UNIT-IV

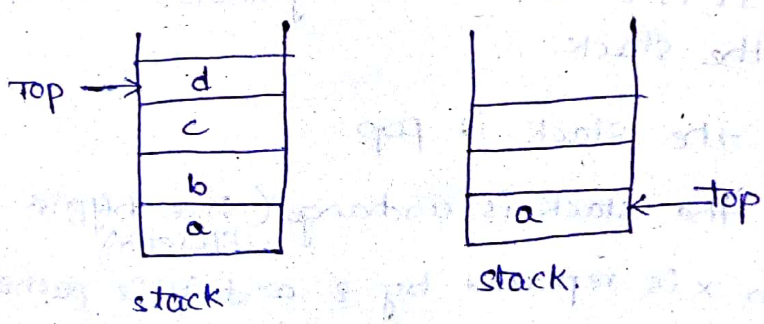
PUSH DOWN AUTOMATA

* Introduction * Basic model [formal definition] * Graphical notation
* Instantaneous Description (ID) * Acceptance of PDA.

Introduction:-

* A PDA is a way to implement a CFG in a similar way we can design FA for Regular Grammar.
* PDA is more powerful than finite state machine.
* FSM has a very limited memory. But a PDA has more memory.
* $\boxed{PDA = FSM + stack}$
* A stack is a way we arrange elements one on the top of stack.
* A stack does two basic operations.
  - i) PUSH :- A new element is added at the top of the stack.
  - ii) pop :- The top element of the stack is read and remove.

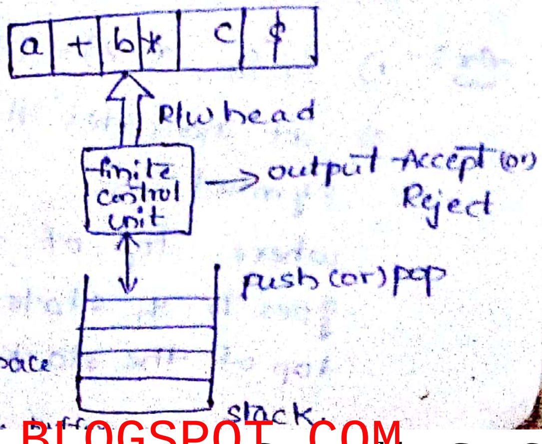
Ex:- push(a) push(b) push(c) push(d)   pop() pop() pop()



* Basic Model of PDA :-

PDA has three Components.

i) input tape
ii) finite control unit
iii) stack



* A stack with infinite size.
* It has unlimited amount of storage space
* Used to store data and remove the data. temporarily which is read by finite ...

Formal definition:-

- Mathematically a PDA is defined with 7 Tuples like

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

$Q \rightarrow$ finite and non-Empty set of states

$\Sigma \rightarrow$ finite and non-Empty set of input symbols

$\Gamma \rightarrow$ finite and non-empty set of stack symbols.

$\delta \rightarrow$ It is a transition function which is defined as

δ :

$$\boxed{Q \times \{\Sigma \cup \epsilon\} \times \Gamma^* \longrightarrow Q \times \Gamma^*}$$

$$\boxed{Q \times \Sigma^* \times \Gamma^* \longrightarrow Q \times \Gamma^*}$$

where 'δ' takes three tuples as i/p like $\delta(q, a, x)$

where i) q is a state in Q.

ii) a is either an i/p symbol in Σ (or) a is also belongs ϵ.

iii) x is a stack symbol i.e; member of Γ

iv) The o/p of δ is finite set of pairs like (p, γ)

where, p: It is a new state.

γ: It is a set of stack symbols. that replace 'x' at the top of the stack.

Ex:- 1) If $\gamma = \epsilon$ then the stack is pop.

2) If $\gamma = x$ then the stack is unchange (since bypass operation)

3) If $\gamma = yz$ then x is replaced by z and 'y' is pushed on to the stack.

