

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

PDA = FSM + stack

- * A stack is a way we arrange elements one on the top of stack.

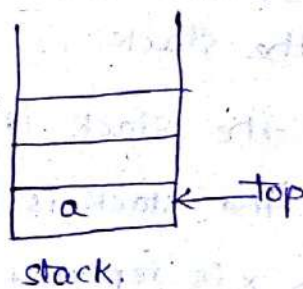
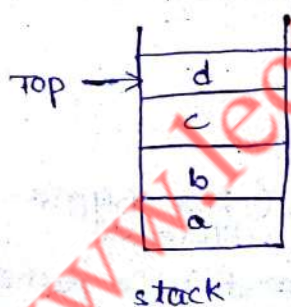
- * A stack does two basic operations.

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)

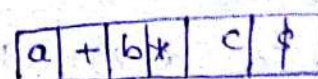
pop()
pop()
pop()



* Basic model of PDA:-

PDA has three Components.

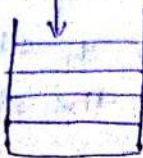
- input tape
- finite control unit.
- stack



↑ RW head



→ output Accept (or) Reject



push (or) pop

stack

- * A stack with infinite size.
- * It has unlimited amount of storage space.
- * Used to store data and remove the data.

Formal definition:-

Mathematically a pda is defined with 7-tuple like

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

$Q \rightarrow$ 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 :

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

$$Q \times \Sigma^* \times \Gamma^* \rightarrow Q \times \Gamma^*$$

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

where i) q is a state in Q .

ii) a is either an input 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, r) where, p : It is a new state.

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

Ex:- i) If $r = \epsilon$ then the stack is pop.

2) If $r = x$ then the stack is unchanged (since bypass operation)

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

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

$\Rightarrow$ It indicates that from state q_0 , reading input 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)

$q_0 \rightarrow$ It is the initial state.

$q_0 \in Q$

$z_0 \rightarrow$ It is the start stack symbol.

