Pushdown Automata (Introduction)

A Pushdown Automata (PDA) is a way to implement a Context Free Grammar in a similar way we design Finite Automata for Regular Grammar

-> It is more powerful than FSM

-> FSM has a very limited memory but PDA has more memory

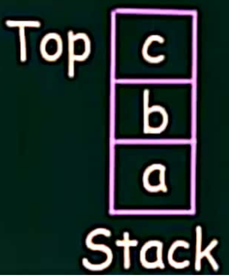
-> PDA = Finite State Machine + A Stack

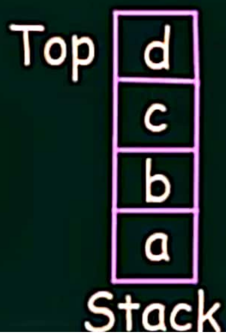
A stack is a way we arrange elements one on top of another

A stack does two basic operations:

PUSH: A new element is added at the Top of the stack

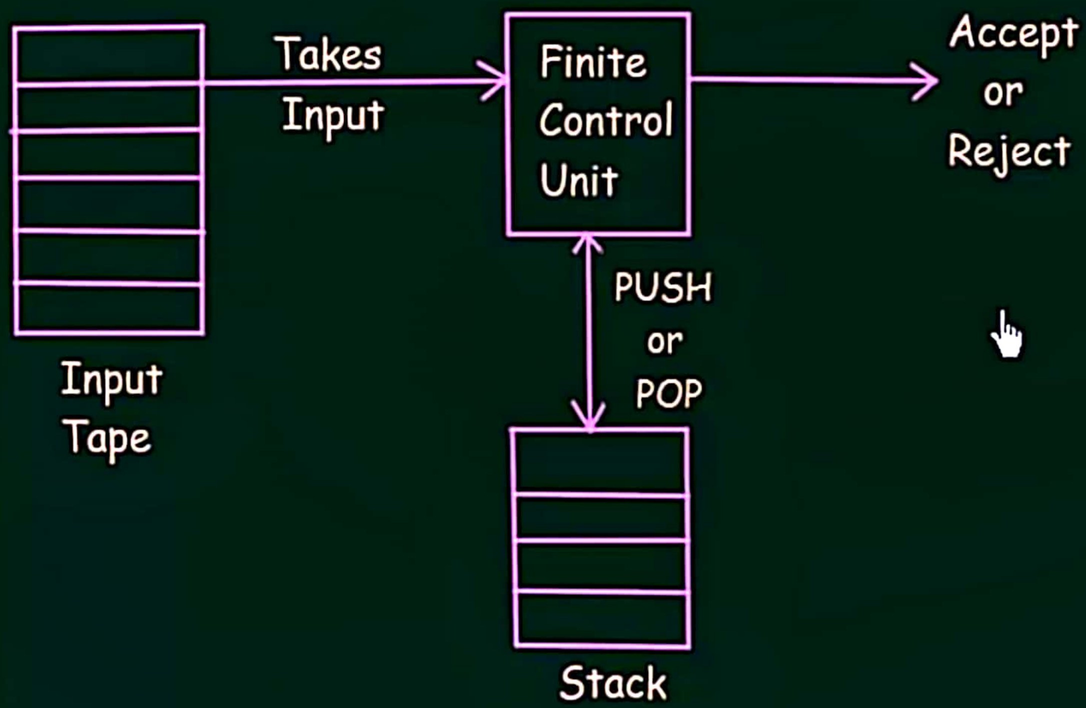
POP: The Top element of the stack is read and removed





A Pushdown Automata has 3 components:

1) An input tape
2) A Finite Control Unit
3) A Stack with infinite size



Pushdown Automata (Formal Definition)

A Pushdown Automata is formally defined by 7 Tuples as shown below:

