CS 302.1 - Automata Theory

Shantanav Chakraborty

Center for Quantum Science and Technology (CQST)

Center for Security, Theory and Algorithms (CSTAR)

IIIT Hyderabad

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Quick Recap

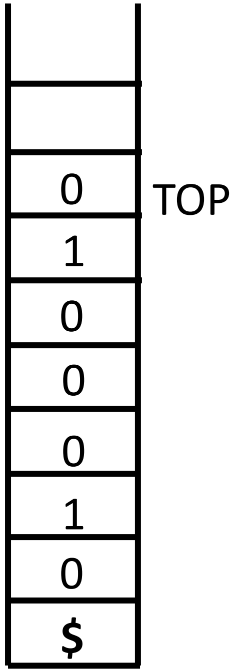
Chomsky Normal Form: If every *rule* of the CFG is of the form

$$\boldsymbol{A \to BC} \qquad [B, C \text{ are not start variables}]$$
$$\boldsymbol{A \to a} \qquad [a \text{ is a terminal}]$$
$$\boldsymbol{S \to \epsilon} \qquad [S \text{ is the Start Variable}]$$

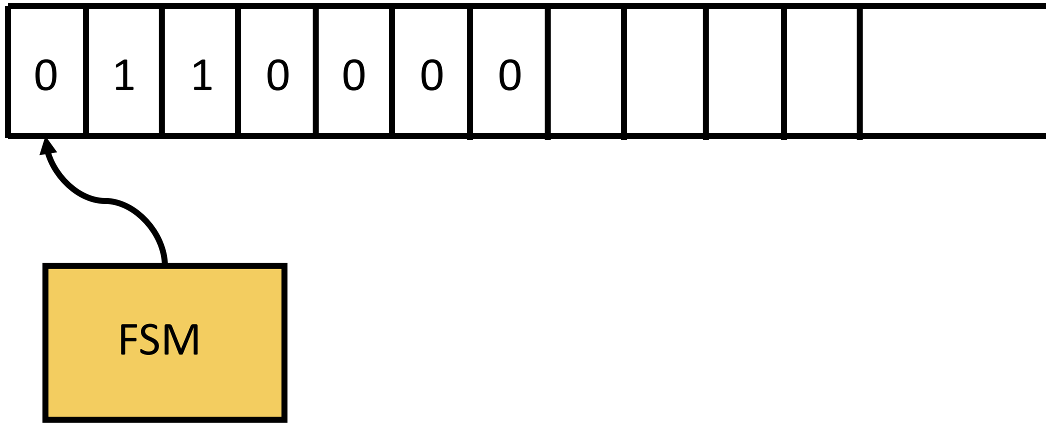
- Any CFG can be converted to a grammar in CNF that generates the same language.
- The number of steps required to derive a string $w = 2|w| - 1$.
- Is crucial in deciding whether w is generated by a CFG G.

Pushdown Automata

- Automata that recognizes CFLs
- FSM + stack
- FSM transitions by reading an input symbol and by interacting with the stack



One-way infinite tape holding the input string



Pushdown Automata

PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Pushdown Automata

PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Informally, the PDA for some language may work as follows:

- Read symbols from the input.

Pushdown Automata

PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Informally, the PDA for some language may work as follows:

- Read symbols from the input.
- As each 0 is read, push 0 on to the stack and remain in the state Q_0.

Pushdown Automata

PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Informally, the PDA for some language may work as follows:

- Read symbols from the input.
- As each 0 is read, push 0 on to the stack and remain in the state Q_0.
- If FSM is at Q_0, and a 1 is read, pop a 0 off the Stack and transition to Q_1.

Pushdown Automata

PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Informally, the PDA for some language may work as follows:

- Read symbols from the input.
- As each 0 is read, push 0 on to the stack and remain in the state Q_0.
- If FSM is at Q_0, and a 1 is read, pop a 0 off the Stack and transition to Q_1.
- If FSM is at Q_1, and a 1 is read, pop a 0 off the Stack and remain at Q_1.

Pushdown Automata

PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Informally, the PDA for some language may work as follows:

- Read symbols from the input.
- As each 0 is read, push 0 on to the stack and remain in the state Q_0.
- If FSM is at Q_0, and a 1 is read, pop a 0 off the Stack and transition to Q_1.
- If FSM is at Q_1, and a 1 is read, pop a 0 off the Stack and remain at Q_1.
- If FSM is at Q_1, and a 1 is read, pop a 0 off the Stack, push 1 on to the stack and transition to Q_2

Pushdown Automata

PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Informally, the PDA for some language may work as follows:

- Read symbols from the input.
- As each 0 is read, push 0 on to the stack and remain in the state Q_0.
- If FSM is at Q_0, and a 1 is read, pop a 0 off the Stack and transition to Q_1.
- If FSM is at Q_1, and a 1 is read, pop a 0 off the Stack and remain at Q_1.
- If FSM is at Q_1, and a 1 is read, pop a 0 off the Stack, push 1 on to the stack and transition to Q_2
- If the input is finished exactly when the stack is empty (TOP = $), ACCEPT the input.

Pushdown Automata

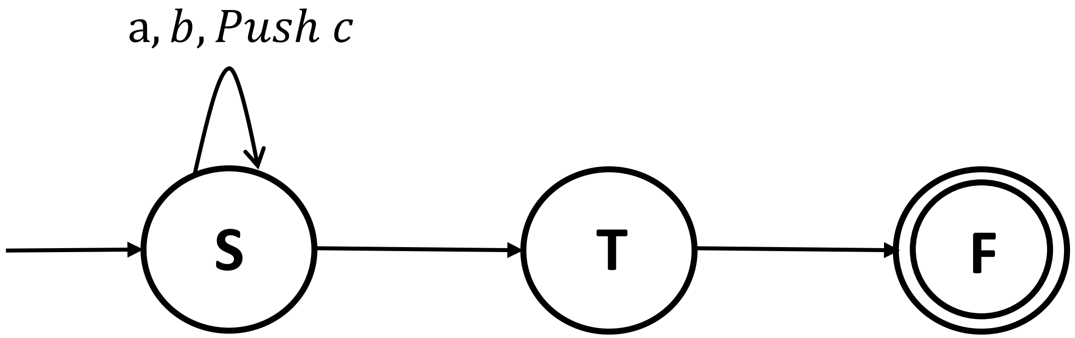
PDAs are **non-deterministic**. (Multiple transitions/input symbol possible)

Informally, the PDA for some language may work as follows:

- Read symbols from the input.
- As each 0 is read, push 0 on to the stack and remain in the state Q_0.
- If FSM is at Q_0, and a 1 is read, pop a 0 off the Stack and transition to Q_1.
- If FSM is at Q_1, and a 1 is read, pop a 0 off the Stack and remain at Q_1.
- If FSM is at Q_1, and a 1 is read, pop a 0 off the Stack, push 1 on to the stack and transition to Q_2
- If the input is finished exactly when the stack is empty (TOP = $), ACCEPT the input.
- REJECT otherwise (Stack becomes empty before all the inputs are read/non-empty after the entire input is read)

Pushdown Automata

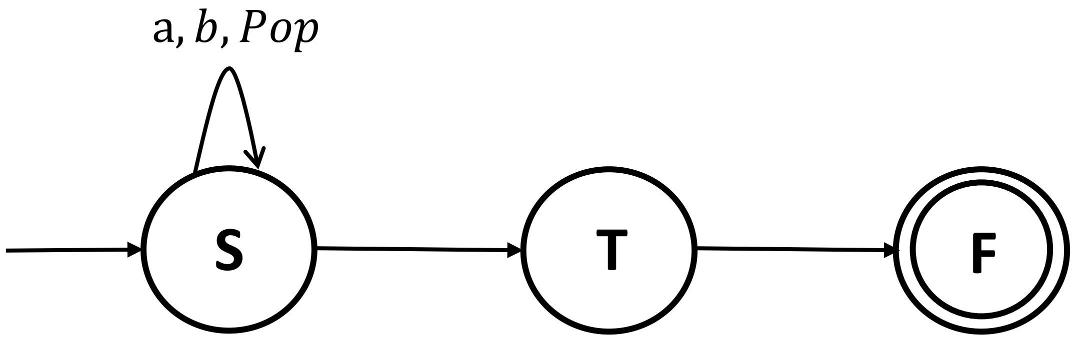
- How to represent a transition in a PDA?



If input symbol = a, Stack top = b, then Pop b and Push c onto the Stack

Pushdown Automata

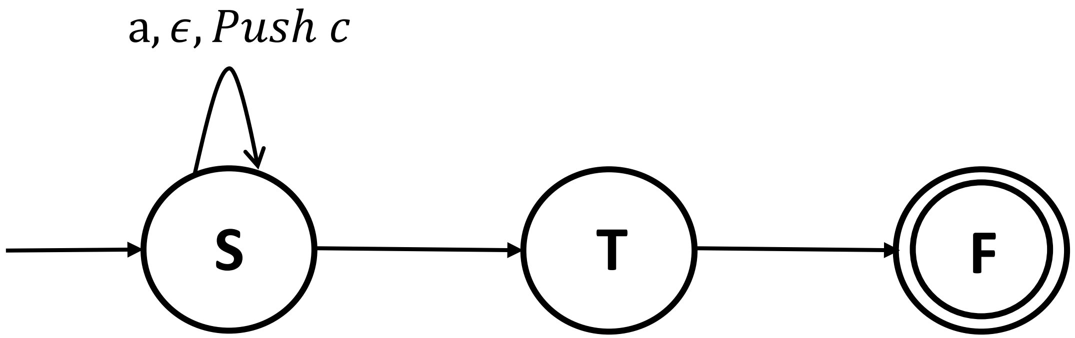
- How to represent a transition in a PDA?



If input symbol = a and Stack top = b, then Pop b

Pushdown Automata

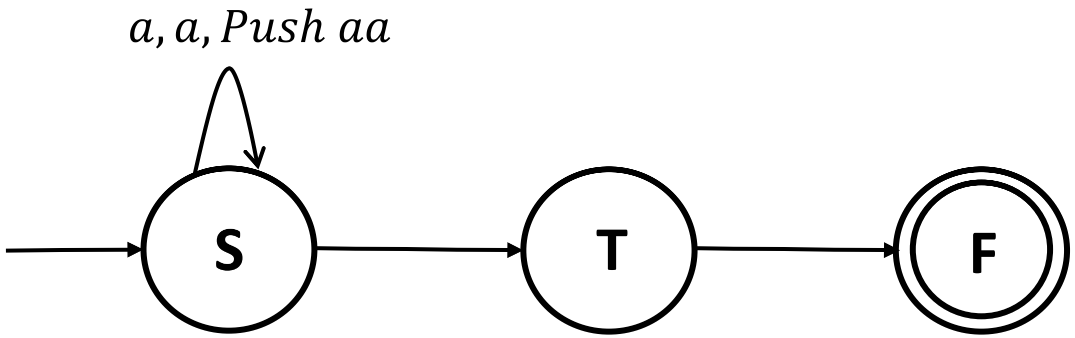
- How to represent a transition in a PDA?