Ex:- 1) $\delta(q_0, a, z) = (q_1, yz)$

$\Rightarrow$ It indicates that from state q_0, reading i/p symbol 'a'.

where top of the stack z then the finite control goes to q_1 state and adding the element y to the top of the stack.

2) $\delta(q_1, a, z) = (q_2, \epsilon)$

⇒ It indicates that 'z' is removed from the stack and state is changed from q_1 to q_2.

3) $\delta(q_1, a, z) = (q_2, z)$

⇒ It indicates that on reading symbol 'a' state is changing from q_1 to q_2 and there is no change in the stack (bypass operation)

q_0 → It is the initial state.

$\boxed{q_0 \in Q}$

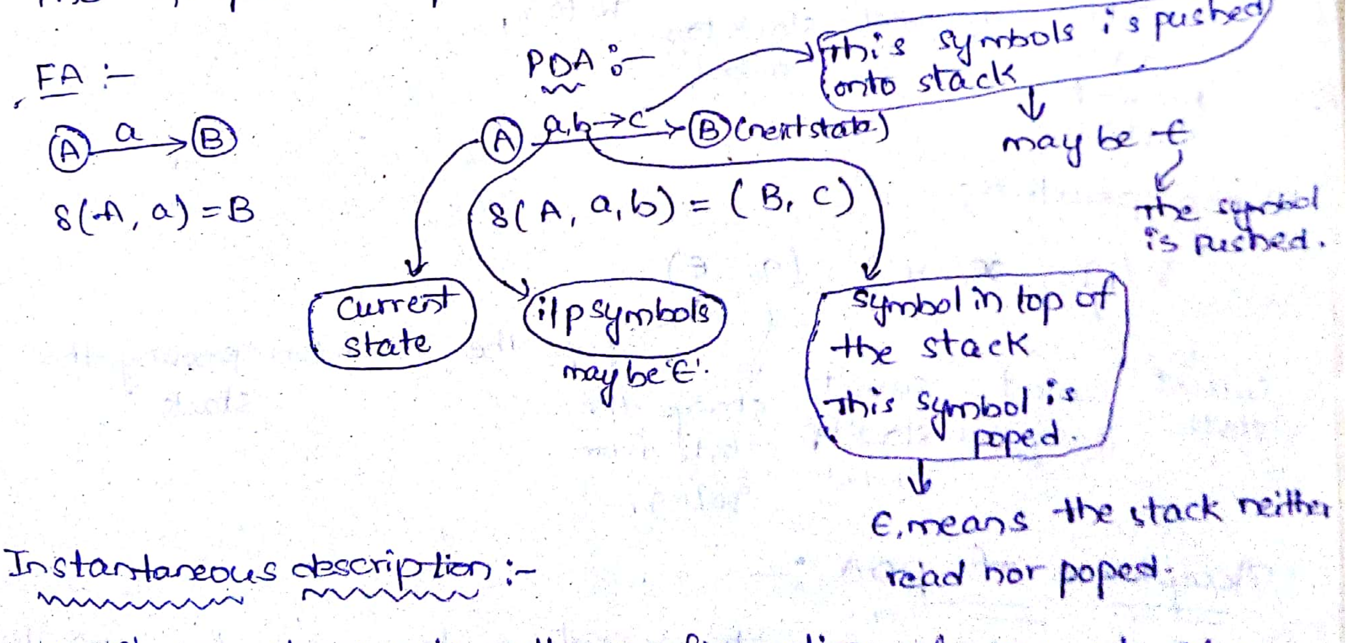
z_0 → It is the start stack symbol.

$\boxed{z_0 \in \Gamma}$

F → It is the set of final (or) accepting state and $(F \subset Q)$.

Graphical representation :-

The Graphical representation of PDA is Transition diagram



Instantaneous description :-

It is used to describe the configuration of PDA at given Instance.

ID remembers the state and stack content.