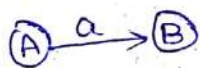
$z_0 \in \Gamma$

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

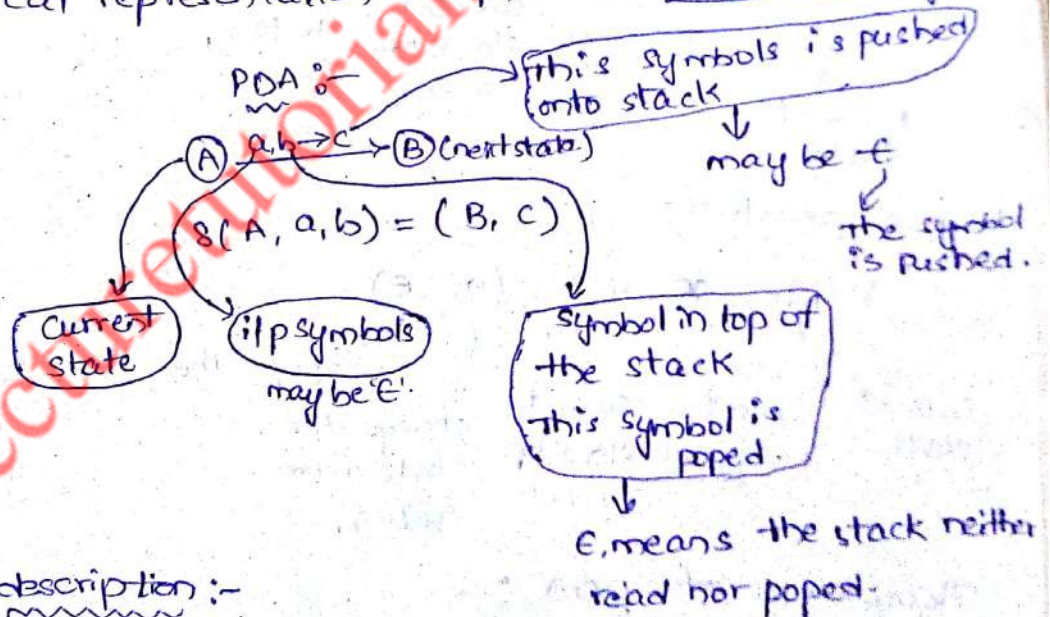
Graphical representation :-

The Graphical representation of PDA is Transition diagram

FA :-



$\delta(A, a) = B$



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

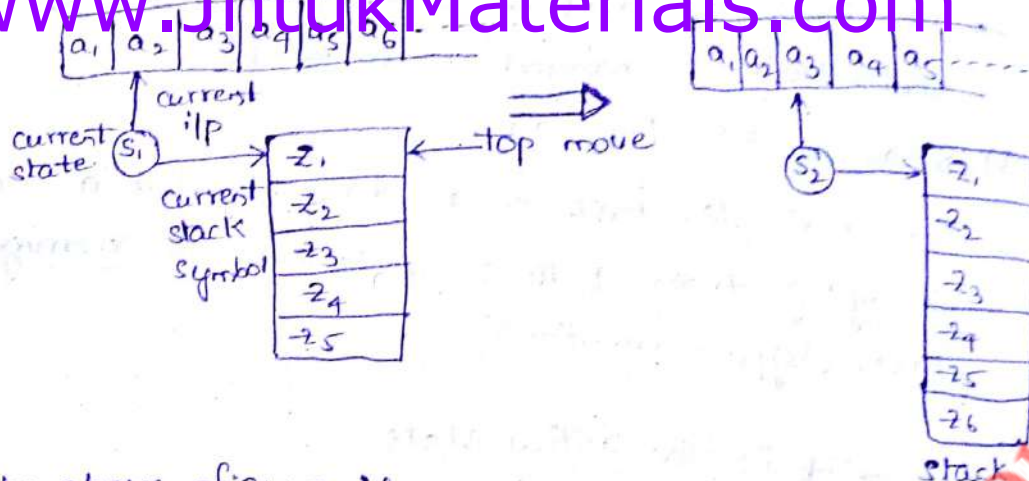
$q \rightarrow$ is a state.

$w \rightarrow$ input symbols of string

$\gamma \rightarrow$ is a string of stack symbols.

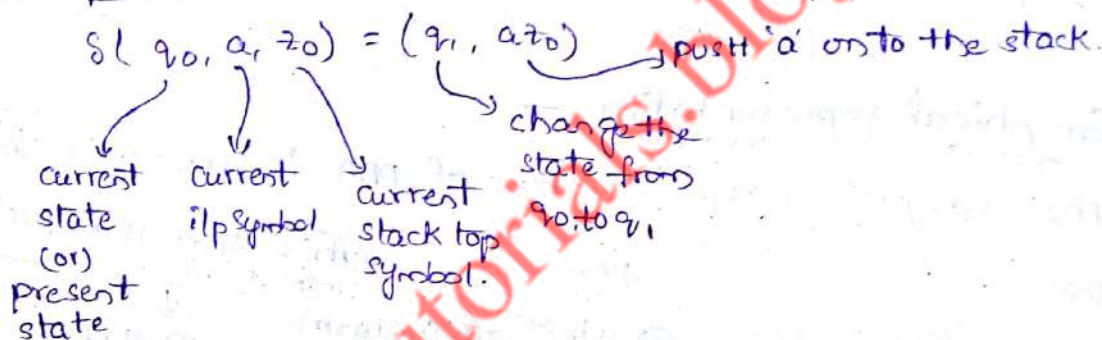
Example:- $\delta(q_0, a_1, z_0) = (q_1, p)$

www.IntukMaterials.com

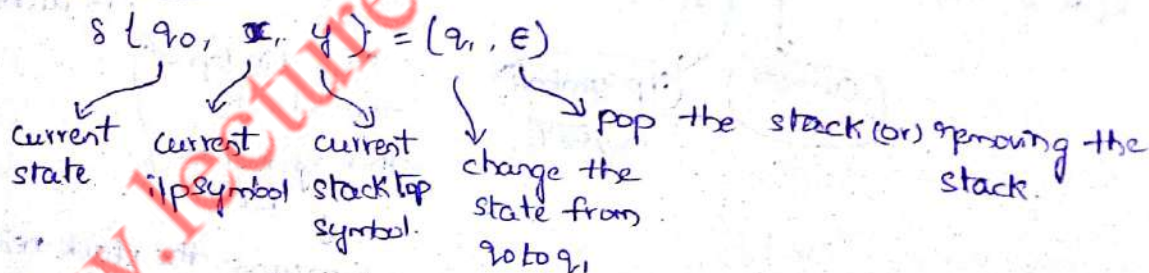


From the above figure, if we are reading the current input symbol 'a₂' at current state 's₁' and current stack symbol 'z₁' then after a move we will reach to state s₂ and there will be some new symbol on the top of the stack. This description can be represented as,

1) push operation:-



2) pop operation:-



Acceptance of PDA:-

There are two ways to accept a language by PDA. They are

- Accepted by empty stack.
- Accepted by final state.