If input symbol = a, then Push c

Pushdown Automata

- How to represent a transition in a PDA?

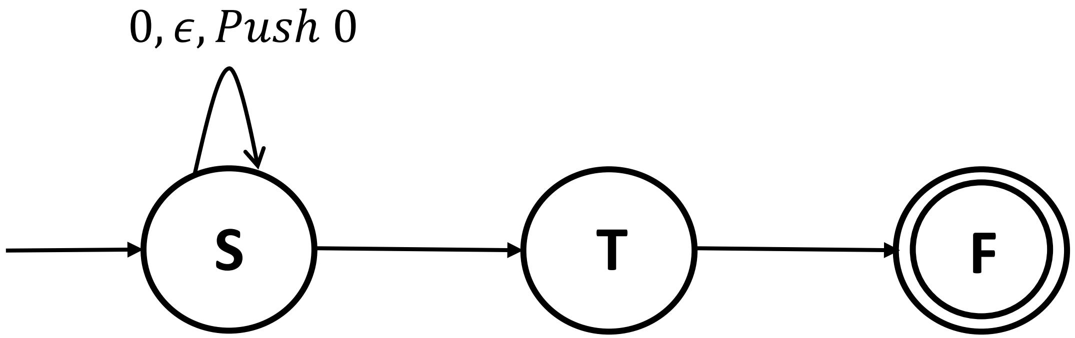


If input symbol = a, then Pop a and Push aa.

So effectively, the PDA pushes a onto the stack if it reads a on the input tape and the stack top = a.

Pushdown Automata

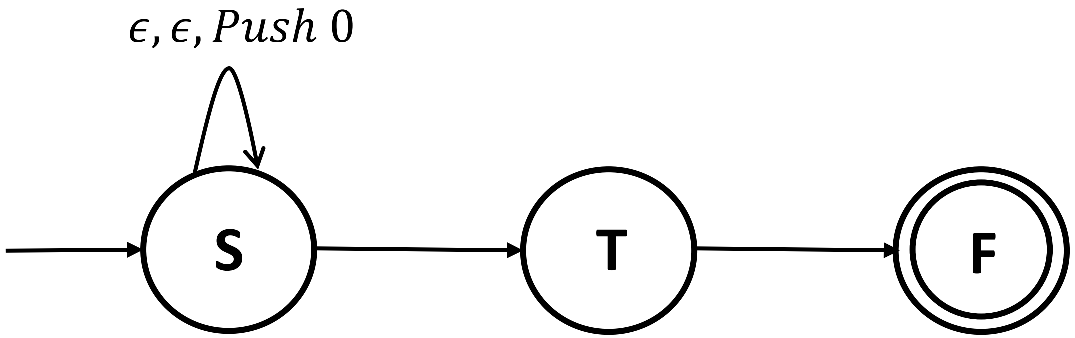
- How to represent a transition in a PDA?



If input symbol = 0, Push 0 onto the Stack irrespective of the element at the top of the stack

Pushdown Automata

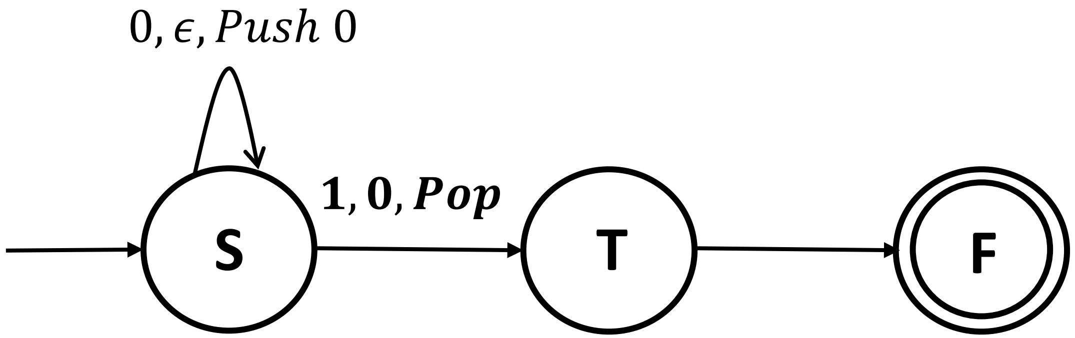
- How to represent a transition in a PDA?



Without reading the input symbol and the Stack top, Push 0 onto the Stack

Pushdown Automata

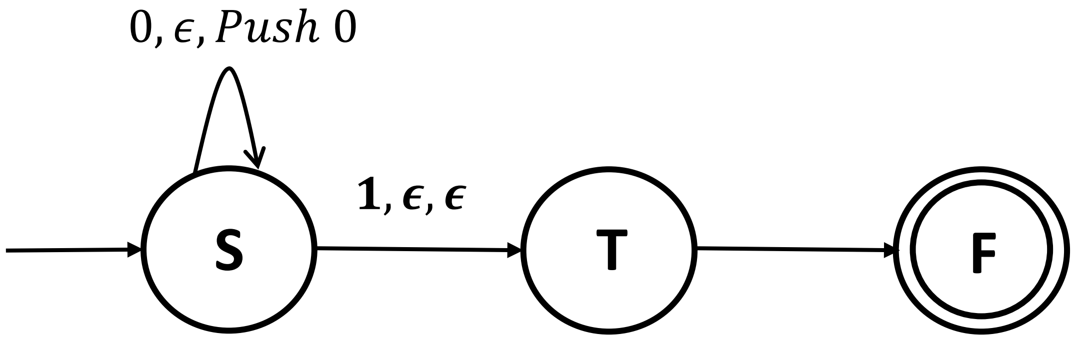
- How to represent a transition in a PDA?



If the input symbol is 1, and the element at the top of the stack is 0, pop it **(Pop 0)**.

Pushdown Automata

- How to represent a transition in a PDA?

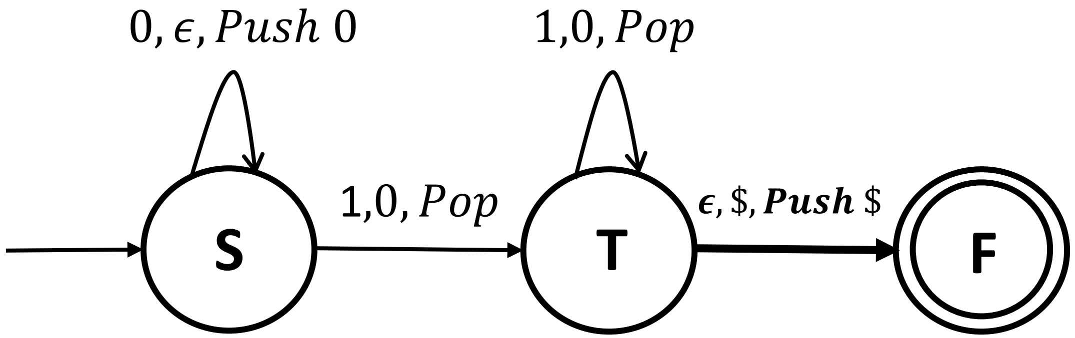


If the input symbol is 1, transition to T by ignoring the stack top completely.

If this happens at every step of the execution of the PDA, then it is as powerful as an NFA.

Pushdown Automata

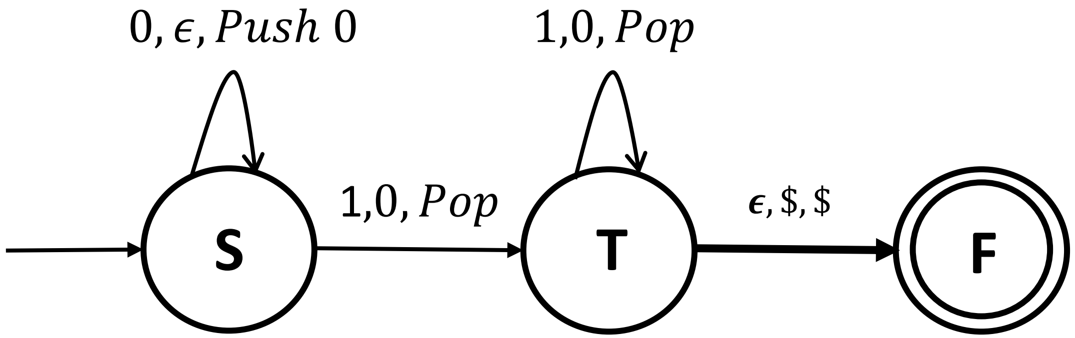
- How to represent a transition in a PDA?



If the Stack is empty, i.e. TOP = $, transition to F from T, without reading the input

Pushdown Automata

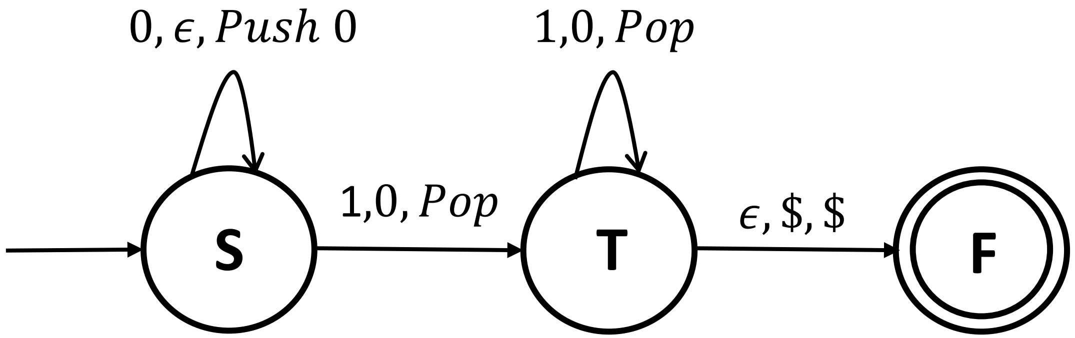
- How to represent a transition in a PDA?



If the Stack is empty, i.e. TOP = \$, transition to F from T, without reading the input

Pushdown Automata

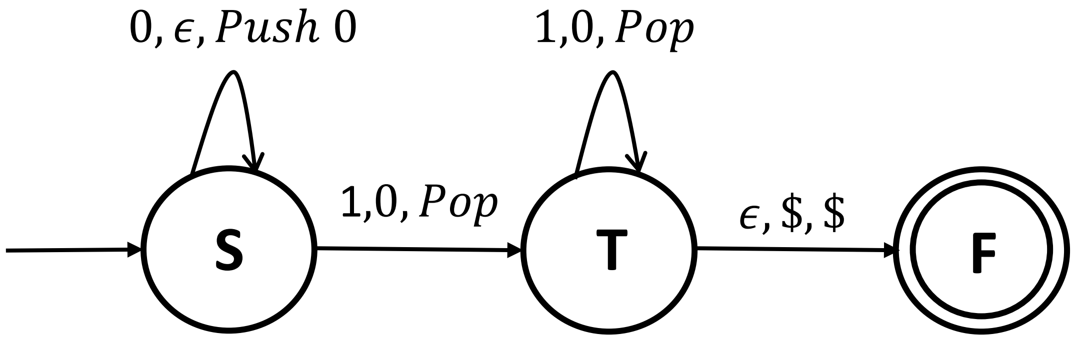
- How to represent a transition in a PDA?