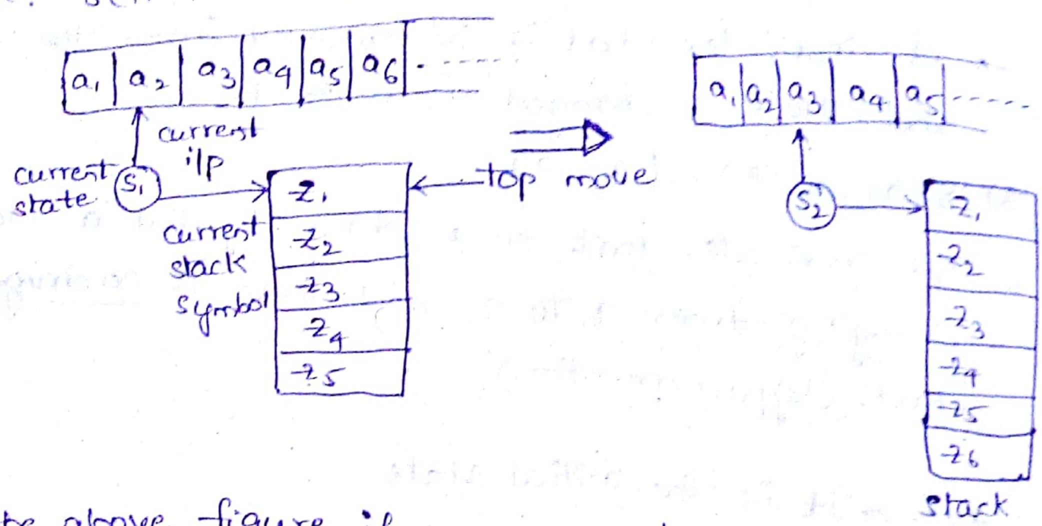
It was defined by Triple (q, w, γ) where

q → is a state.

w → input symbols of string

γ → is a string of stack symbols.

Example :- $\delta(P, a, z_0) = (q, \beta)$



From the above figure, if we are reading the current i/p symbol 'a_2' at current state 's_1' and current stack symbol 'z_1' then after a move we will reach to state s_2. and there will be some new symbol on the top of the stack. This description can be represented as.

1) PUSH operation :—

$\delta(q_0, a, z_0) = (q_1, az_0)$ → push 'a' onto the stack.

- q_0 → current state (or) present state
- a → current i/p symbol
- z_0 → current stack top symbol.
- q_1 → change the state from q_0 to q_1

2) pop operation :—

$\delta(q_0, x, y) = (q_1, \epsilon)$ → pop the stack (or) removing the stack.

- q_0 → current state
- x → current i/p symbol
- y → current stack top symbol.
- q_1 → change the state from q_0 to q_1

Acceptance of PDA :—

There are two ways to accept a language by PDA. They are i) Accepted by empty stack.

ii) Accepted by final state.

Accepted by empty stack :—

The given language Accepted by empty stack to be defined as $L(M) = \{ w \mid \delta(q_0, w, z_0) \xRightarrow{*} (P, \epsilon, \epsilon) \text{ for some } p \in Q \}$

that is, if stack becomes empty After scanning entire string then it is accepted by PDA, otherwise, not accepted.

Accepted by final string:-

The given language accepted by final state to be defined as $L(M) = \{ w \mid \delta(q_0, w, z_0) \overset{*}{\Rightarrow} (P, \epsilon, \gamma) \text{ for Some } P \in F \text{ and } \gamma \in T^* \}$
that is, eventhough stack is not empty, after scanning i/p string. if the finite control reaches to the final state then it is accepted. otherwise, not accepted.

Design of PDA:-

Types of PDA:-

i) Deterministic PDA :- if all derivations in the design has to give only single move.

ii) Non Deterministic PDA :- if derivation generates more than one move in the designing of a particular task.

