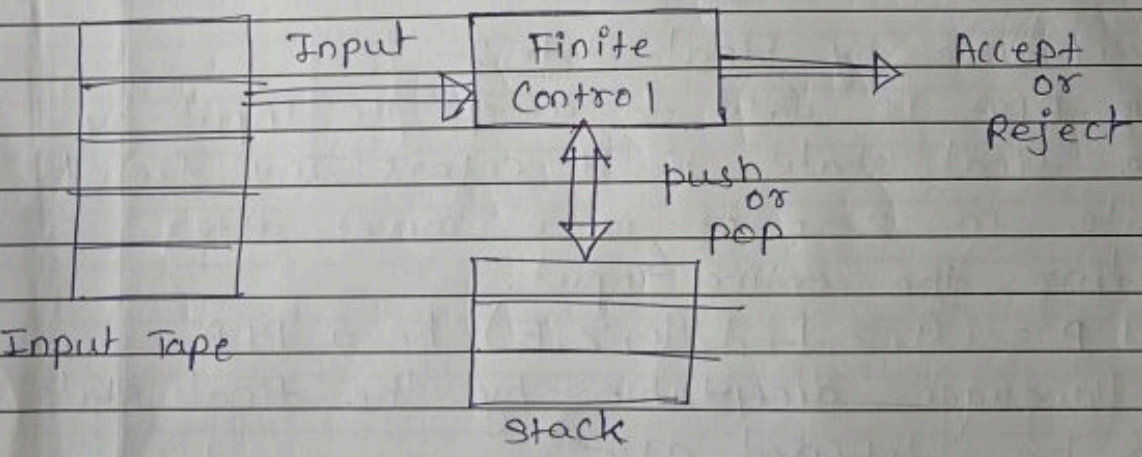
Unit - IV
Pushdown Automata.

Sajivani Adul.

PDA → Pushdown Automata is a way to implement a CFG in the same way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.

- A PDA is more powerful than FA. any language which can be acceptable by FA can also be acceptable by PDA (Pushdown Automata).
- PDA also accepts a class of language which even cannot be accepted by FA. Thus PDA is more superior to FA.



Component of PDA :-

- Input Tape
- Finite Control
- Stack

Definition :- PDA can be defined as a collection of 7 components.

$PDA = (Q, \Sigma, \Gamma, q_0, z, F, \delta)$

→ Q : Finite set of states
→ Σ : Input set
→ Γ : stack symbol which can be pushed & popped from the stack.
→ q_0 : Initial state.
→ Z : start symbol which is in Γ.
→ F : set of final states.
→ δ : mapping function / Transition function which is used for moving from current state to next state.

PDA Acceptance :
A language can be accepted by pushdown automata using two approaches:

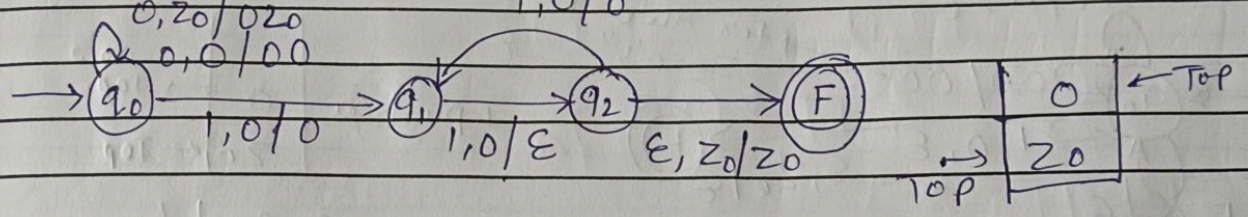
1) Acceptance by Final state :-
→ The PDA is said to accept its input by the final state if it enters any final state in zero or more moves after reading the entire input.
→ Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ be a PDA.
The language acceptable by the final state can be defined as :-
→ $L(PDA) = \{ W \mid (q_0, W, Z) \vdash^* (P, \varepsilon, \varepsilon), q \in F \}$.

2) Acceptance by Empty Stack :-
→ On reading the input string from the initial configuration for some PDA, the stack of PDA gets empty.
→ Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ be a PDA.
The language acceptable by empty stack can be defined as :-
→ $N(PDA) = \{ W \mid (q_0, W, Z) \vdash^* (P, \varepsilon, \varepsilon), q \in Q \}$

Example 1. →

Construct a PDA that accepts the language L over {0, 1} by empty stack which accepts all the string of 0's & 1's in which a number of 1's are twice of number of 0's. OR $L = \{0^n 1^{2n}\}$, $n > 0$.