What is the language accepted by this PDA?

Pushdown Automata

- How to represent a transition in a PDA?

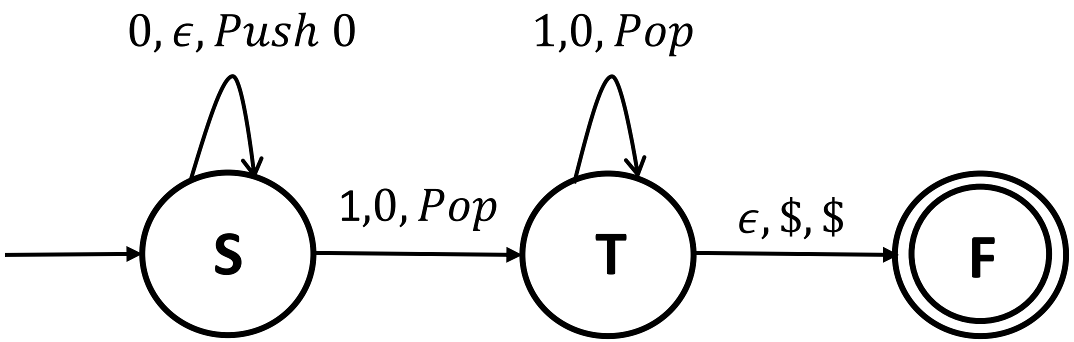


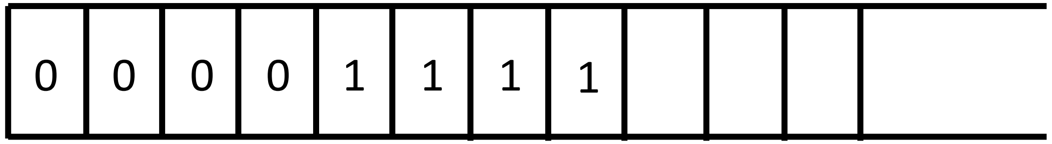
What is the language recognized by this PDA?

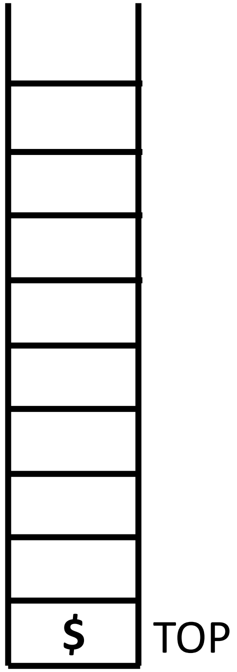
Verify that it is $\boldsymbol{L = \{0^n 1^n, n \geq 1\}}$

Pushdown Automata

What is the language recognized by this PDA?

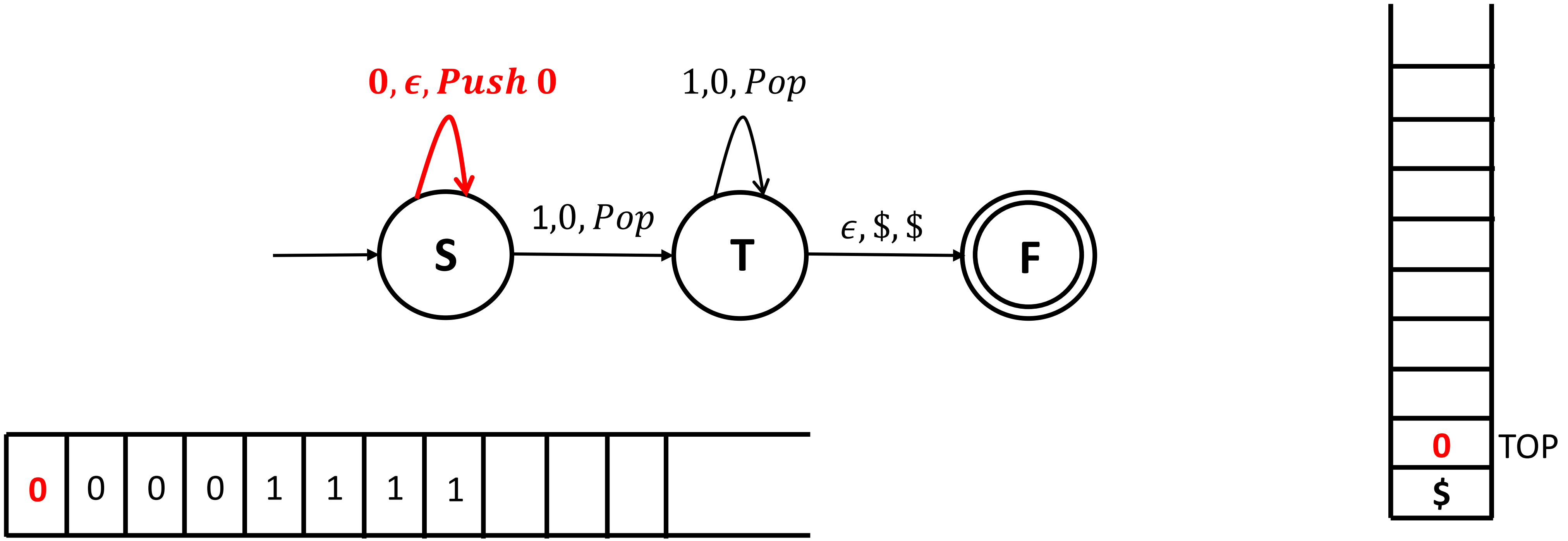






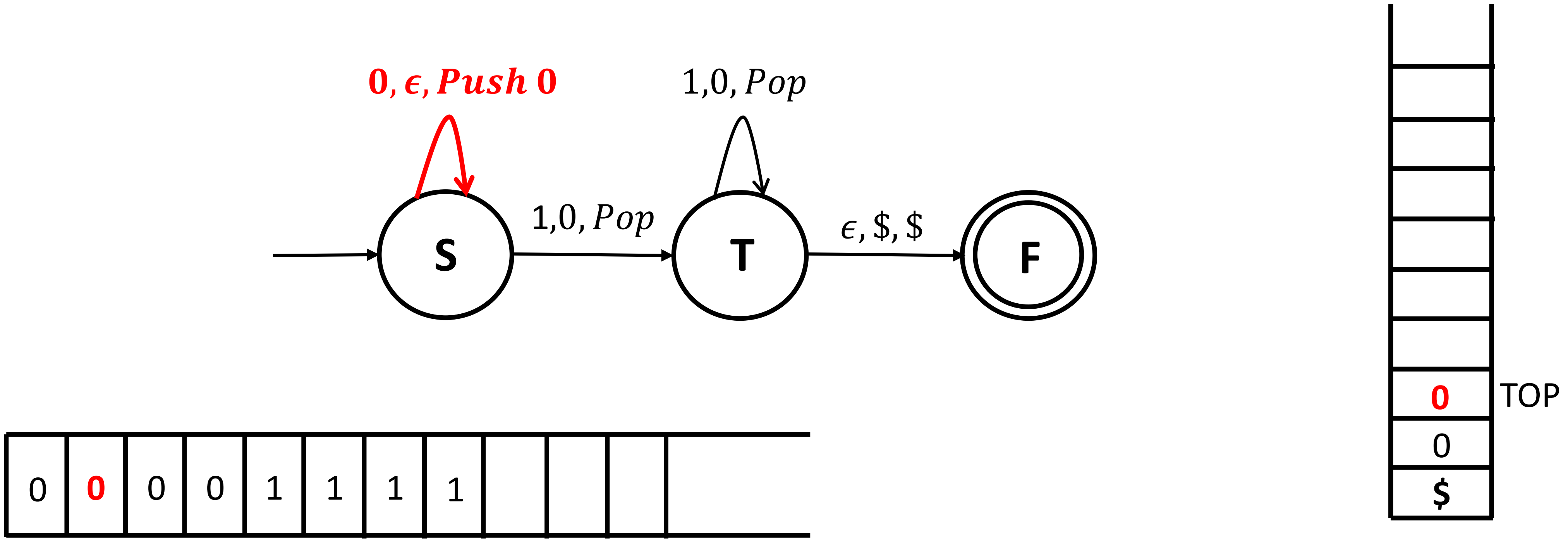
Pushdown Automata

What is the language recognized by this PDA?



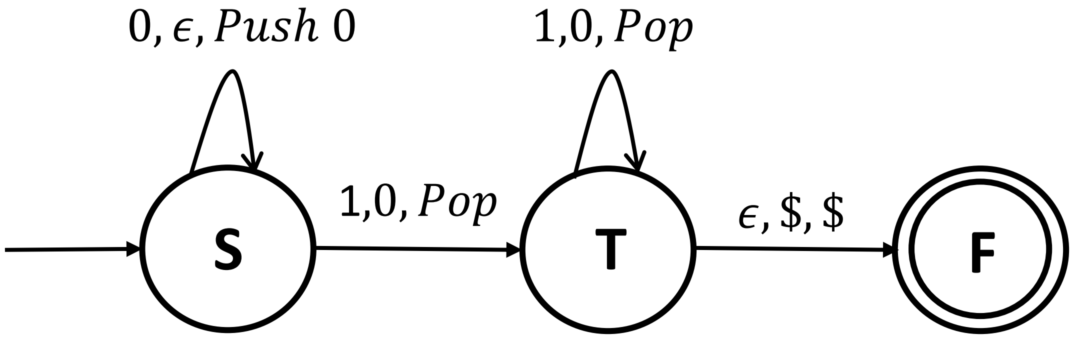
Pushdown Automata

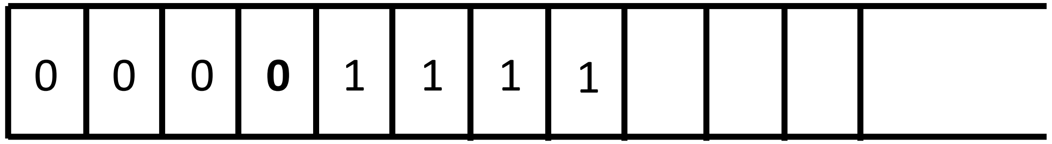
What is the language recognized by this PDA?