1) Design a PDA that accepts equal no. of A's and B's.

sol:- δ:

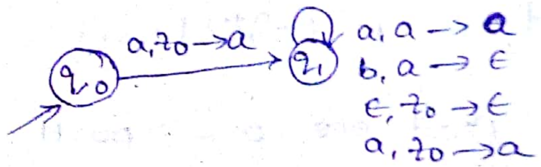
$\delta(q_0, a, z_0) = (q_1, a z_0)$

$\delta(q_1, a, a) = (q_1, aa)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

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

$\delta(q_1, a, z_0) = (q_1, a z_0)$



∴ The PDA machine for the above language is defined as

$M = (Q, \Sigma, T, \delta, q_0, z_0, F)$ where $Q = \{q_0, q_1\}$

$\Sigma = \{a, b, \epsilon\}$

$T = \{z_0\}$

δ:

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{\ \}$

(1) Read A's -> push operation, Read B's -> push operation
Consider a String $w = \{abab\}$ Read a's

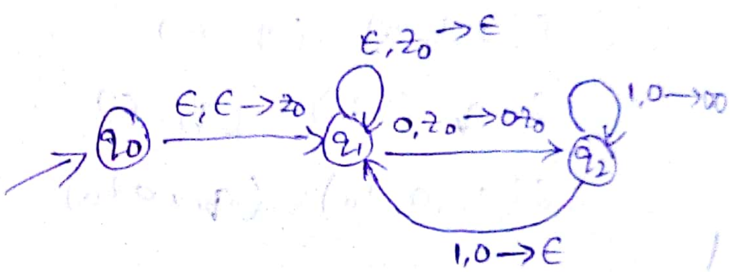
$\delta(q_0, abab, z_0) = \delta(q_1, bab, a z_0)$

③ Design a PDA for the language $L=\{0^n \text{ and } 1^{2n} \mid n\geq 1\}$

sol:- $L=\{0^n 1^{2n} \mid n\geq 1\}$

Read one 0 ⟶ push

Read two 1's ⟶ pop



$\delta(q_0, \epsilon, \epsilon) = (q_1, z_0)$

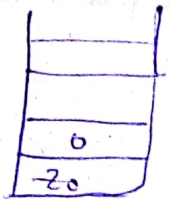
$\delta(q_1, 0, z_0) = (q_2, 0z_0)$

$\delta(q_2, 0, 0) = (q_2, 00)$

$\delta(q_2, 1, 0) = (q_2, 00)$

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

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



$\delta(q_1, 1, 0) = (q_1, \epsilon)$

$\delta(q_1, 1, 0) = (q_1, 0z_0)$

④ consider the string $w=\{001111\}$

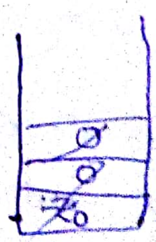
$\delta(q_1, 001111, z_0) = \delta(q_2, 01111, 0z_0)$

$= \delta(q_2, 1111, 00)$

$= \delta(q_2, 111, 00)$

$= \delta(q_1, 11, 0z_0)$

$= \delta(q_1, 1, 0z_0)$



stack

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

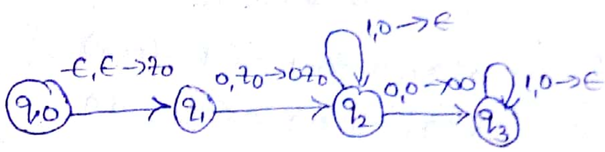
$= \delta(q_1, \epsilon, \epsilon)$

//.

Design a PDA for the language $L = \{0^n 1^n \mid n \geq 1\}$

Read 0's → PUSH

Read 1's → pop



$\delta(q_0, \epsilon, \epsilon) = (q_1, z_0)$

$\delta(q_1, 0, z_0) = (q_2, 0z_0)$

$\delta(q_2, 0, 0) = (q_2, 00)$

$\delta(q_2, 1, 0) = (q_3, \epsilon)$

$\delta(q_3, 1, 0) = (q_3, \epsilon)$

$\delta(q_3, \epsilon, z_0) = (q_3, \epsilon, \epsilon)$

* Design a PDA for the language $L = \{ww^R \mid w \in (a+b)^*\}$

ii) $L = \{wcw^R \mid w \in (a+b)^*\}$

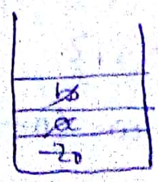
sol) i) $L = \{ww^R \mid w \in (a+b)^*\}$

In this language contains palindrome string. i.e;

if $w = ab$, $w^R = ba$ then $ww^R = abba$ is a palindrome.