$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where,

Q = A finite set of States

Σ = A finite set of Input Symbols

Γ = A finite Stack Alphabet

δ = The Transition Function

q_0 = The Start State

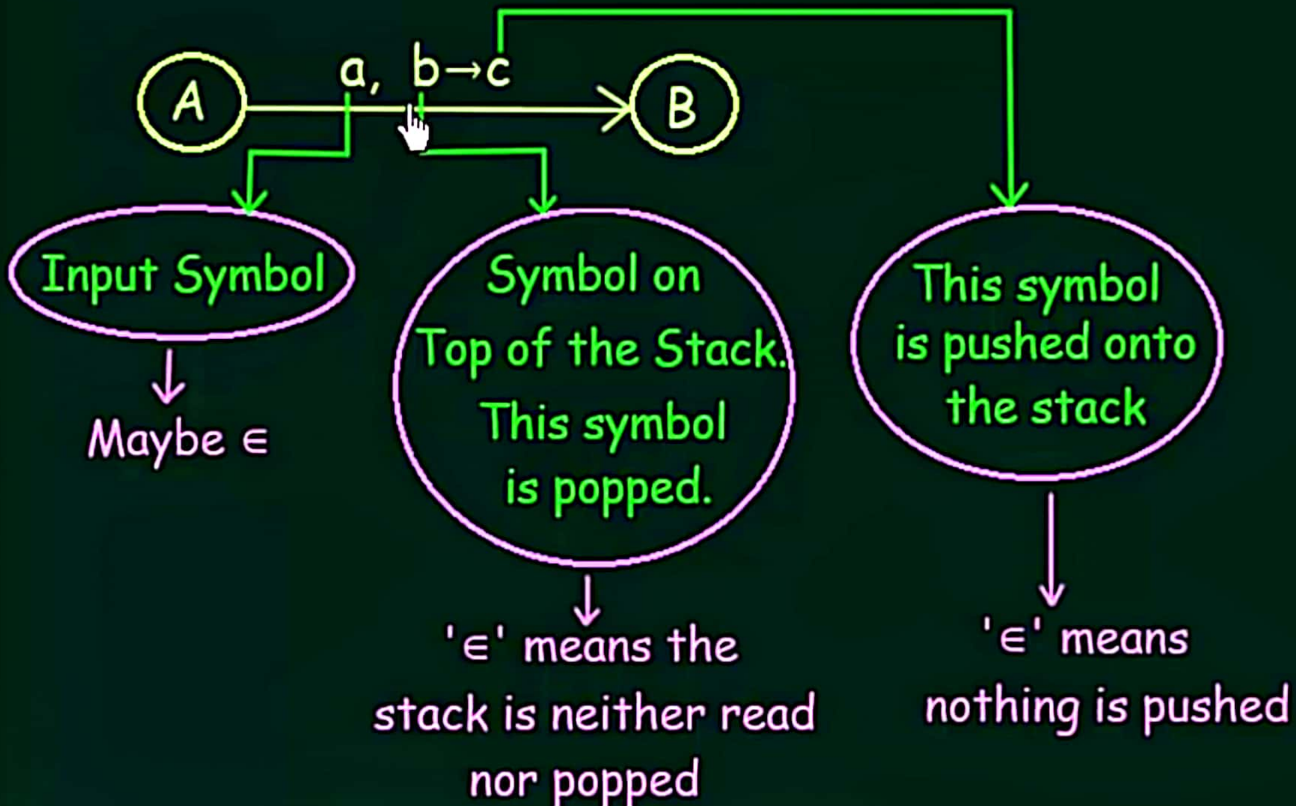
z_0 = The Start Stack Symbol

F = The set of Final / Accepting States

Pushdown Automata (Graphical Notation)

Finite State Machine

Pushdown Automata



CFG to PDA

Rule 1 : For each Variable A

$$\delta(q, \varepsilon, A) = (q, \beta)$$

where $A \rightarrow \beta$ is a production of Grammar.

Rule 2 : For each Terminal "a"

$$\delta(q, a, a) = (q, \varepsilon)$$

Example :-

$S \rightarrow 0S1 \mid 00 \mid 11$

The equivalent PDA for the given grammar.

The equivalent PDA for the given grammar.