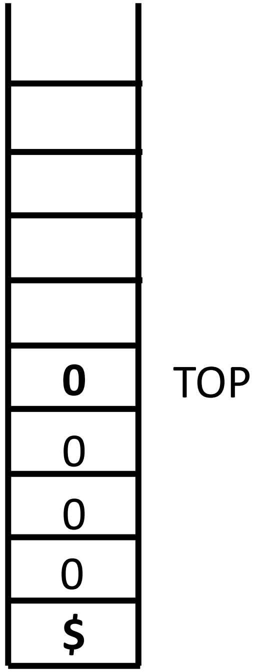


Pushdown Automata

What is the language recognized by this PDA?

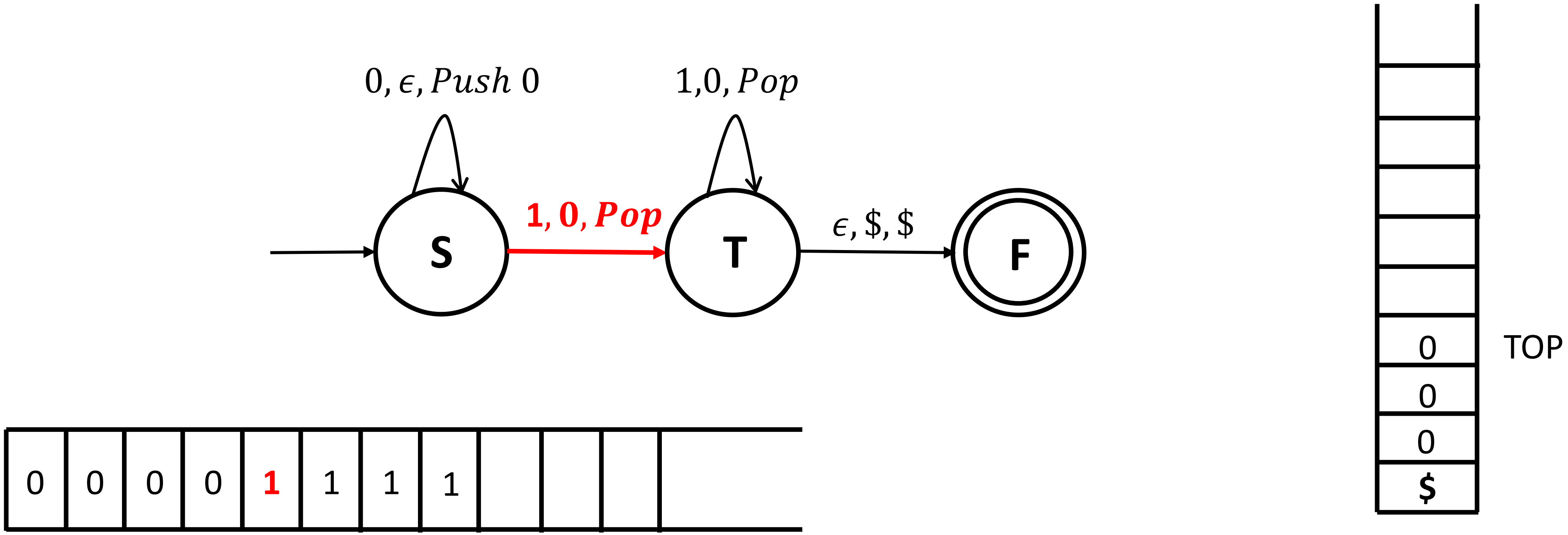






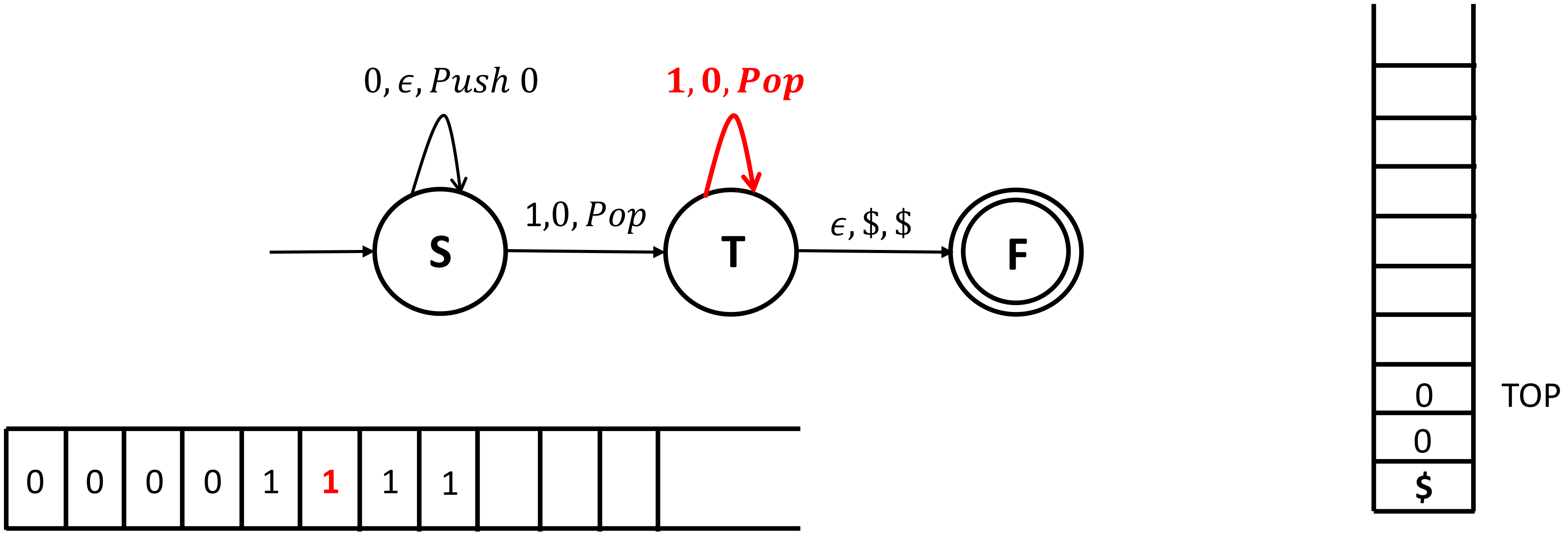
Pushdown Automata

What is the language recognized by this PDA?



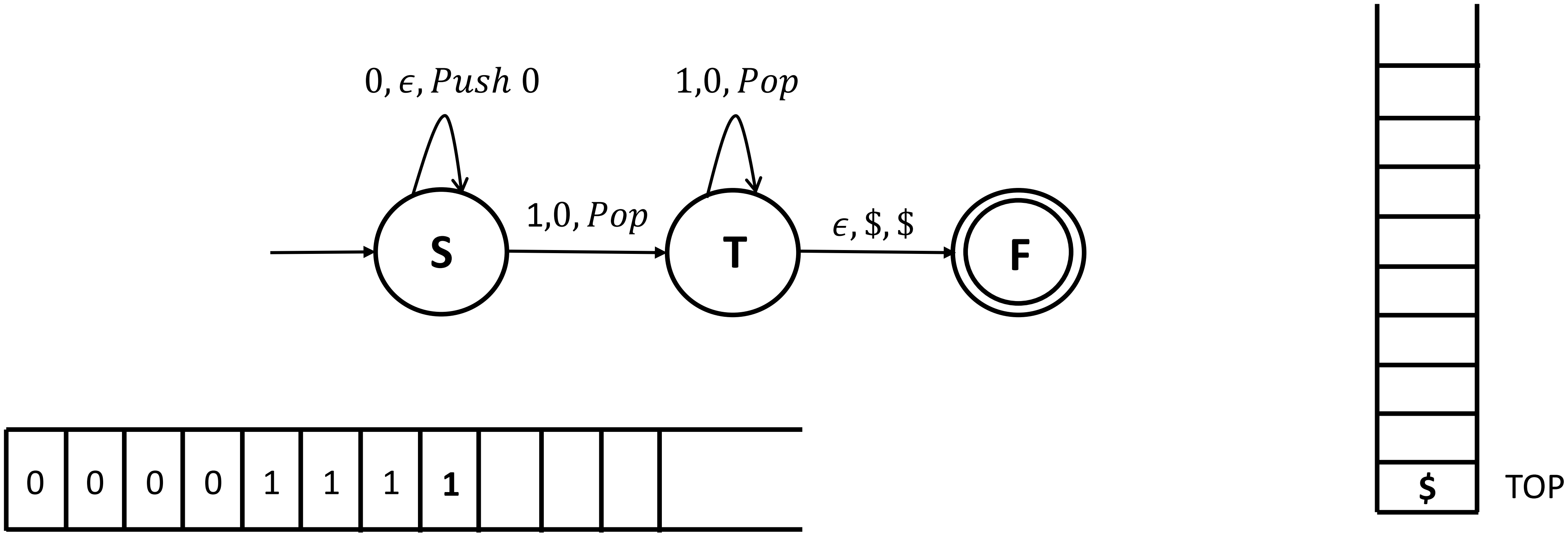
Pushdown Automata

What is the language recognized by this PDA?



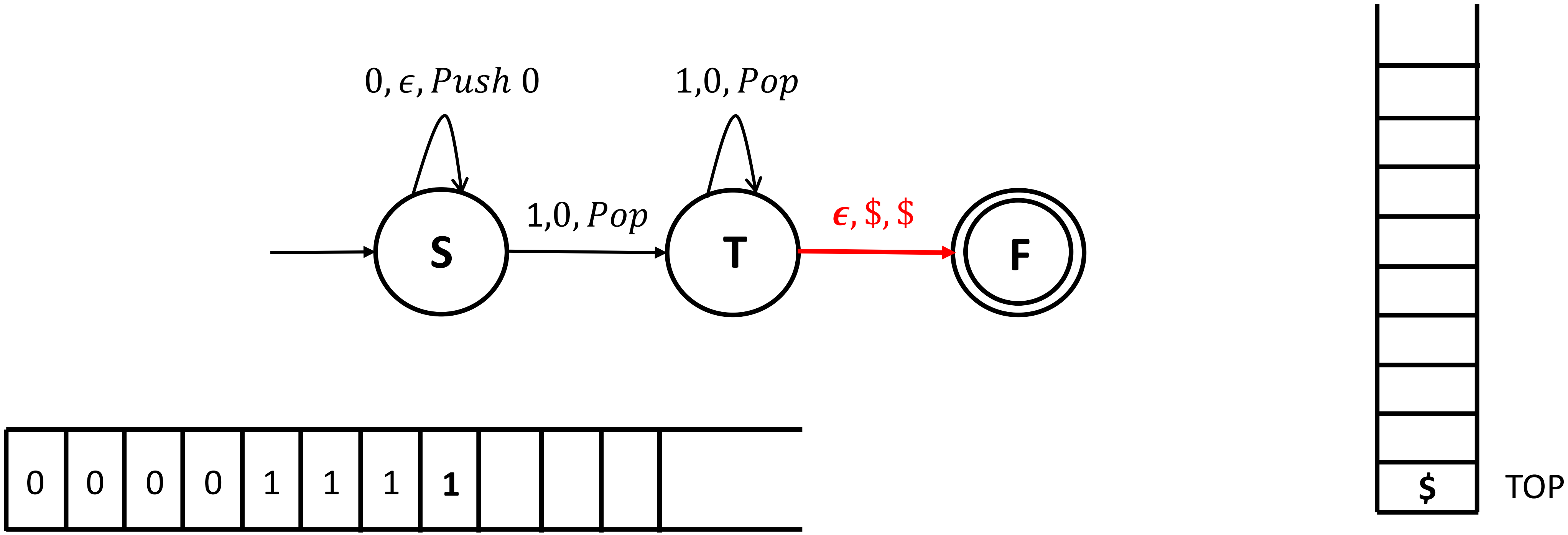
Pushdown Automata

What is the language recognized by this PDA?