⇒ I Assume the string – | 0 | 0 | 1 | 1 | 1 | 1 |



Transition Function:

1) $\delta(q_0, 0, Z_0) = (q_0, 0, Z_0)$

2) $\delta(q_0, 0, 0) = (q_0, 0, 0)$

3) $\delta(q_0, 1, 0) = (q_1, 0)$

4) $\delta(q_1, 1, 0) = (q_2, \varepsilon)$

5) $\delta(q_2, 1, 0) = (q_1, 0)$

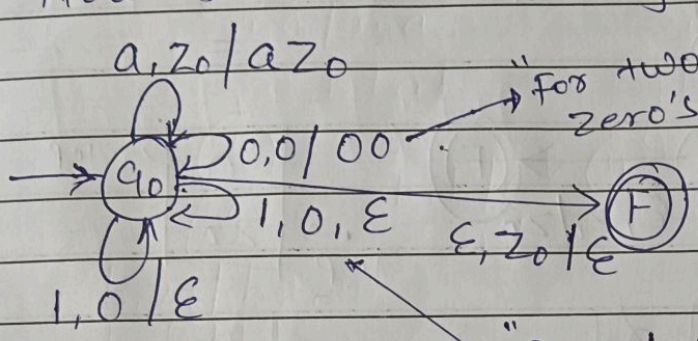
6) $\delta(q_2, \varepsilon, Z_0) = (F, Z_0)$.

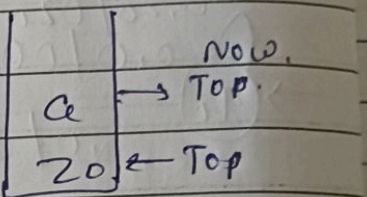
① Construct PDA for given string.
$L = \{w / na(w) = nb(w)\}$

⇒

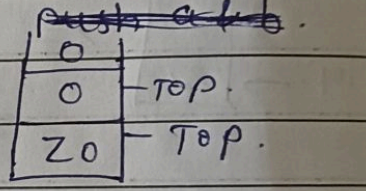
I assume string like abaabb.
Language for this is -
$L = \{ab, ba, abab, abaabb, ababab \ldots\}$.

Now I assume string abaabb....





2. Read b.
pop a.
~~push a & b~~.



Transition function -

1) $\delta(q_0, a, Z_0) = (q_0, a, Z_0)$

2) $\delta(q_0, 1, 0) = (q_0, \varepsilon)$

3) $\delta(q_0, 0, 0) = (q_0, 0, 0)$

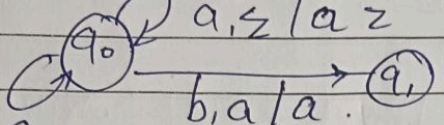
4) $\delta(q_0, 1, 0) = (q_0, \varepsilon)$

5) $\delta(q_0, \varepsilon, Z_0) = (F, \varepsilon)$.

Deterministic Pushdown Automata :-

→ A Pushdown Automata is said to be deterministic if all derivations in the design has to give only single move.

eg -



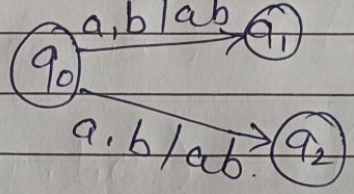
OR.

$L = (0^n 1^{2n})$.

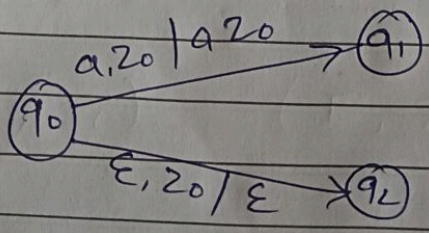
Non- Deterministic Pushdown Automata :-

→ The NDPDA is very much similar to NFA.
→ A PDA is non-deterministic with one transition we can take more than (one) 1 move. then we can say it is NDPDA.
→ we can do it in two ways

1) if two or more edges labeled with same i/p & stack symbols.