$S \to 0S1 \mid 00 \mid 11$

$\delta(q, \varepsilon, S) = (q, 0S1), (q, 00), (q, 11)$ ... ①

$\delta(q, 0, 0) = (q, \varepsilon)$ ... ②

$\delta(q, 1, 1) = (q, \varepsilon)$ ... ③

$S \to 0S1$

$\quad \to 0111$

0111

$\delta(q, 0111, S)$ using ①

$\delta(q, 0111, \cancel{0}S1)$ ②

$\delta(q, 111, S1)$ ①

$\delta(q, 111, \cancel{1}11)$ ③

$\delta(q, 11, \cancel{1}1)$ ③

$\delta(q, 1, \cancel{1})$ ③

$\delta(q, \varepsilon, \varepsilon)$ Accept.

Rule 1 For each <u>Variable</u> A

$$\delta(q, \varepsilon, A) = (q, \beta)$$ where $A \rightarrow \beta$ is a production of Grammar.

Rule 2 For each <u>terminal</u> 'a'

$$\delta(q, a, a) = (q, \varepsilon)$$

Example:-

$S \rightarrow 0BB$

$B \rightarrow 0S \mid 1S \mid 0$

Test 010^4

The equivalent PDA for the given grammar.

$$\delta(q, \varepsilon, S) = (q, 0BB)$$

$$\delta(q, \varepsilon, B) = (q, 0S), (q, 1S), (q, 0)$$

$$\delta(q, 0, 0) = (q, \varepsilon)$$

$$\delta(q, 1, 1) = (q, \varepsilon)$$

Example:-

$S \to 0BB$

$B \to 0S \mid 1S \mid 0$

Test 010^4

010000

$\delta(q, \varepsilon, S) = (q, 0BB)$ .... (1)

$\delta(q, \varepsilon, B) = (q, 0S), (q, 1S), (q, 0)$ ... (2)

$\delta(q, 0, 0) = (q, \varepsilon)$ .... (3)

$\delta(q, 1, 1) = (q, \varepsilon)$ .... (4)

$\delta(q, 010000, S)$ Using Tran (1)

$\delta(q, 010000, \cancel{0}BB)$ (3)

$\delta(q, 10000, BB)$ (2)

$\delta(q, 10000, \cancel{1}SB)$ (4)

$\delta(q, 0000, SB)$ (1)

$\delta(q, 0000, \cancel{0}BBB)$ (3)

$\delta(q, 000, BBB)$ (2)

$(q, 00, BB)$ (2)

$(q, 00, \cancel{0}B)$ (3)

$(q, 0, B)$ (2)

$(q, 0, \cancel{0})$ (3)

$(q, \varepsilon, \varepsilon)$ Accept.