Pushdown Automata

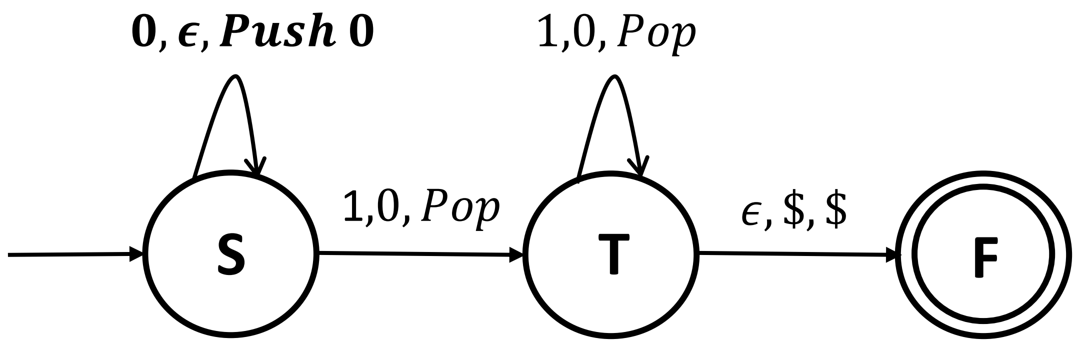
What is the language recognized by this PDA?



The language recognized by the PDA: $\boldsymbol{L = \{0^n 1^n, n \geq 1\}}$

Pushdown Automata

What is the language recognized by this PDA?

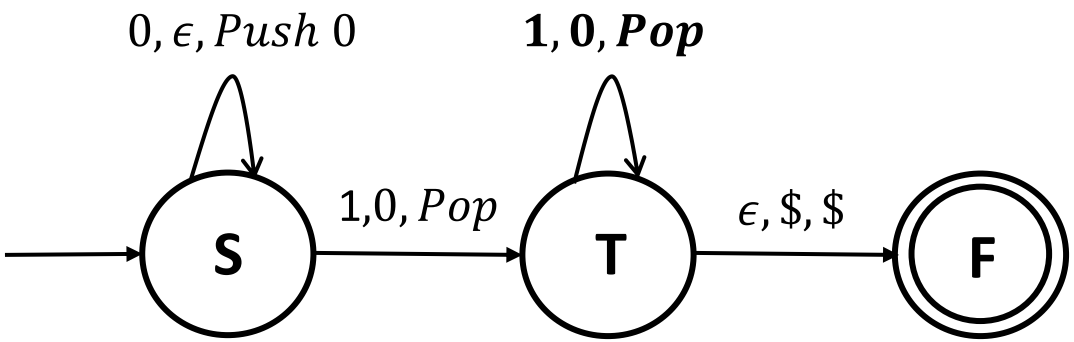


In some references (such as Sipser):

- The transitions of the PDA are labelled as "$\boldsymbol{a}, \boldsymbol{b} \rightarrow \boldsymbol{c}$", implying: If the input symbol read is $\boldsymbol{a}$, the element at the top of the stack is $\boldsymbol{b}$, then pop $\boldsymbol{b}$ and push $\boldsymbol{c}$ on to the Stack.

Pushdown Automata

What is the language recognized by this PDA?



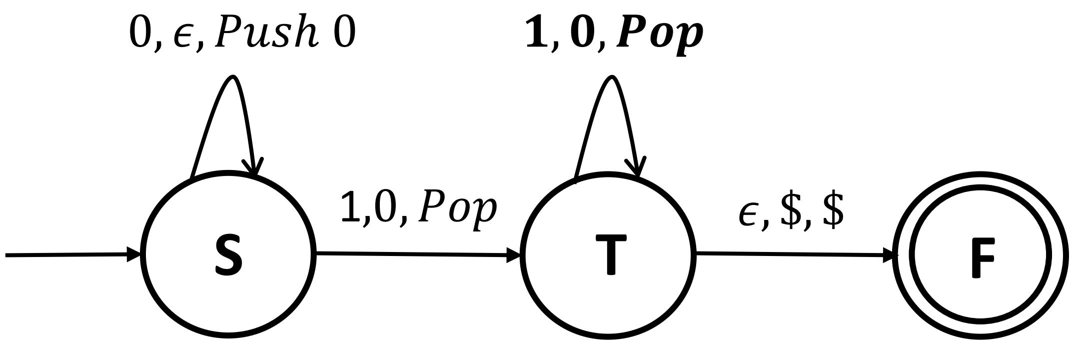
In some references (such as Sipser):

- The transitions of the PDA are labelled as "$\boldsymbol{a}, \boldsymbol{b} \rightarrow \boldsymbol{c}$", implying: If the input symbol read is $\boldsymbol{a}$, the element at the top of the stack is $\boldsymbol{b}$, then pop $\boldsymbol{b}$ and push $\boldsymbol{c}$ on to the Stack.

- The label "$\boldsymbol{a}, \boldsymbol{b} \rightarrow \boldsymbol{\epsilon}$" implies that if the input symbol is $\boldsymbol{a}$ and the the element at the top of the stack is $\boldsymbol{b}$, then **pop**.

Pushdown Automata

What is the language recognized by this PDA?



In some references (such as Sipser):

- The transitions of the PDA are labelled as "$\boldsymbol{a, b \to c}$", implying: If the input symbol read is $\boldsymbol{a}$, the element at the top of the stack is $\boldsymbol{b}$, then pop $\boldsymbol{b}$ and push $\boldsymbol{c}$ on to the Stack.
- The label "$\boldsymbol{a, b \to \epsilon}$" implies that if the input symbol is $\boldsymbol{a}$ and the the element at the top of the stack is $\boldsymbol{b}$, then **pop**.
- The symbol signifying the bottom of the Stack \$ is pushed at the very beginning.

Pushdown Automata

Formally, a PDA M is a 6-tuple $(\mathrm{Q}, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Pushdown Automata

Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j

Pushdown Automata

Formally, a PDA M is a 6-tuple $(\mathrm{Q}, \Sigma, \Gamma, \delta, q_0, F\ )$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j

- $\boldsymbol{\delta(q_i, a, \epsilon) = (q_j, c)}$:

Pushdown Automata

Formally, a PDA M is a 6-tuple $(\mathrm{Q}, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** $[\ \Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}\ ]$
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j

- $\boldsymbol{\delta(q_i, a, \epsilon) = (q_j, c)}$: If the input symbol read is a, then push c onto the stack and transition from q_i to q_j

Pushdown Automata

Formally, a PDA M is a 6-tuple $(\mathrm{Q}, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j

- $\boldsymbol{\delta(q_i, a, \epsilon) = (q_j, c)}$: If the input symbol read is a, then push c onto the stack and transition from q_i to q_j

- $\boldsymbol{\delta(q_i, a, b) = (q_j, \epsilon)}$:

Pushdown Automata

Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F\ )$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** $[\ \Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}\ ]$
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j
- $\boldsymbol{\delta(q_i, a, \epsilon) = (q_j, c)}$: If the input symbol read is a, then push c onto the stack and transition from q_i to q_j
- $\boldsymbol{\delta(q_i, a, b) = (q_j, \epsilon)}$: If the input symbol read is a, and the stack top = b, then pop b and transition from q_i to q_j
- $\boldsymbol{\delta(q_i, \epsilon, \$) = (q_j, \$)}$:

Pushdown Automata

Formally, a PDA M is a 6-tuple $(\mathrm{Q}, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j

- $\boldsymbol{\delta(q_i, a, \epsilon) = (q_j, c)}$: If the input symbol read is a, then push c onto the stack and transition from q_i to q_j

- $\boldsymbol{\delta(q_i, a, b) = (q_j, \epsilon)}$: If the input symbol read is a, and the stack top = b, then pop b and transition from q_i to q_j

- $\boldsymbol{\delta(q_i, \epsilon, \$) = (q_j, \$)}$: Transition from q_i to q_j if the stack is empty.

Pushdown Automata

Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j

- If the input symbol read is a and the stack top = a, then Push a and remain at q_i : ?

Pushdown Automata

Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

Transition function:

- $\boldsymbol{\delta(q_i, a, b) = (q_j, c)}$: If the input symbol read is a and the stack top = b, then pop b, push c onto the stack and transition from q_i to q_j

- If the input symbol read is a and the stack top = a, then Push a and remain at q_i : $\boldsymbol{\delta(q_i, a, a) = (q_i, aa)}$

Pushdown Automata

Formally, a PDA M is a 6-tuple $(\mathrm{Q}, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** $[\, \Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}\,]$
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

- The Language of the PDA P is the set of strings the PDA accepts, i.e.

$$L = \{w | P \text{ accepts } w\}$$

- If $\mathcal{L}(P) = L$, then the PDA P recognizes L

Pushdown Automata

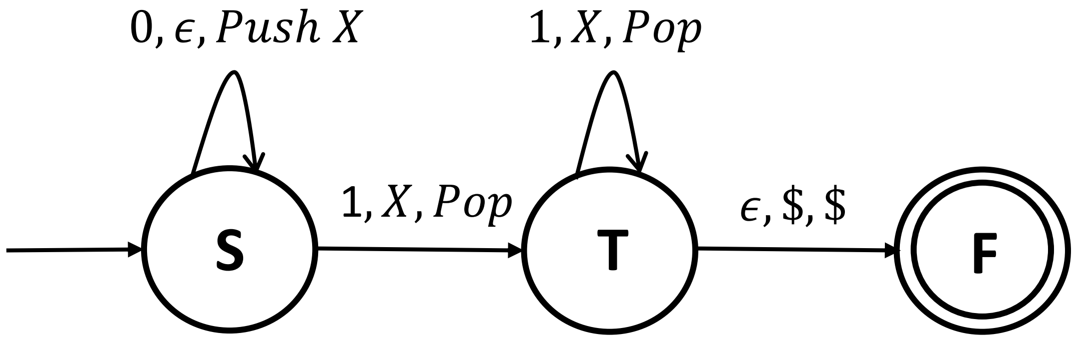
Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