2) when a state have two edges with same stack symbol & one of i/p symbol is epsilon.



eg ① $L = WW^R$, $W \in (a, b)^*$

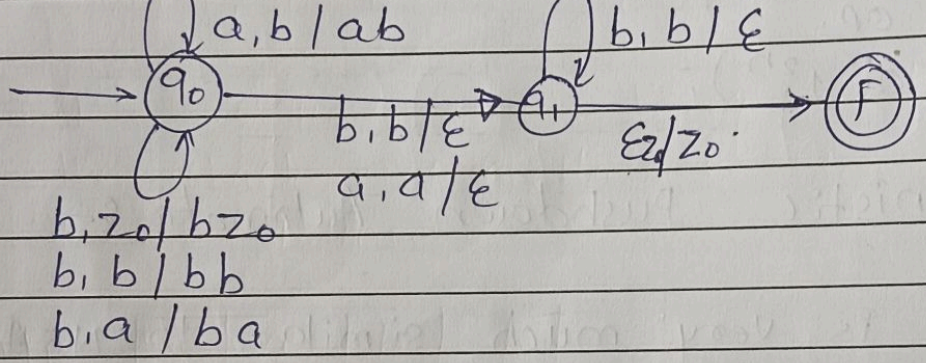
⇒

$L = \{abba, baab, abbbba, baaaab \ldots\}$.

(For every string do two diff. operation push & pop)

push - Assume that this is not middle symbol / alpha.

pop - Assume that this is the middle symbol / alpha.



→ CFG to PDA Conversion.

→ Follow following steps :-

→ Convert the given productions of CFG into GNF. (Context free Grammer to Greibach Normal form).

→ PDA will only have one state {q}.

→ Initial symbol of CFG will be the initial symbol in the PDA.

→ for non-terminal symbol, add the following rule:
$\delta(q, \varepsilon, A) = (q, \alpha)$.
where production rule is $A \rightarrow \alpha$.

→ for each terminal symbols, add following rule:
$\delta(q, a, a) = (q, \varepsilon)$ for every terminal symbol.

→ Tuples of PDA - $(Q, \Sigma, \delta, \Gamma, Z_0, q_0, F)$.

→ To convert CFG to PDA. there is two Techinque - 1) GNF
2) No-GNF. } · PDA empty stack.

→ Now, the GNF means, The production rule is start from terminals or starting terminal followed by any ~~Terminal~~ variables.

eg 1) → $S \rightarrow a$
$S \rightarrow aV^*$

The above eg. is comes under GNF.

eg 2) → $S \rightarrow aSb$ → Not in GNF.

→ If Grammer is not in GNF form the use-
Tuples for :- $(q, \Sigma, \delta, (V \cup T), S, q, \phi)$.

q - only one single state.
Σ - Input alphabet / set of terminals.
δ -
Γ - $(V \cup T)$ - push into stack - variable & Terminals.
S - Initial stack symbol.
q - single state.
ϕ - no final state.
(because we use PDA acceptance by empty stack).

eg (1) $S \to aSb$ } production is not in GNF.
$S \to ab$ }

⇒ Step 1 - Write different rule for above Production.

1) $\delta(q, \epsilon, S) = (q, asb)$
2) $\delta(q, \epsilon, S) = (q, ab)$
3) $\delta(q, a, a) = (q, \epsilon)$
4) $\delta(q, b, b) = (q, \epsilon)$

If I assume the string aabb then. the stack representation for this string is as follow :-

i/p - | a | a | b | b | .....

1) | S | - Top ... No need to read i/p symbol. just add Production rule.

2) | a | S | b | ... if i/p is a & Top of stack is 'S' then replace 'S' with production rule.

3) | S | b | ... if both the i/p are same. then pop the symbol from stack. i.e pop 'a'

4) | a | $ | b | b | ... Reapeat step 1. ..... apply 2nd production rule.

5) | $ | b | b | ... Repeat step 3.

6) | b | b | ... Repeat step 3 i.e. Pop 'b' if i/p is 'b' & Top is 'b' (Top →)

7) | b | .... Repeat step 6.

8) | | .... stack is empty

eg (2) · $S \to OBB$ · GNF ⇒ all tuples are same.
$B \to OS / 1S / O$ $(V \cup T)^* = V$ (only variable)

⇒
1) $\delta(q, 0, S) = (q, BB)$
2) $\delta(q, 0, B) = (q, S)$
3) $\delta(q, 1, B) = (q, S)$
4) $\delta(q, 0, B) = (q, \epsilon)$.

I assume string 01.0000

0	1	0	0	0	0
	↑				

1) | S | - Top ... Initial we the stack symbol is 'S' also read i/p & increment pointer.

2) | B | ... if i/p is 1 & top of stack is 'B'.
| B | then apply 3rd production rule.

