PDA

PDA

- Pushdown Automata
- PDA is the Class of Automata associated with CFL
- Finite Automata cannot recognize all context free languages as FA has strictly finite memory whereas the recognition of a CFL may require storing an unbounded amount of information.

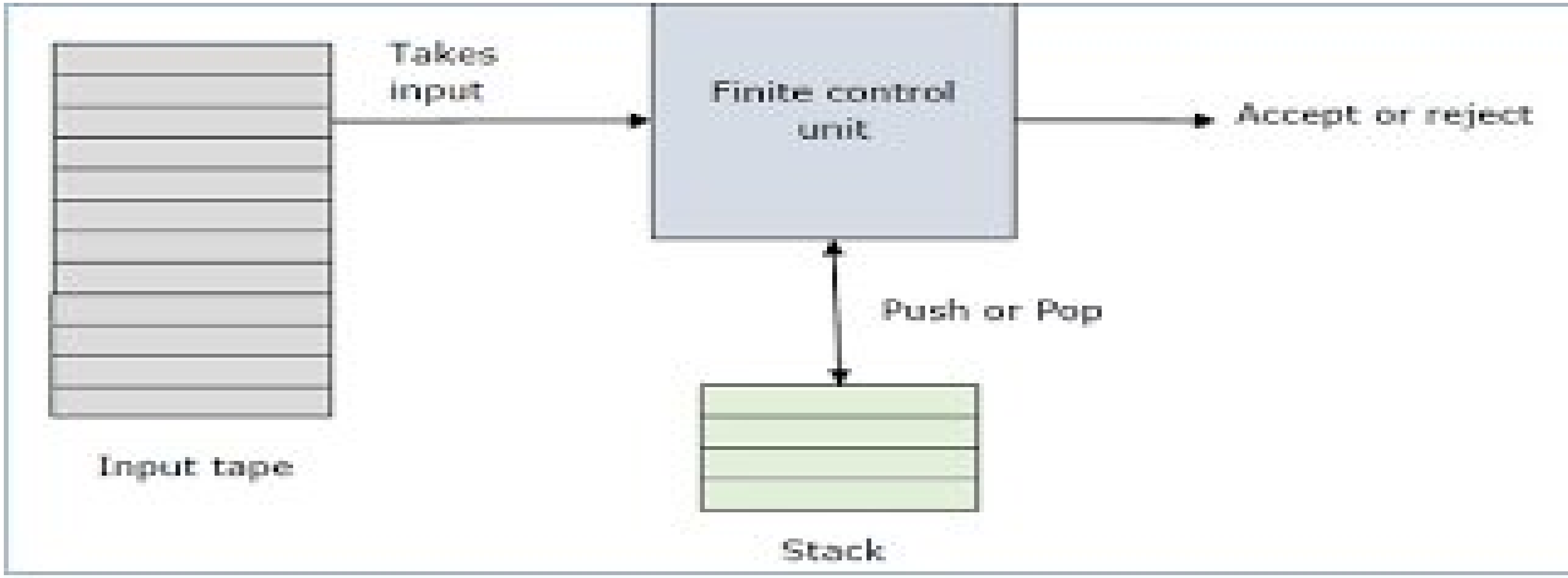
PDA

- Eg-
- L={$a^n b^n$|n>=0}
- When scanning the string, we must check that all a's are precede the first b, we also need to count the number of a's
- Since n is unbounded, this counting cannot be done with a finite memory.
- We want a machine that can count without limit

PDA

- A pushdown automaton is −

"Finite state machine" + "a stack"



- A pushdown automaton has three components −

1) an input tape,
2) a control unit, and
3) a stack with infinite size.

- The stack head scans the top symbol of the stack.