Grammar

$S \to aSTb$

$S \to b$

$T \to Ta$

$T \to \lambda$

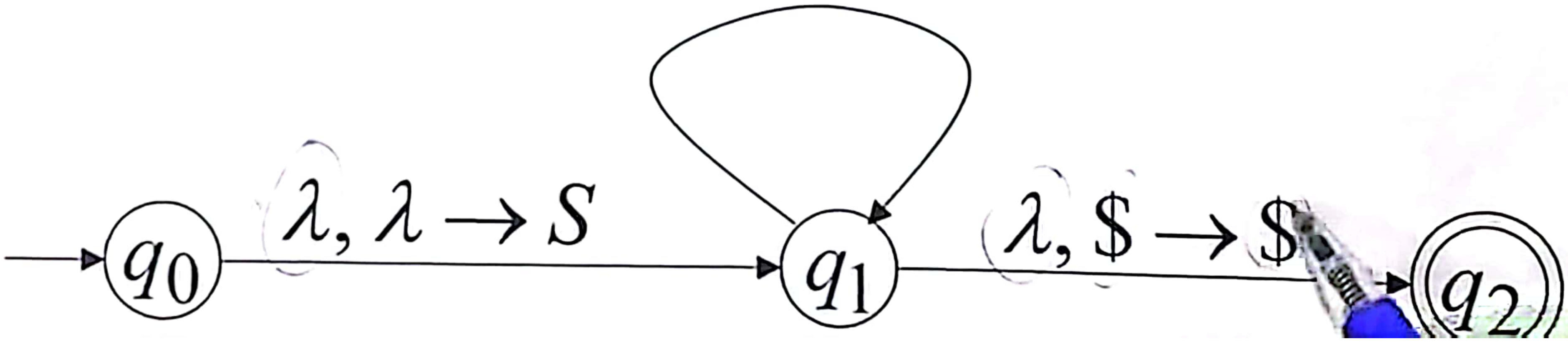
PDA

$\lambda, S \to aSTb$

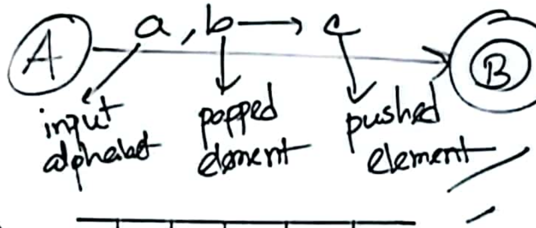
$\lambda, S \to b$

$\lambda, T \to Ta$ $\quad$ $a, a \to \lambda$

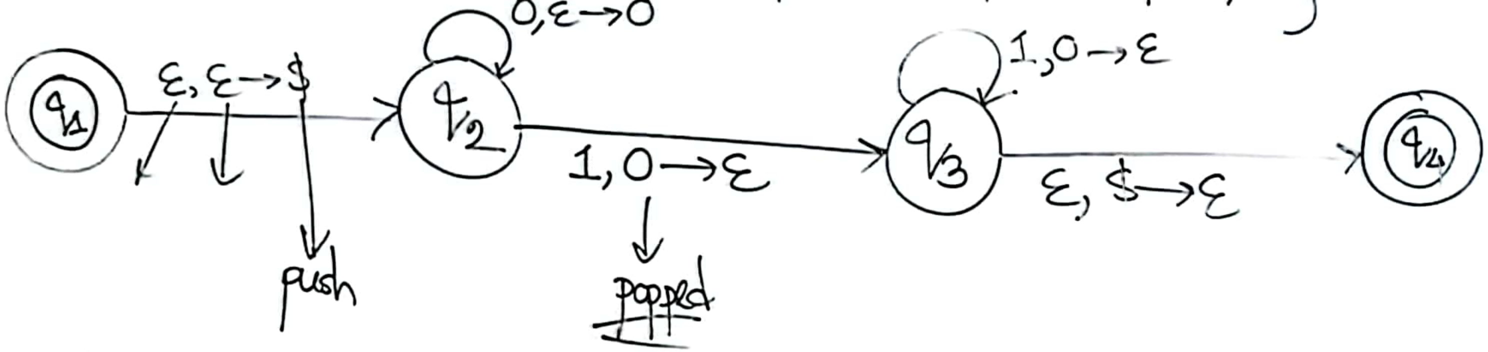
$\lambda, T \to \lambda$ $\quad$ $b, b \to \lambda$

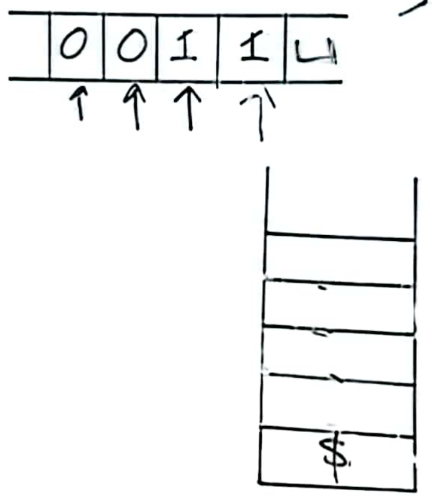


Push Down Automata (PDA)