Accepted by empty stack:-

The given language accepted by empty stack to be defined as $L(M) = \{w \mid \delta(q_0, w, z_0) \Rightarrow (p, \epsilon, \epsilon) \text{ for some } p \in M\}$ that is, if stack becomes empty after scanning entire string then it is accepted by PDA otherwise not.

Accepted by final string:-

the given language accepted by a final state is defined as

$$L(M) = \{ w \mid \delta(q_0, w, z_0) \xrightarrow{*} (p, \epsilon, f) \text{ for some } p \in F \text{ and } f \in \Gamma \}$$

that is, even though stack is not empty, after scanning it's 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:-

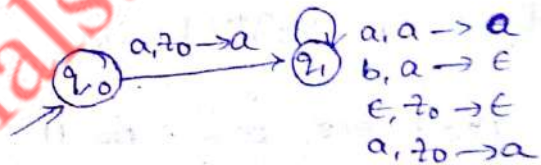
$$\delta: \delta(q_0, a, z_0) = (q_1, a, z_0)$$

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

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

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

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



$\therefore$ The PDA machine for the above language is defined as

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

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

$$\Gamma = \{z_0\}$$

$\delta:$

$$q_0 = \{q_0\}$$

$$z_0 = \{z_0\}$$

$$F = \{q_1\}$$

Read A's $\rightarrow$ push operation, Read B's $\rightarrow$ push operation

(1) Consider a String $w = \{abab\}$ Read a's

$$\delta(q_0, abab, z_0) = \delta(q_1, bab, az_0)$$

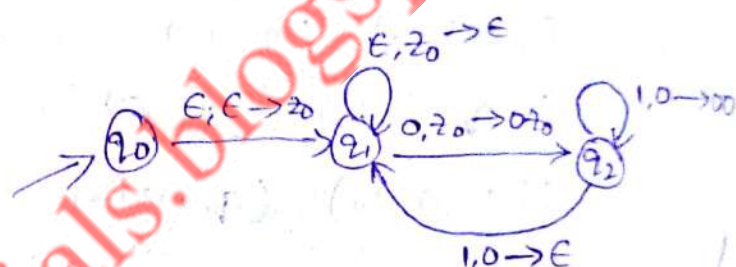
③ Design a PDA for the language $L = \{0^n 1^{2n} \mid n \geq 1\}$

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

Read one 0 $\rightarrow$ push

Read two 1's $\rightarrow$ pop

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



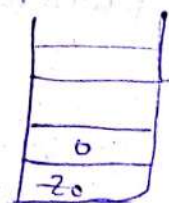
$$\delta(q_1, 0, z_0) = (q_2, 0z_0)$$

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

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

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

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



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

$$\delta(q_1, 1, 0) = (q_1, 00)$$

④ consider the string $w = \{001111\}$

$$\delta(q_1, 001111, z_0) = \delta(q_2, 0111, 0z_0)$$

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

$$= \delta(q_2, 11, 00)$$

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

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



stack

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

Read 0's $\rightarrow$ push

Read 1's $\rightarrow$ pop

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

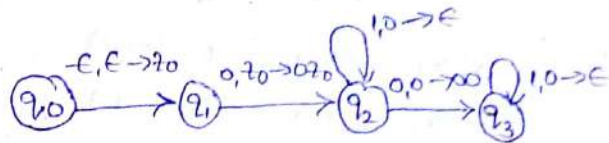
$$\delta(q_1, 0, z_0) = (q_2, 0z_0)$$

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

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

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

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



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

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

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

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 the string.

* In the mid position we can't read any input 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)

www.JntukMaterials.com

$$w = ab$$

$$w^R = ba$$

$$wcw^R = abcba$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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 input 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

$$\text{as } \delta(q_0, x, z_0) = (q, xp) \text{ where}$$

q_0 is current state

x is current input symbol

z_0 is current stack symbol

q is next state

xp shows top of the stack.

if δ denotes a unique transition for each input then PDA is said to be deterministic PDA

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

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

Non deterministic 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 input 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 $\delta(q, x) = (q', y)$

q_0 is current state
 x is current ip symbol
 z_0 is stack symbol
 q is next state
 xp is top of the stack.