Deterministic PDA: Formal Definition

- A list of seven elements is called a 7-tuple,

- A PDA can be defined as a 7-tuple

PDA: Formal Definition

PDA is denoted by 7 Tuple: $M=(Q, \sum, \Gamma, \delta, q_0, z, F)$ where

- **Q:Finite set of Internal states of the Control Unit**
- **$\sum$:Finite input alphabet**
- **Γ: Finite Set of Stack Symbols/Pushdown symbols**
- **δ:Q X ($\sum$ U {ε}) X Γ ->Q X Γ^*: Transition function(TF)**

  or
- **$\delta: Q \times \sum_\varepsilon \times \Gamma_\varepsilon \rightarrow Q \times \Gamma_\varepsilon^*$**
- **q_0: Start state of Control Unit, $q_0 \in Q$**
- **z : Stack start Symbol, $z \in \Gamma$, Initially present in the stack**
- **F: Set of Final States/ Accept State, $F \subseteq Q$**

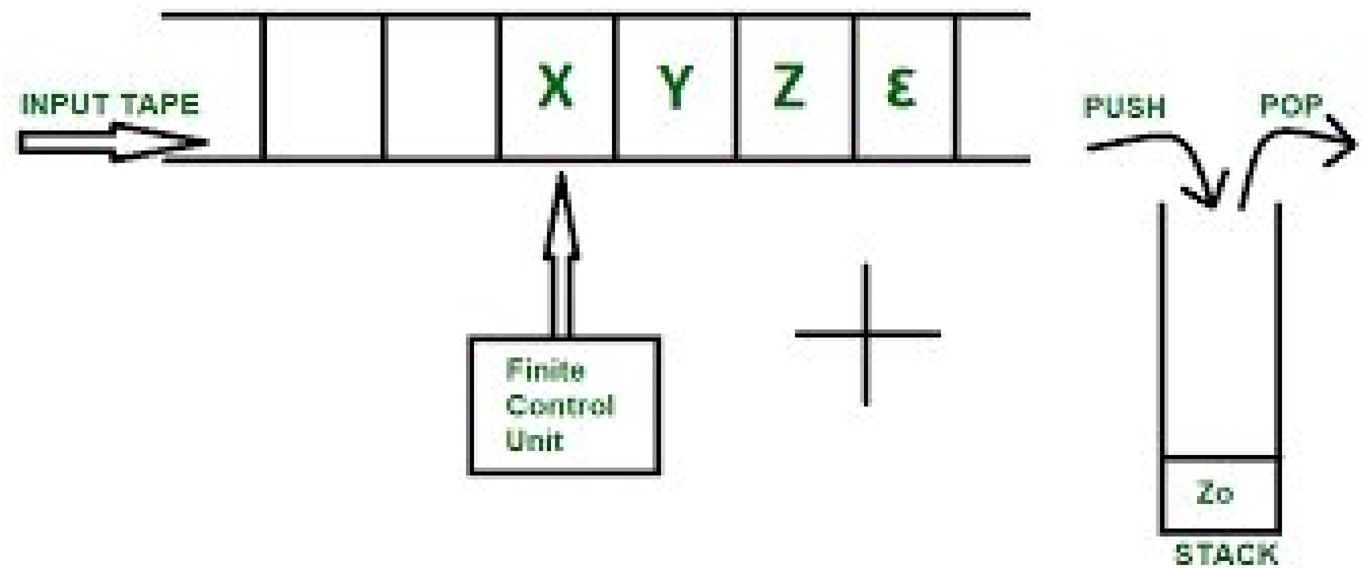
δ Function

δ Function:-

- **δ:Q X (∑ U {ε}) X Γ -> Q X Γ *: Transition function(TF)**

or

- **$\delta: Q \times \Sigma_{\varepsilon} \times \Gamma_{\varepsilon} \rightarrow Q \times \Gamma_{\varepsilon}{}^{*}$**