3) | S | .... i/p '0' Top of stack S. then.
| B | apply 1st production rule.

4) | B | .... i/p '0' top B then pop 'B'
| B | means apply 4th production rule.
| B |

5) | B | .... Repeat step 4.
| B |

6) | B | .... Repeat step 4.

7) | | .... stack is empty.

eg ③ $S \longrightarrow AB$

$A \longrightarrow 0S \mid 0$

$B \longrightarrow 1S \mid 1$

⟹ Production Rules :-

1) $\delta(q, \epsilon, S) = (q, AB)$

2) $\delta(q, \epsilon, A) = (q, 0S)$

3) $\delta(q, \epsilon, A) = (q, 0)$

4) $\delta(q, \epsilon, B) = (q, 1S)$

5) $\delta(q, \epsilon, B) = (q, 1)$

6) $\delta(q, 0, 0) = (q, \epsilon)$

7) $\delta(q, 1, 1) = (q, \epsilon)$.

→ PDA to CFG :-

$L = (\{q_0, q_1\}, \{a, b\}, \{a, z_0\}, \delta, q_0, z_0, \phi)$

$\delta(q_0, a, z_0) = (q_0, az_0)$ $\quad \delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, a)$ $\quad \delta(q_1, b, a) = (q_1, a)$
$\delta(q_1, a, a) = (q_1, \epsilon)$ $\quad \delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

⇒ There are total 4 tuples in CFG -

$L = (V, T, P, S)$.

$V = (S, [q_0\ a\ q_0], [q_0\ a\ q_1], [q_1\ a\ q_0], [q_1\ a\ q_1], [q_0\ z_0\ q_0], [q_0\ z_0\ q_1], [q_1\ z_0\ q_0], [q_1\ z_0\ q_1])$

$T = \{a, b\}$

$S \Rightarrow [q_0\ z_0\ q_0]$
$S \rightarrow [q_0\ z_0\ q_1]$
} start symbol Production.

Now there are total 6 transitions are given now we have to write production rule for same -

1) $\delta(q_0, a, z_0) = (q_0, az_0)$ ..... $2^2 = 4$ = Prod^n

$[q_0, z_0, q_0] \rightarrow a\,[q_0, a, q_0]\,[q_0, z_0, q_0]$
$[q_0, z_0, q_0] \rightarrow a\,[q_0, a, q_1]\,[q_1, z_0, q_0]$
$[q_0, z_0, q_1] \rightarrow a\,[q_0, a, q_0]\,[q_0, z_0, q_1]$
$[q_0, z_0, q_1] \rightarrow a\,[q_0, a, q_1]\,[q_1, z_0, q_1]$

2) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$

$[q_0, a, q_0] \Rightarrow a\,[q_0, a, q_0]\,[q_0, a, q_0]$
$[q_0, a, q_0] \rightarrow a\,[q_0, a, q_1]\,[q_1, a, q_0]$
$[q_0, a, q_1] \rightarrow a\,[q_0, a, q_0]\,[q_0, a, q_1]$
$[q_0, a, q_1] \Rightarrow a\,[q_0, a, q_1]\,[q_1, a, q_1]$

3) $\delta(q_0, b, a) = (q_1, a)$ ..... $2^1 = 2$ = Prod^n.

$[q_0, a, q_0] \Rightarrow b\,[q_1, a, q_0]$
$[q_0, a, q_1] \Rightarrow b\,[q_1, a, q_1]$

3) $\delta(q_1, b, a) = (q_1, a)$ ... $2^1 = 2$

$[q_1, a, q_0] \Rightarrow b\,[q_1, a, q_0]$

$[q_1, a, q_1] \Rightarrow b\,[q_1, a, q_1]$

4) $\delta(q_1, a, a) = (q_1, \epsilon)$ ... $2^\epsilon = 0 = 1$ prod^n. (only one prod^n).

$[q_1, a, q_1] \Rightarrow a$.

5) $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$[q_1, z_0, q_1] \Rightarrow \epsilon$.

→ Pumping lemma for CFG =

- To prove certain languages are not a context free.

- let L be a CFL
- let n be a constant.
- Any string z in L, $|z| \geq n$.
- split $z = uvwxy$ such that,

i) $|vwx| \leq n$

ii) $vx \neq \varepsilon$ or $|vx| \geq 1$

iii) for all $i \geq 0$, $uv^i wx^i y \in L$.