- * We can read no. of a's and b's and pushed them into stack until we can reach the mid position of i/p string.
- * In the mid position we can't read any i/p and can't push onto stack.
- * After mid position when we read a (or) b then pop them from the stack. This process is repeated until stack is empty.

$\delta(q_0, \epsilon, \epsilon) = (q_1, z_0)$



$\delta(q_1, a, z_0) = (q_1, az_0)$

$\delta(q_1, b, z_0) = (q_1, bz_0)$

$\delta(q_1, \epsilon, \epsilon) = (q_2, z_0)$

$\delta(q_2, a, a) = (q_3, \epsilon)$

$\delta(q_2, b, b) = (q_3, \epsilon)$

$\delta(q_3, \epsilon, z_0) = (q_4, \epsilon, \epsilon)$

ii) $L = \{ \cancel{wcw} \; wcw^R \mid w = (a+b)^* \}$

$w = ab$

$w^R = ba$

$wcw^R = abcba$

$\delta(q_0, \epsilon, \epsilon) = (q_1, z_0)$

$\delta(q_1, a, z) = (q_1, az_0)$

$\delta(q_1, b, z_0) = (q_1, bz_0)$

$\delta(q_1, a, a) = (q_1, aa)$

$\delta(q_1, a, b) = (q_1, ab)$

$\delta(q_1, b, a) = (q_1, ba)$

$\delta(q_1, b, b) = (q_1, bb)$

$\delta(q_1, c, z_0) = (q_2, z_0)$

$\delta(q_1, c, a) = (q_2, a)$

$\delta(q_1, c, b) = (q_2, b)$

$\delta(q_2, a, a) = (q_3, \epsilon)$

$\delta(q_3, b, b) = (q_3, \epsilon)$

$\delta(q_3, \epsilon, z_0) = (q_4, \epsilon, \epsilon)$

Deterministic pushDown Automata:-

A DPDA is 7 Tuple like $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where Q is finite and non empty set of states

Σ is finite and nonempty set of i/p Alphabet

Γ is finite set of stack symbols

δ is a mapping function used for mapping (or) moving from current state to next state. is defined as $\delta(q_0, x, z_0) = (q, x\beta)$ where

q_0 is current state

x is current i/p symbol

z_0 is current stack symbol

q is next state

$x\beta$ shows top of the stack.

if δ denotes a unique transition for each i/p then PDA is said to be deterministic PDA

Ex:- 1) $L = \{ a^n b^n \mid n \geq 1 \}$

2) $L = \{ wcw^R \mid w = (a+b)^* \}$

Nondeterministic PDA:-

It is a 7 tuple like $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

Q is finite and non Empty set of states

Σ is finite and non Empty set of i/p Alphabet

Γ is finite set of stack symbols.

δ is a mapping function used for moving from current state to next state and is defined as $\delta(q_0, x, z_0) = (q, x\beta)$

q_0 is current state

x is current i/p symbol

z_0 is stack symbol

q is next state

$x\beta$ is top of the stack.

if 'δ' denotes more than one transition for a particular i/p symbol then the PDA is said to be non-deterministic PDA.

Ex:- $L = \{ ww^R \mid w = (a+b)^* \}$

Context free grammar and push Down automata:-

Conversion of CFG to PDA.

conversion of PDA to CFG.

i) Conversion of CFG to PDA:-

* for constructing a PDA from given CFG it is necessary to convert this CFG to some Normal form like GNF.

* For converting given CFG to PDA, By this method the necessary condition is that the first symbol on RHS of production rule must be a terminal symbol. This rule that can be used to obtain PDA from CFG.