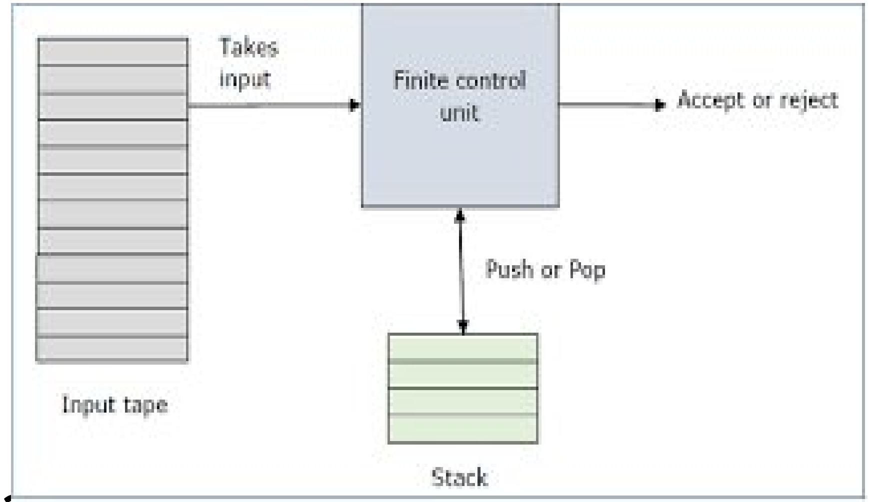
δ Function

δ Function:-

- **$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$: Transition function(TF)**

or

- **$\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow Q \times \Gamma_\varepsilon^*$**



Domain-

- **The Arguments are :-**

1) **Current State of the control Unit**
2) **Current Input Symbol**
3) **Current Symbol on the Top of the stack**

Range-

- **The Set of pair (q,x) where :-**

1) **q is Next state of the control Unit**
2) **x is a string which is put on the top of the stack , in place of the single symbol there before**

δ Function

δ Function:-

- **$\delta : Q \times (\Sigma \cup \{\varepsilon\} \times \Gamma \rightarrow Q \times \Gamma^*$: Transition function(TF)**

or

- **$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow Q \times \Gamma_\varepsilon^*$**

Domain-

- **The Arguments are :-**

1) **Current State of the control Unit**
2) **Current Input Symbol- Can be ε, indicating a move that does not consume an input symbol, Also called ε or Null Transition**
3) **Current Symbol on the Top of the stack- δ is defined so that it needs a stack symbol, no move is possible if the stack is empty**

Instantaneous Description

Let $M=(Q, \sum, \Gamma, \delta, q_0, z, F)$ be a pda

An Instantaneous Description(ID) is (q,x,α)

where $q \in Q$, $x \in \sum_\varepsilon$, $\alpha \in \Gamma_\varepsilon$

1) **q is the Current State of the control Unit**
2) **x is the Unread part of the input string/The part of the input to be processed**
   - **Say $a_1,a_2,....a_n$**
   - **The PDA will process in $a_1,a_2,....a_n$ order only**
3) **α are the stack contents with the left most symbol indicating the top pf the stack**

- **This triplet is called ID**

Instantaneous Description

- **A move from one ID to another will be denoted by Turnstile notation**

- $\vdash$ sign is called a “turnstile notation” and represents one move.

Move Relation-

Let M be a PDA

A move relation between IDs are defined as:-

$(q, a1a2\ldots\ldots\ldots an, z1z2\ldots.zn) \vdash (q', a2\ldots\ldots\ldots an, \beta z2\ldots.zn)$

if $\delta(q,a1,z1)=(q', \beta)$

PDA Example 1

Design a PDA for accepting the language $L=\{a^n b^n | n>=1\}$

Logic-

PDA Example 1

Design a PDA for accepting the language $L=\{a^n b^n | n>=1\}$

Rules-

PDA Example 1

Design a PDA for accepting the language $L=\{a^nb^n | n>=1\}$

Simulation-

PDA Example 2

Design a PDA for accepting the language $L=\{a^n b^{2n} | n>=1\}$

Logic-

PDA Example 2

Design a PDA for accepting the language $L=\{a^n b^{2n} | n>=1\}$

Rules-

PDA Example 2

Design a PDA for accepting the language $L=\{a^n b^{2n} | n>=1\}$

Simulation-

PDA Example 3

Design a PDA for accepting the language $L=\{w \mid w \in (a + b)^* \text{ and } n_a(w)=n_b(w)\}$

Logic-

PDA Example 3

Design a PDA for accepting the language $L=\{w \mid w \in (a + b)^* \text{ and } n_a(w)=n_b(w) \}$