- The Language of the PDA P is the set of strings the PDA accepts, i.e.

$$L = \{w | P \text{ accepts } w\}$$

- If $\mathcal{L}(P) = L$, then the PDA P recognizes L
- Stack alphabet **can be different** from the input alphabet



Pushdown Automata

Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

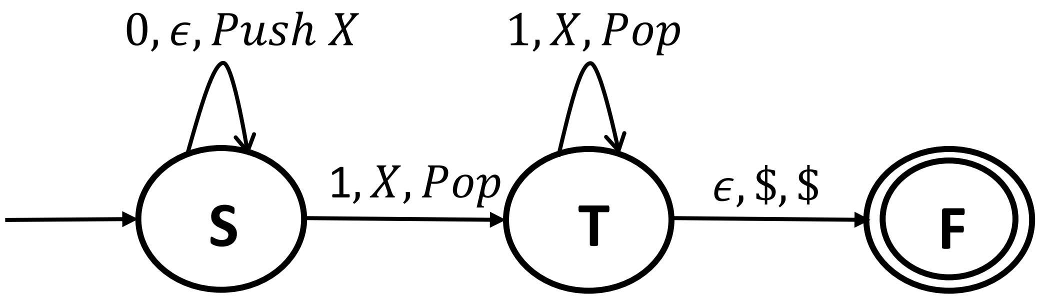
- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

- The Language of the PDA P is the set of strings the PDA accepts, i.e.

$$L = \{w | P \text{ accepts } w\}$$

- If $\mathcal{L}(P) = L$, then the PDA P recognizes L

- Stack alphabet **can be different** from the input alphabet



$$\delta(S, 0, \epsilon) = (S, X)$$

Pushdown Automata

Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

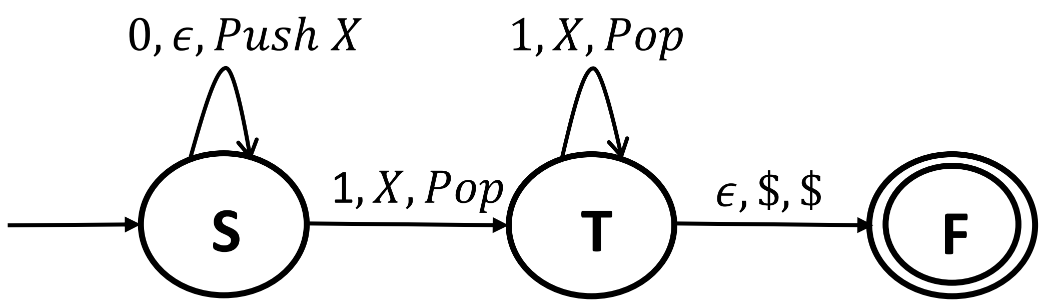
- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

- The Language of the PDA P is the set of strings the PDA accepts, i.e.

$$L = \{w | P \text{ accepts } w\}$$

- If $\mathcal{L}(P) = L$, then the PDA P recognizes L

- Stack alphabet **can be different** from the input alphabet



$$\delta(S, 0, \epsilon) = (S, X)$$
$$\delta(S, 1, X) = (T, \epsilon)$$

Pushdown Automata

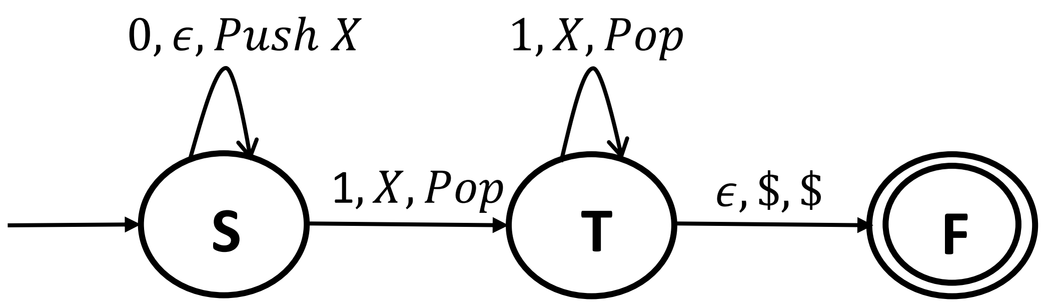
Formally, a PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets.***
- Γ is the ***Stack alphabet***
- $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \mapsto \mathcal{P}(Q \times \Gamma_\epsilon)$ is the ***transition function*** [$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$]
- $q_0 \in Q$ is the ***start state***.
- $F \subseteq Q$ is the set of ***accepting states***.

- The Language of the PDA P is the set of strings the PDA accepts, i.e.

$$L = \{w | P \text{ accepts } w\}$$

- If $\mathcal{L}(P) = L$, then the PDA P recognizes L
- Stack alphabet **can be different** from the input alphabet



$$\delta(S, 0, \epsilon) = (S, X)$$
$$\delta(S, 1, X) = (T, \epsilon)$$
$$\delta(T, 1, X) = (T, \epsilon)$$
$$\delta(T, \epsilon, \$) = (F, \$)$$

Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \ | \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
    - The PDA does this non-deterministically (by taking ϵ transitions).

Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \; |\boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).

- The above intuition is applicable for even length palindromes of the form ww^R.

- What about odd length palindromes?
  - Non-determinism to the rescue once again

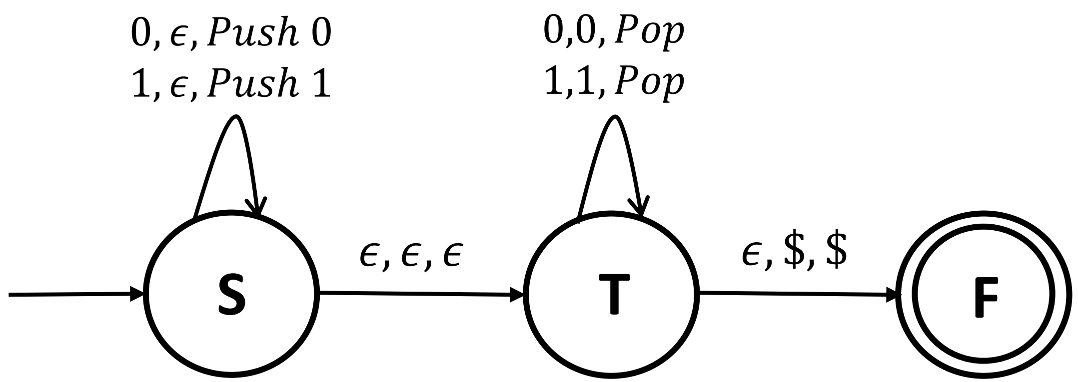
Pushdown Automata

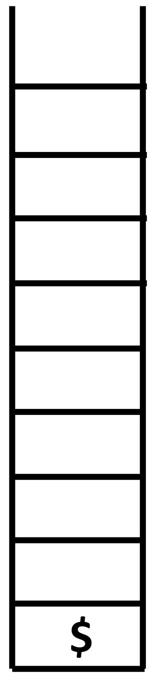
Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
    - The PDA does this non-deterministically (by taking ϵ transitions).





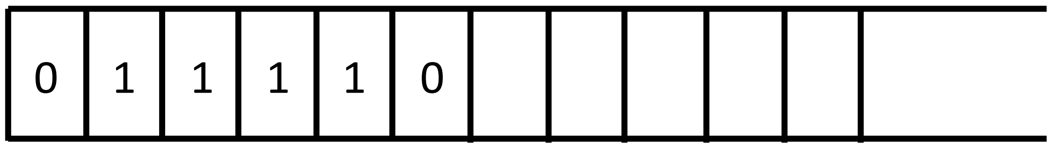
Pushdown Automata

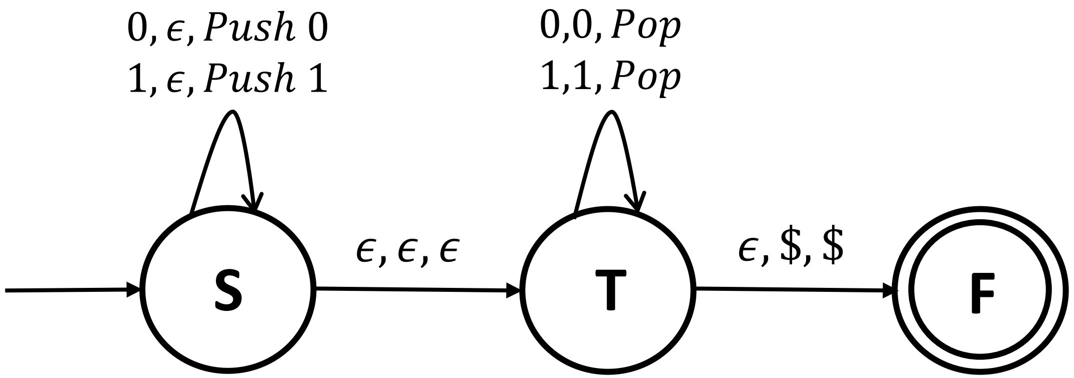
Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

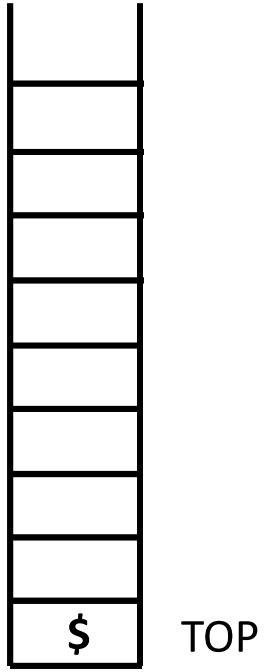
Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
    - The PDA does this non-deterministically (by taking ϵ transitions).