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

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

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

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

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 $S \rightarrow OBB$

$B \rightarrow OS$

$B \rightarrow IS$

$B \rightarrow O$

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

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

www.JntukMaterials.com

$$P \rightarrow S \rightarrow OBB$$

$$B \rightarrow OS$$

$$B \rightarrow IS$$

$$B \rightarrow O$$

$$S = \{s\}$$

$$\text{Rule 1: } A \rightarrow \alpha$$

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

$$S \rightarrow OBB$$

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

$$B \rightarrow OS$$

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

$$B \rightarrow IS$$

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

$$B \rightarrow O$$

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

$$\text{Rule 2}$$

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

$$M = (Q, \Sigma, T, S, q_0, z_0, F)$$

$$Q = \{q\}$$

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

$$T = \{s, B, O, I\}$$

S = it is a transition symbol. defined as

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

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

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

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

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

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

$$q_0 = \{q\}$$

$$z_0 = \{z_0\}$$

$$F = \{\}$$

2) construct a pda for the following CFG

$S \rightarrow OSI$
 $S \rightarrow A$
 $A \rightarrow IAO | S | \epsilon$

Sol: The Given CFG is

$S \rightarrow OSI$
 $S \rightarrow A$
 $A \rightarrow IAO$
 $A \rightarrow S$
 $A \rightarrow \epsilon$

elimination of ϵ -production:-

$A \rightarrow \epsilon \times$ $A \rightarrow IAO$
 $S \rightarrow A$ $A \rightarrow IEO$
 $S \rightarrow \epsilon \times$ $A \rightarrow IO$
 $S \rightarrow OSI$ $A \rightarrow S$
 $S \rightarrow OEI$ $A \rightarrow \epsilon \times$
 $S \rightarrow OI$

$S \rightarrow OSI | OI$
 $S \rightarrow A$
 $A \rightarrow IAO | IO$
 $A \rightarrow S$

elimination of unit productions:-

$S \rightarrow A \times$ $A \rightarrow S \times$
 $S \rightarrow IAO | IO$ $A \rightarrow OSI | OI$

$\therefore$ The resultant CFG is.

$S \rightarrow IAO$
 $S \rightarrow IO$
 $A \rightarrow OSI$
 $A \rightarrow OI$

$\therefore$ The Simplified CFG is.

$S \rightarrow IAO$
 $S \rightarrow IO$
 $A \rightarrow OSI | IAO | IO$
 $A \rightarrow OI$

Method-2
 $P \rightarrow I$
 $Q \rightarrow O$

$S \rightarrow IAO$ $S \rightarrow IO$ $A \rightarrow OSI$ $A \rightarrow IO$ $S \rightarrow OSI$
 $S \rightarrow IAQ \checkmark$ $S \rightarrow IQ \checkmark$ $A \rightarrow OSP \checkmark$ $A \rightarrow IQ \checkmark$ $S \rightarrow OSP \checkmark$
 $A \rightarrow IAO$ $A \rightarrow OI$ $S \rightarrow OI$
 $A \rightarrow IAQ \checkmark$ $A \rightarrow OP \checkmark$ $S \rightarrow OP \checkmark$

$\therefore$ The simplified CFG in GNF is

$S \rightarrow IAQ$ $A \rightarrow OSP$
 $S \rightarrow IQ$ $A \rightarrow IQ$
 $S \rightarrow OSP$ $A \rightarrow IAQ$
 $S \rightarrow OP$ $A \rightarrow OP$
 $P \rightarrow I$
 $Q \rightarrow O$

Rule:-1
 $A \rightarrow \alpha$

$\therefore$ The PDA is

$S \rightarrow IAQ$ $S \rightarrow OSP$ $A \rightarrow OSP$
 $\delta(q_1, \epsilon, S) = (q_1, IAQ)$ $\delta(q_1, \epsilon, S) = (q_1, OSP)$ $\delta(q_1, \epsilon, A) = (q_1, OP)$
 $S \rightarrow IQ$ $S \rightarrow OP$ $A \rightarrow IQ$
 $\delta(q_1, \epsilon, S) = (q_1, IQ)$ $\delta(q_1, \epsilon, S) = (q_1, OP)$ $\delta(q_1, \epsilon, A) = (q_1, IQ)$

$A \rightarrow 1A0$
 $S(q, \epsilon, A) = (q, A0)$
 $A \rightarrow 0P$
 $S(q, \epsilon, A) = (q, 0P)$
 $P \rightarrow 1$
 $S(q, \epsilon, P) = (q, 1)$
 $Q \rightarrow 0$
 $S(q, \epsilon, Q) = (q, 0)$