Transition Rules-

- **String "aababb"**

PDA Example 3

Design a PDA for accepting the language $L=\{w \mid w \in (a + b)^* \text{ and } n_a(w)=n_b(w)\}$

Simulation-

PDA Example 4

Design a PDA for accepting the language L={w | w ∈ $(a + b)^*$ and $n_a(w) > n_b(w)$ }

Logic-

- **String “aababab”**

PDA Example 4

Design a PDA for accepting the language L={w | w ∈ $(a + b)^*$ and $n_a(w)>n_b(w)$ }

Transition Rules-

PDA Example 4

Design a PDA for accepting the language L={w | w ∈ (a + b)* and $n_a(w)>n_b(w)$ }

Simulation-

PDA Example 5

Design a PDA for accepting the language L={w | w ∈ (a + b)* and $n_a(w)<n_b(w)$ }

Logic-

- **String “abbab”**

PDA Example 5

Design a PDA for accepting the language L={w | w ∈ (a + b)* and $n_a(w) < n_b(w)$ }

Transition Rules-

PDA Example 5

Design a PDA for accepting the language L={w | w ∈ (a + b)* and $n_a(w)<n_b(w)$ }

Simulation-

PDA Example 6

Design a PDA that accepts a string of well formed paranthesis.

Consider the paranthesis as (,),{,},[,]

PDA Example 7

Design a PDA for language L={wcw^R|w is in (0|1)* and wR is reverse of w}

Logic-

String 101c101

PDA Example 7

Design a PDA for language L={wcw^R|w is in (0|1)* and wR is reverse of w}

Logic-

- The PDA will go on pushing the symbols onto the stack till it encounters c in the input
- It reads c but will not push it onto the stack
- For every symbol read after c, it checks whether it matches with the topmost symbol of the stack
- If input read is 0 ,top=0,pop
- If input read is 1,top=1,pop
- If input ends, we reach zo, replace z0 by null
- Stack empty
- String accepted

PDA Example 7

Design a PDA for language L={wcw^R|w is in (0|1)* and wR is reverse of w}

Transition Rules

Deterministic PDA

Non-Deterministic PDA

NPDA: Formal Definition

NPDA is denoted by 7 Tuple: M=(Q, ∑, Γ, δ, q_0, z, F) where

- **Q:Finite set of Internal states of the Control Unit**
- **∑:Finite input alphabet**
- **Γ: Finite Set of Stack Symbols/Pushdown symbols**
- **δ:Q X (∑ U {ε}) X Γ ->$2^{Q \times \Gamma^*}$: Transition function(TF)**

  or
- **δ:Q X Σ_ε X Γ_ε ->$2^{Q \times \Gamma^*}$**
- **q_0: Start state of Control Unit, q0 ∈ Q**
- **z : Stack start Symbol, z ∈ Γ, Initially present in the stack**
- **F: Set of Final States/ Accept State, F ⊆ Q**

Non Determinism-Example 1

Design a PDA for language L={ww^R|w is non-empty even palindromes over {a,b} , wR is reverse of w}

Initial a and b read , when top=z0, it will push

Further, If we read a, top=a, Two Moves possible

1) **Pop and advance input tape head one position**
2) **Push the element read and advance input tape head one position**

Further, If we read b, top=b, Two Moves possible

1) **Pop and advance input tape head one position**
2) **Push the element read and advance input tape head one position**

Non Determinism-Example 1

Even length palindromes

String “baab”

String “baaaab”

Non Determinism-Example 1

Design a PDA for language L={ww^R|w is non-empty even palindromes over {a,b} , wR is reverse of w}

δ(q0,a,zo)=(q0,az0)

δ(q0,b,zo)=(q0,bz0)

δ(q0,a,a)={(q0,aa),(q1, ε)}

δ(q0,b,b)={(q0,bb),(q1, ε)}

δ(q1,a,a)=(q1, ε)

δ(q1,b,b)=(q1, ε)

δ(q1,ε,z0)=(q1, ε)

Practice