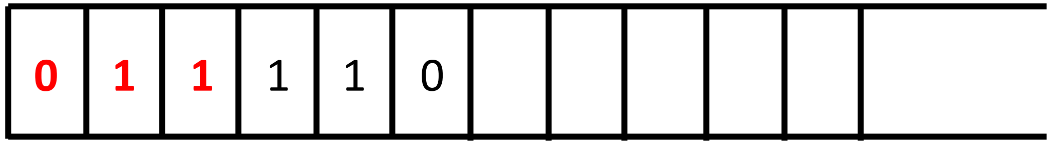
Pushdown Automata

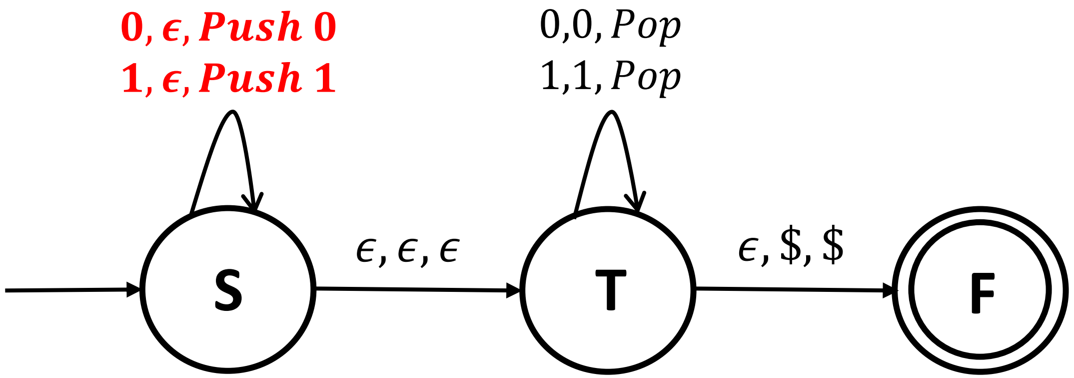
Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

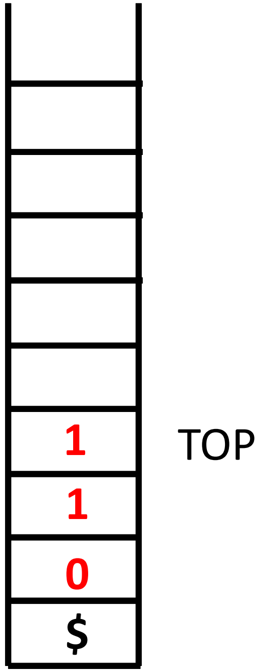
Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).







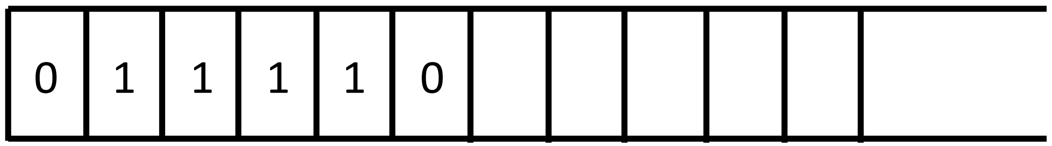
Pushdown Automata

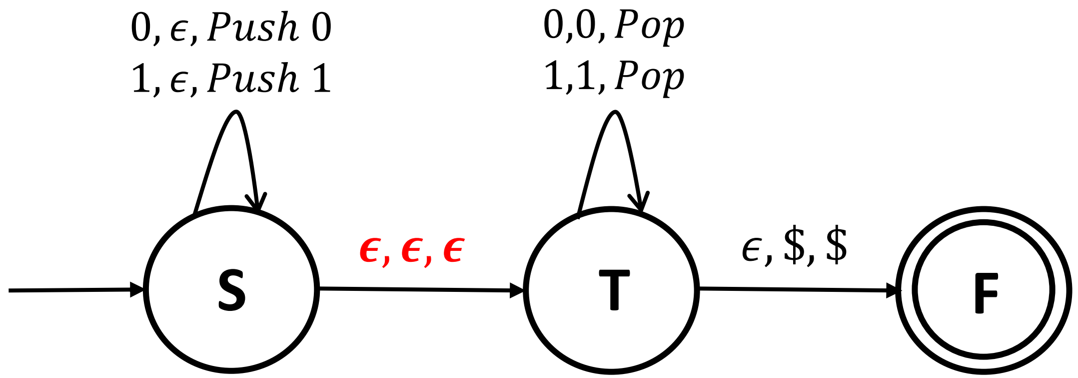
Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

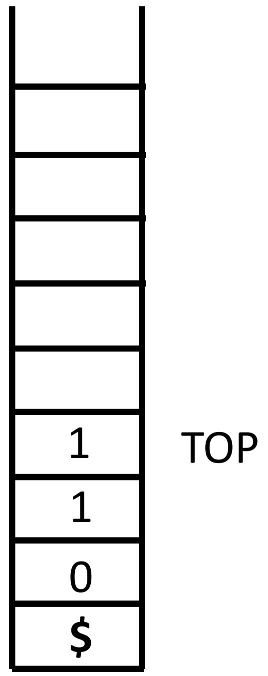
Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).







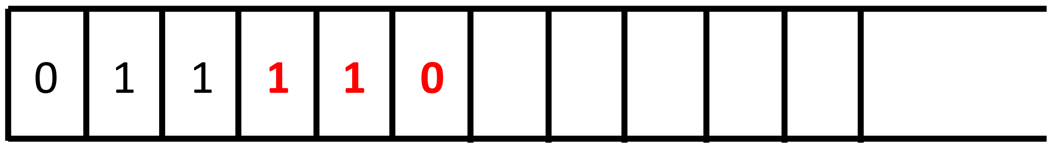
Pushdown Automata

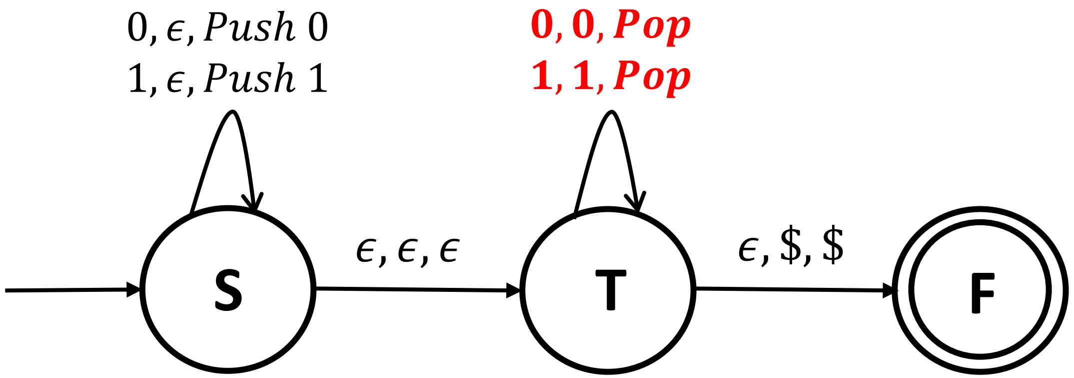
Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \ |\boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

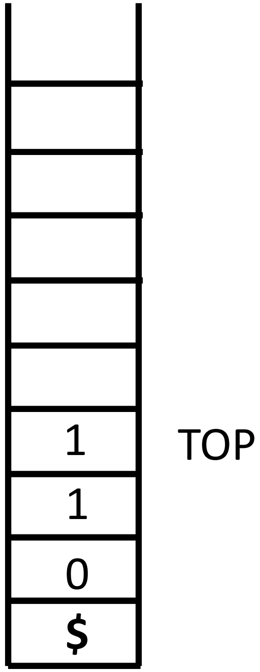
Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).







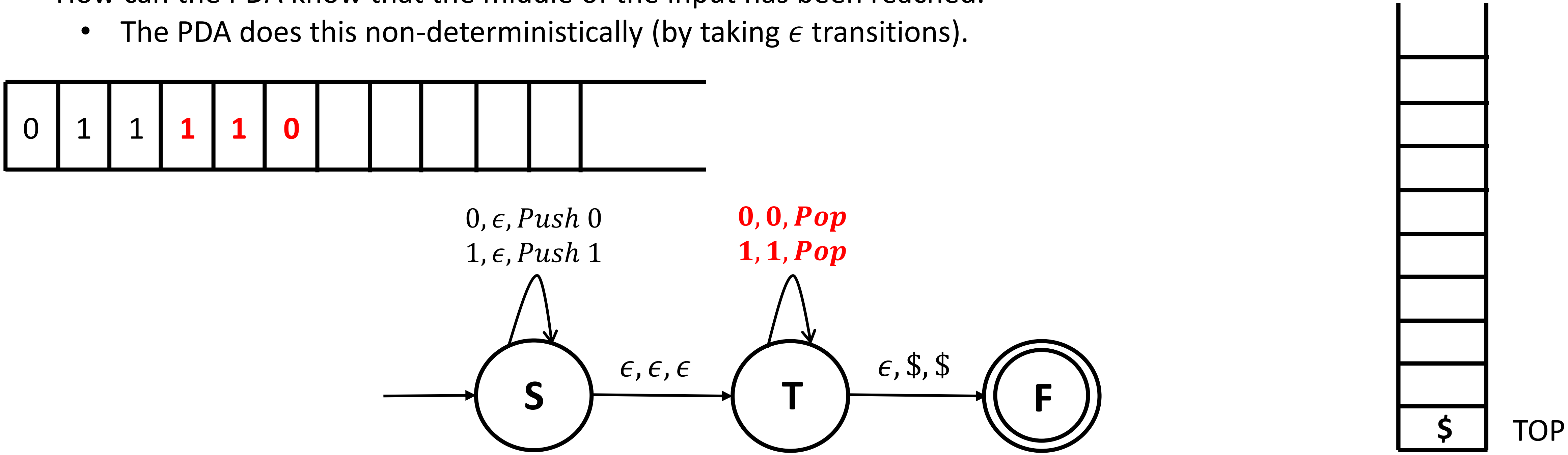
Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).



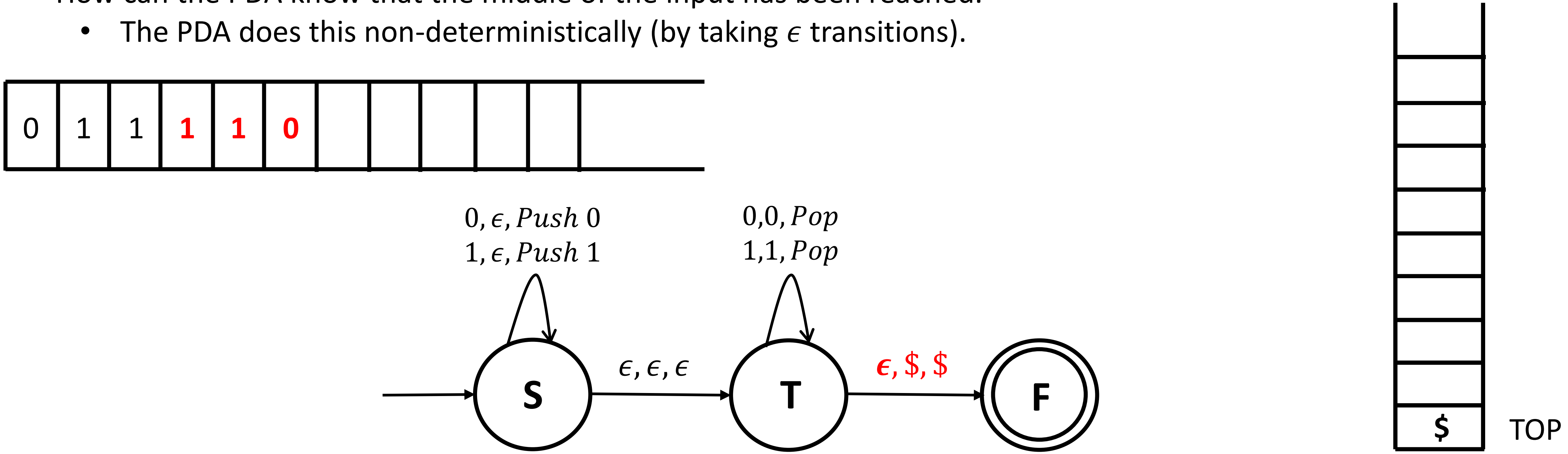
Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid$ **$\boldsymbol{w}$ is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).