* Construct a PDA that accepts, $L = \{ 0^n 1^n \mid n \geq 0 \}$





$(q_1, \varepsilon, \varepsilon) \rightarrow (q_2, \$)$

$(q_2, 0, \varepsilon) \rightarrow (q_2, 0)$

$(q_2, 1, 0) \rightarrow (q_3, \varepsilon)$

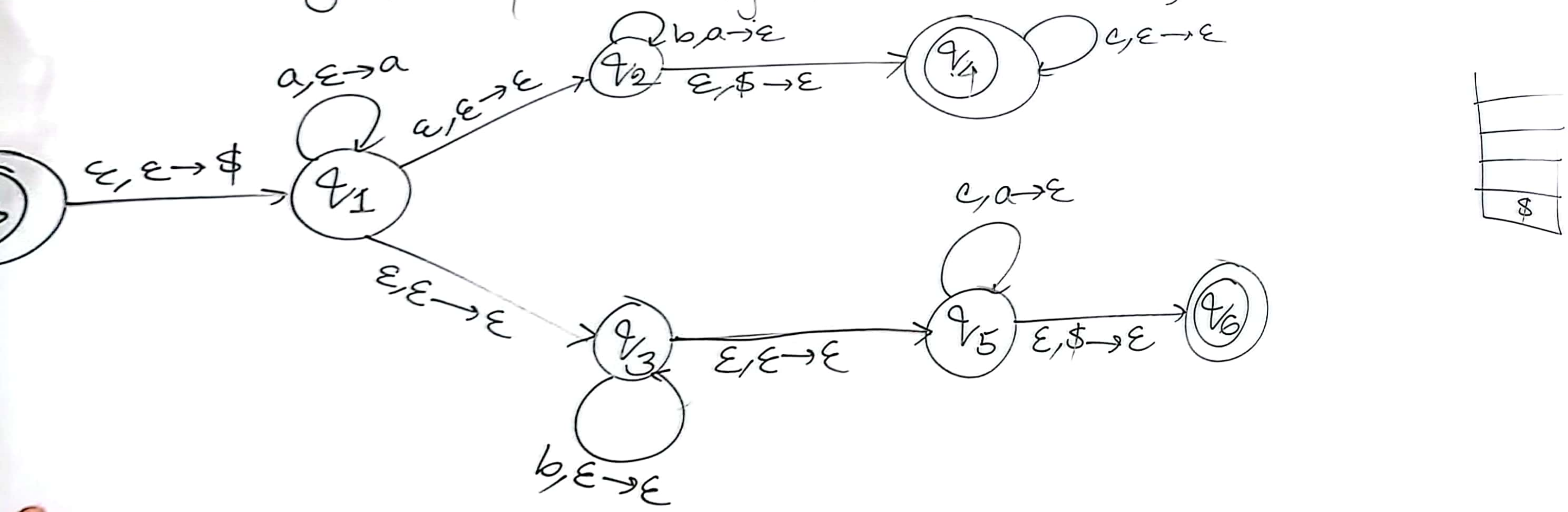
$(q_3, 1, 0) \rightarrow (q_3, \varepsilon)$

$(q_3, \varepsilon, \$) \rightarrow (q_4, \varepsilon)$

Push Down Automata (PDA)

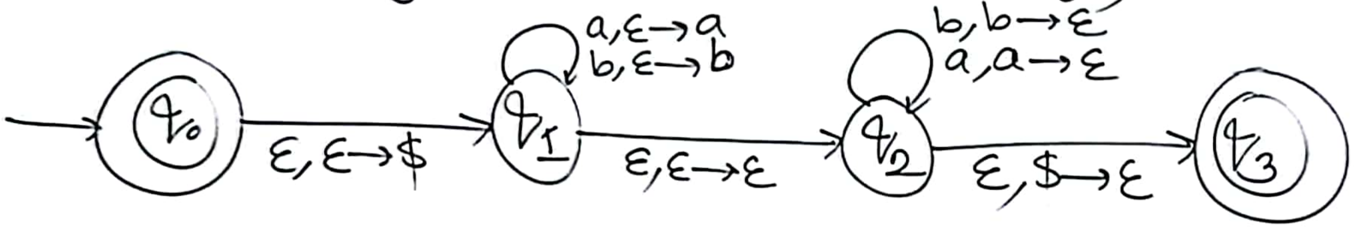
Draw PDA for, $L = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k \}$

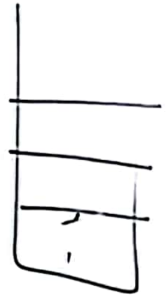
a=b
a=c



Push Down Automata (PDA)

* Draw PDA for, $L = \{ ww^R \mid w \in \{a, b\}^* \}$





* Design PDA for $L = \{ a^m b^n \mid m < n \}$