Algorithm:-

Rule 1:- for non-terminal symbols, add following rule

$\delta(q, \epsilon, A) = (q, \alpha)$ where the production rule is $A \to \alpha$.

Rule 2:- for each terminal symbols, add following rule

$\delta(q, a, a) = (q, \epsilon)$ for every terminal symbol 'a' in given CFG.

Ex:- construct a PDA for the given CFG $S \to 0BB$

$B \to 0S$

$B \to 1S$

$B \to 0$

Sol:- The given CFG $G = (V, T, P, S)$ where V = non-terminals $\{S, B\}$

$T = \{0, 1\}$

$P:$ $S \to 0BB$

$B \to 0S$

$B \to 1S$

$B \to 0$

$S = \{S\}$

Rule 1:- $A \to \alpha$

$\delta(q, \epsilon, A) = (q, \alpha)$

$S \to 0BB$

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

$B \to 0S$

$\delta(q, \epsilon, B) = (q, 0S)$

$B \to 1S$

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

$B \to 0$

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

Rule:-2

Terminals

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

$T = \{0, 1\}$

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

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

$\therefore$ The corresponding PDA for the given CFG is defined as

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

$Q = \{q\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{S, B, 0, 1\}$

δ = it is a transition symbol defined as

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

$\delta(q, \epsilon, B) = (q, 0S)$

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

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

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

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

$q_0 = \{q\}$

$z_0 = \{z_0\}$

$F = \{\ \}$

2) construct a PDA for the following CFG S->OS1
S->A
A->1AO|S|ε.

sol: The Given CFG is S->OS1
S->A
A->1AO
A->S
A->ε.

elimination of ε-production:-

A->ε ×
S->A
S->ε ×
S->OS1
S->Oε1
S->O1

A->1AO
A->1εO
A->1O
A->S
A->ε ×

∴ S->OS1|O1
S->A
A->1AO|1O
A->S.

elimination of unit productions:-

S->A ×
S->1AO|1O

A->S ×
A->OS1|O1

∴ The resultant CFG is.

S->1AO
S->1O
A->OS1
A->O1

∴ The Simplified CFG is.

S->1AO
S->1O
A->OS1|1AO|~~0~~1O
A->O1
S->OS1|O1

Method-2
P->1
Q->O

S->1AO
S->1AQ✓

S->1O
S->1Q✓

A->OS1
A->OSP✓

A->1O
A->1Q✓

S->OS1
S->OSP✓

A->1AO
A->1AQ✓

A->O1
A->OP✓

S->O1
S->OP.✓

∴ The simplified CFG in GNF is S->1AQ A->OSP
S->1Q A->1Q
S->OSP A->1AQ
S->OP A->OP
P->1
Q->O

Rule-1
A->α.

∴The PDA is

S->1AQ.
δ(q, ε, S) = (q, 1AQ)

S->OSP
δ(q, ε, S) = (q, OSP)

A->OSP
δ(q,ε,A) = (q,OSP)

S->1Q
δ(q, ε, S) = (q, 1Q)

S->OP
δ(q, ε, S) = (q, OP)

A->1Q
δ(q,ε,A) = (q,1Q)

$A \to 1AQ$

$\delta(q, \epsilon, A) = (q, 1AQ)$

$A \to 0P$

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

$P \to 1$

$\delta(q, \epsilon, P) = (q, 1)$

$Q \to 0$

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

Rule 2:— Terminals.

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

$T = \{0, 1\}$

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

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

Method :- 2

The Given CFG is $S \to 0S1$

$S \to A$

$A \to 1A0$

$A \to S$

$A \to \epsilon$.

The resultant PDA is $S \to 0S1$

$\delta(q, \epsilon, S) = (q, 0S1)$

$S \to A$

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

$A \to 1A0$

$\delta(q, \epsilon, A) = (q, 1A0)$

$A \to S$

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

$A \to \epsilon$

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

Construct PDA for the following CFG $S \to aABB | aAA$

$A \to aBB | a$

$B \to bBB | A$

sol:- The Given CFG is $S \to aABB$

$S \to aAA$

$A \to aBB$

$A \to a$

$B \to bBB$

$B \to A$

elimination of unit production :—

$B \to A$ ×

$B \to aBB$

$B \to a$.