Pushdown Automata

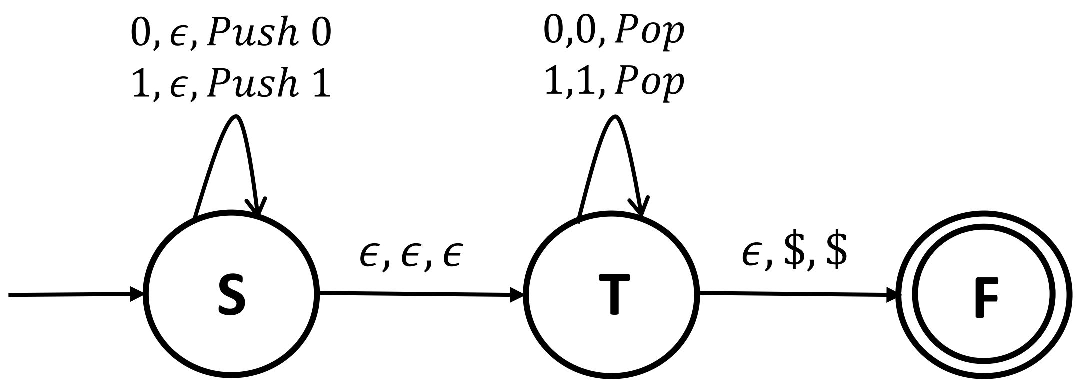
Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid$ $\boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
    - The PDA does this non-deterministically (by taking ϵ transitions).

- What about odd length palindromes?



Recognizes even length palindromes of the form: ww^R

Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

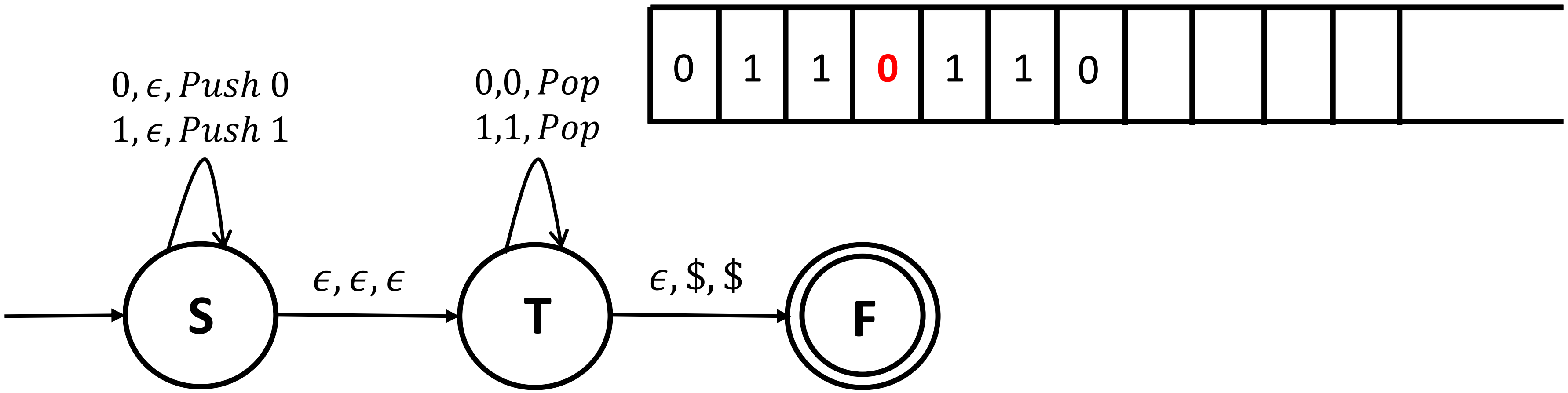
Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).

- What about odd length palindromes?

Odd length palindromes are of the form $\boldsymbol{wcw^R}$, such that $\boldsymbol{c} \in \boldsymbol{\Sigma}$



Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid$ **$\boldsymbol{w}$ is a Palindrome**$\}$. Design a PDA P that recognizes L.

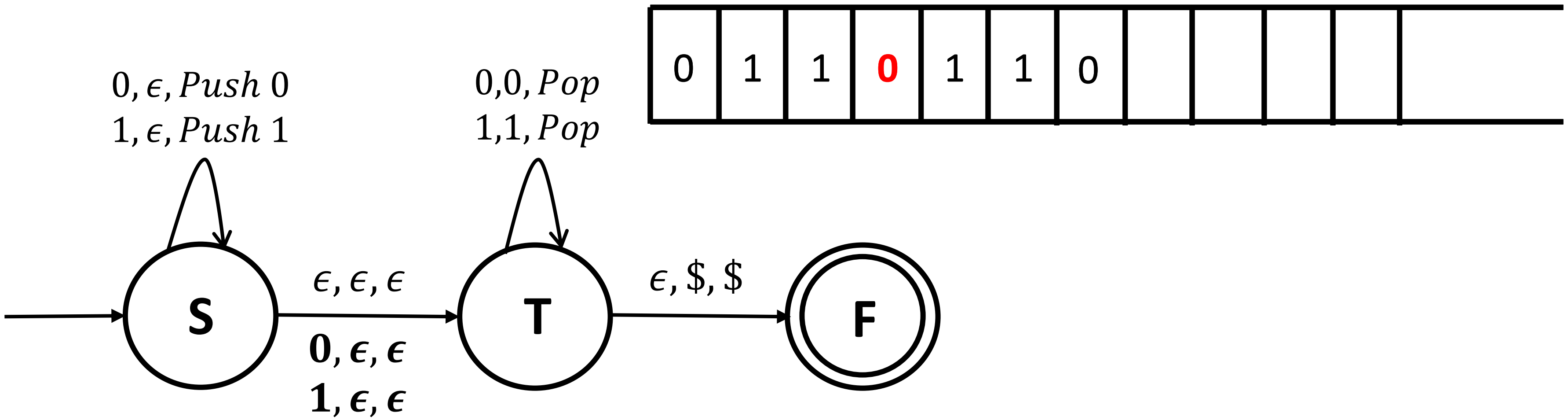
Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).

- What about odd length palindromes?

Odd length palindromes are of the form $\boldsymbol{wcw^R}$, such that $\boldsymbol{c \in \Sigma}$



Pushdown Automata

Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

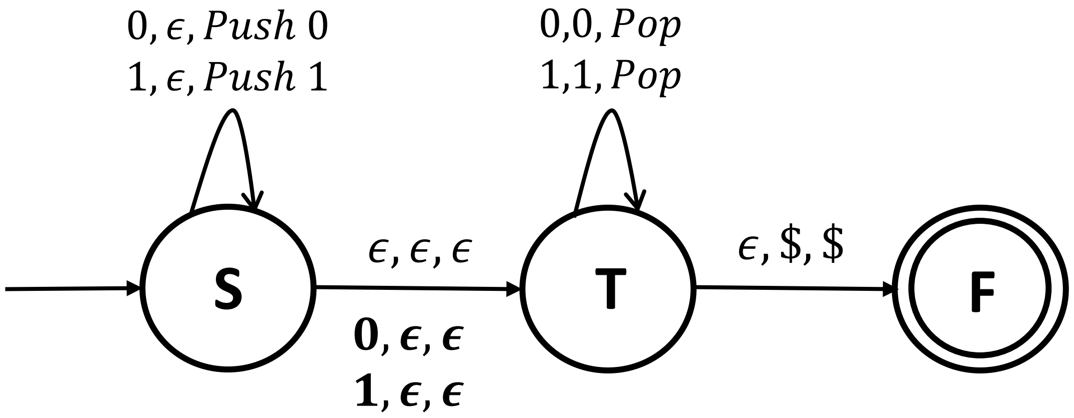
Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).

- What about odd length palindromes?

The transitions $\boldsymbol{0, \epsilon, \epsilon}$ **and** $\boldsymbol{1, \epsilon, \epsilon}$ **allow the PDA to consume one symbol** and then begin matching what it has encountered thus far.



Pushdown Automata

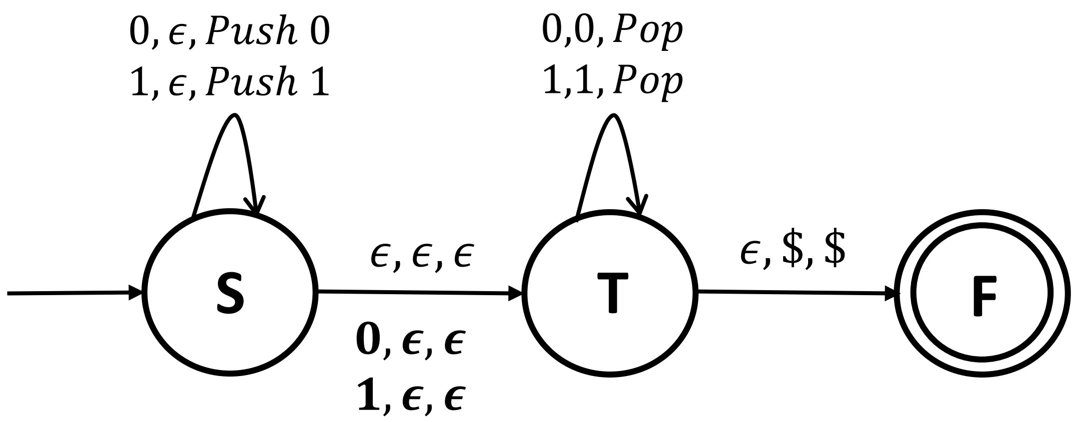
Let $\Sigma = \{0,1\}$ consider the language $\boldsymbol{L} = \{\boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{w}$ **is a Palindrome}**. Design a PDA P that recognizes L.

Intuition

- Push first half of the input string onto the stack.
- Verify that the second half of the symbols match the first half: Keep Popping the stack until the end of the input.

- How can the PDA know that the middle of the input has been reached.
  - The PDA does this non-deterministically (by taking ϵ transitions).

- What about odd length palindromes?



The transitions $\boldsymbol{0}, \boldsymbol{\epsilon}, \boldsymbol{\epsilon}$ **and** $\boldsymbol{1}, \boldsymbol{\epsilon}, \boldsymbol{\epsilon}$ **allow the PDA to consume one symbol** and then begin matching what it has encountered thus far.

This allows the PDA to **recognize strings of the form:** $\boldsymbol{\omega c w^R}$, where the aforementioned transitions non-deterministically guessed $c \in \{0,1\}$

Thank You!