- To design G, we must understand how P operates on the strings generated by A_{pq} .
  - For any such string x, P's **first move** on x must be a push,
    - Since stack is empty.
  - Similarly, the **last move** on x must be a pop.
    - Since the stack should be empty.

EQUIVALENCE OF PDA WITH CFG

- Proof idea: (Cont.)
- **Step2:**

- Two possibilities occur during P's computation on x.
- Either the symbol popped at the end is the symbol that was pushed at the beginning, or not.





$$A_{pq} \longrightarrow A_{\mathrm{pr}}\, A_{\mathrm{rq}}$$

$$A_{pq} \longrightarrow \mathrm{a}\, A_{\mathrm{rs}}\, \mathrm{b}$$