∴ After eliminating unit production $B \to A$, The resultant CFG in GNF is, $S \to aABB$

$S \to aAA$

$A \to aBB$

$A \to a$

$B \to bBB$

$B \to aBB$

$B \to a$

∴ The PDA is:

$S \to aABB$

$\delta(q, \epsilon, S) = (q, aABB)$

$S \to aAA$

$\delta(q, \epsilon, S) = (q, aAA)$

$A \to aBB$

$\delta(q, \epsilon, A) = (q, aBB)$

$A \to a$

$\delta(q, \epsilon, A) = (q, a)$

$B \to bBB$

$\delta(q, \epsilon, B) = (q, bBB)$

$B \to aBB$

$\delta(q, \epsilon, B) = (q, aBB)$

$B \to a$

$\delta(q, \epsilon, B) = (q, a)$

Conversion of PDA to CFG :-

* If $M = (Q, \Sigma, T, \delta, q_0, z_0, F)$ is a PDA, then their exists CFG "G" which is accepted by PDA (M).

Let 'G' be a CFG which is generated by PDA. The "G" can be defined as $G = (V, T, P, S)$ where 'S' is the start symbol and the set of non-terminals $V = \{S, q, q', z_0\}$ where $S, q, q' \in Q$ and $z_0 \in T$.

Now, we get set of production rules using the following algorithm.

Algorithm :-

Rule 1 :- The start symbol production rule can be $S \to [q, z_0, q']$

where q indicates present state

q' indicates next state.

z_0 is the stack symbol.

Rule 2 :- If there exists a move of PDA as then the production rule can be return as $\delta(q, a, z_0) = (q', \epsilon)$

$[q, z_0, q'] \to a$

3) If there exists a move of PDA as $\delta(q, a, Z_0) = (q', Z_1 Z_2 Z_3 \ldots Z_n)$ then the production rules can be written as

$$[q, Z_0, q'] \rightarrow a[q, Z_1, q_1][q_1, Z_2, q_2][q_2, Z_3, q_3] \cdots [q_{n-1}, Z_n, q']$$

Ex:- construct a CFG from the following PDA $M = (\{q_0, q_1\}, \{0, 1\}, \{S, A\}, \delta, q_0, S, \phi)$ and

δ:

$\delta(q_0, 1, S) = (q_0, AS)$

$\delta(q_0, \epsilon, S) = (q_0, \epsilon)$

$\delta(q_0, 1, A) = (q_0, AA)$

$\delta(q_0, 0, A) = (q_1, A)$

$\delta(q_1, 1, A) = (q_1, \epsilon)$

$\delta(q_1, 0, S) = (q_0, S)$

Sol:- Let we will construct a CFG $G = (V, T, P, S)$ where

$T = \{0, 1\}$

$V = \{S \cup [q_0, S, q_0], [q_0, S, q_1], [q_1, S, q_0], [q_1, S, q_1], [q_0, A, q_0], [q_0, A, q_1], [q_1, A, q_0], [q_1, A, q_1]\}$

Now, Let us build the production rules as.

using rule ① the production rules for start symbol is

P_1: $S \rightarrow [q_0, S, q_0]$

P_2: $S \rightarrow [q_0, S, q_1]$

using Rule ③ of the Algorithm for the $\delta(q_0, 1, S) = (q_0, AS)$.

$q_0 < {q_0 \atop q_1}$ P_3: $[q_0, S, q_0] \rightarrow 1[q_0, A, q_0][q_0, S, q_0]$

$q_0 < {q_0 \atop q_1}$ P_4: $[q_0, S, q_0] \rightarrow 1[q_0, A, q_1][q_1, S, q_0]$

P_5: $[q_0, S, q_1] \rightarrow 1[q_0, A, q_0][q_0, S, q_1]$

P_6: $[q_0, S, q_1] \rightarrow 1[q_0, A, q_1][q_1, S, q_1]$

Now, for $\delta(q_0, \epsilon, S) = (q_0, \epsilon)$ using Rule ② of Algorithm we get.