Rule 2: Terminal

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

$T = \{0, 1\}$

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

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

method: 2

The Given CFG is $S \rightarrow 0S1$

$S \rightarrow A$

$A \rightarrow 1A0$

$A \rightarrow S$

$A \rightarrow \epsilon$

The resultant PDA is $S \rightarrow 0S1$

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

$S \rightarrow A$

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

$A \rightarrow S$

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

$A \rightarrow 1A0$

$A \rightarrow \epsilon$

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

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

Construct PDA for the following CFG $S \rightarrow aABB/aAA$

$A \rightarrow aBB/a$

$B \rightarrow bBB/bA$

sol: The Given CFG is $S \rightarrow aABB$

$S \rightarrow aAA$

$A \rightarrow aBB$

$A \rightarrow a$

$B \rightarrow bBB$

$B \rightarrow A$

elimination of unit production:-

$B \rightarrow Ax$

$B \rightarrow aBB$

$B \rightarrow a$

$\therefore$ After eliminating unit production $B \rightarrow A$, The resultant

CFG in GNF is $S \rightarrow aABB$

$B \rightarrow aBB$

$S \rightarrow aAA$

$B \rightarrow a$

$A \rightarrow aBB$

$A \rightarrow a$

$B \rightarrow bBB$

THE PDA IS
 $s \rightarrow aABB$
 $s(q, \epsilon, s) = (q, aABB)$

$s \rightarrow aAA$
 $s(q, \epsilon, s) = (q, aAA)$

$A \rightarrow aBB$
 $s(q, \epsilon, A) = (q, aBB)$

$A \rightarrow a$
 $s(q, \epsilon, A) = (q, a)$

$B \rightarrow bBB$
 $s(q, \epsilon, B) = (q, bBB)$

$B \rightarrow aBB$
 $s(q, \epsilon, B) = (q, aBB)$

$B \rightarrow a$
 $s(q, \epsilon, B) = (q, a)$

conversion of PDA to CFG :-

* If $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is a PDA. then there 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 $z_0 \in \Gamma$.

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

Algorithm :-

Rule 1 :- The start symbol production rule can be $s \rightarrow [q, z_0, q']$ 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 $s(q, a, z_0) = (q', \epsilon)$

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

3) If there exists a move of PDA as $\delta(q, a, z_0) = (q', z_1 z_2 z_3 \dots)$
 then the production rules can be written as

$$[q, z_0, q'] \Rightarrow a[q_1, z_1, q_1][q_1, z_2, q_2][q_2, z_3, q_3] \dots [q_n, z_n, q_n]$$

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

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

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

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

$$q_0 < q_0 \quad P_3: [q_0, S, q_0] \rightarrow 1 [q_0, A, q_0] [q_0, S, q_0]$$

$$q_0 < q_1 \quad P_4: [q_0, S, q_0] \rightarrow 1 [q_0, A, q_1] [q_0, S, q_0]$$

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

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

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

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

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

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

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

Now for $\delta(q_0, 0, A) = (q_1, A)$ using Rule ② of Algorithm

www.JntukMaterials.com

$$P_9: [q_0, A, q_0] \Rightarrow \emptyset [q_0, A, q_1] [q_1, A, q_0]$$

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

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

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



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

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

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

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

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



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

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

PDA with two stacks : —

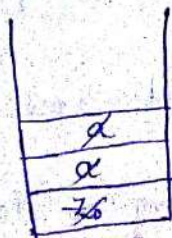
① PDA with one stack :

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

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.

① $L = \{a^n b^n c^n \mid n \geq 1\}$

www.JntukMaterials.com

read a's $\rightarrow$ push
read b's $\rightarrow$ pop
read c's $\rightarrow$ no change



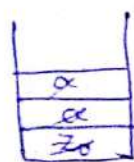
stack.

a a b b c c \$
↑ ↑ ↑ ↑ ↑ ↑ ↑

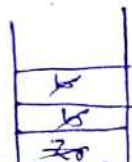
when read c there is no change in stack without completion of reading string 'w' the stack is empty. so, string is not accepted.

② PDA with two stacks:-

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



stack 1



stack 2

w = a a b b c c \$
↑ ↑ ↑ ↑ ↑ ↑ ↑

read a's $\rightarrow$ push (on stack₁)

read b's $\rightarrow$ push (on stack₂)

read c's $\rightarrow$ pop (a from stack₁ and 'b' from stack₂)

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

$\therefore$ 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.