P_7: $[q_0, S, q_0] \rightarrow \epsilon$

$\cancel{P_8: [q_0, S, A] \rightarrow q_0, A}$

Now for $\delta(q_0, 1, A) = (q_0, AA)$ using Rule ③ of Algorithm.

P_8: $[q_0, A, q_0] \rightarrow 1[q_0, A, q_0][q_0, A, q_0]$

~~Now for $\delta(q_0, 0, A) = (q_1, A)$ using Rule ⑤ of Algorithm~~

$P_9: [q_0, A, q_0] \rightarrow 1 [q_0, A, q_1] [q_1, A, q_0]$

$P_{10}: [q_0, A, q_1] \rightarrow 1 [q_0, A, q_0] [q_0, A, q_1]$

$P_{11}: [q_0, A, q_1] \rightarrow 1 [q_0, A, q_1] [q_1, A, q_1]$

Now, for $\delta(q_0, 0, A) = (q_1, A)$ using

$q_0 < q_0, q_1$ $\qquad$ $q_1 < q_0, q_1$

$P_{12}: [q_0, A, q_0] \longrightarrow 0 [q_1, A, q_0]$

$P_{13}: [q_0, A, q_1] \longrightarrow 0 [q_1, A, q_1]$

Now, for $\delta(q_1, 1, A) = (q_1, \epsilon)$

$P_{14}: [q_1, A, q_1] \rightarrow 1$

Now, for $\delta(q_1, 0, S) = (q_0, S)$

$q_1 < q_0, q_1$ $\qquad$ $q_0 < q_0, q_1$

$P_{15}: [q_1, S, q_0] \rightarrow 0 [q_0, S, q_0]$

$P_{16}: [q_1, S, q_1] \rightarrow 0 [q_0, S, q_1]$

PDA with two stacks:—

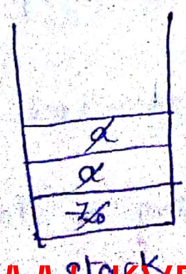
ⓐ PDA with one stack:

① $L = \{a^n b^n \mid n \geq 1\}$

consider the string $w = aabb$

read a's $\rightarrow$ push

read b's $\rightarrow$ pop



a a b b \$

when stack is empty then the string aabb is accepted.

stack

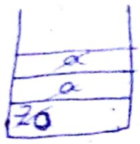
① $L = \{a^n b^n c^n \mid n \geq 1\}$

consider string $w = aabbcc$

read a's → PUSH

read b's → pop

read c's → nochange



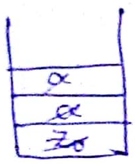
stack.

a a b b c c $

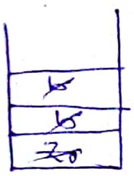
when read c there is no change in stack without completion of reading string 'w' the stack is empty. so, string is not accepted.

ⓑ PDA with two stacks:—

$L = \{a^n b^n c^n \mid n \geq 1\}$



stack 1



stack 2

w = aabbcc$

read a's → PUSH (on stack₁)

read b's → PUSH (on stack₂)

read c's → pop (a from stack₁ and 'b' from stack₂)

when two stacks are empty then string 'w' is accepted.

∴ The PDA with two stacks is more powerful than a ~~stack~~ PDA with one stack.

FA + 0-stack = NFA or DFA

FA + 1-stack = PDA.

FA + 2-stack = PDA with two stacks.

Applications of PDA:—

* used for deriving a string from the grammar.
* used for designing Top-down parser and bottom-up parser in compiler design.
* It works on regular grammar and Context-free grammars.
* It accepts regular language and CFL.
* It has remembering capability by maintaining a stack.
* It is more powerful than FA.