so, now,

step 3 : Generate new production rules by ~~rewritting~~ them.

~~L(G)'~~ = $P'' = \begin{cases} (1)\ S \rightarrow aSa \\ (2)\ S \rightarrow bSb \\ (3)\ S \rightarrow aa \\ (4)\ S \rightarrow bb. \end{cases}$

Ex2 : Consider following Grammar,

$S \rightarrow a \mid Xb \mid aYa$

$X \rightarrow Y \mid \epsilon$

$Y \rightarrow b \mid X.$

step 0 :-

$P = \{$ (1) $S \rightarrow a$

(2) $S \rightarrow Xb$

(3) $S \rightarrow aYa$

(4) $X \rightarrow Y$

(5) $X \rightarrow \epsilon$ ← ϵ production

(6) $Y \rightarrow b$

(7) $Y \rightarrow X$

step 1 : Identify of ϵ - Production & Nullable Non - Terminals & Unit production.

Since $X \rightarrow \epsilon$, & $X \Rightarrow \epsilon$

X becomes Nullable Non - Terminal.

so, remove ϵ - Production from Grammar.

Now, $Y \Rightarrow X \Rightarrow \epsilon$

∴ Y becomes Nullable Non - Terminal.

$L(G) = L(G)'$

$P' = \{$
1. $S \to a$
2. $S \to Xb$
3. $S \to aYa$
4. $X \to Y$
5. $Y \to b$
6. $Y \to X$

new list $P' =$
1. $S \to a$
2. $S \to a$
3. $S \to b$
4. $S \to aa$
5. $S \to aYa$
6. $X \to Y$
7. $Y \to b$
8. $Y \to X$

step 2 : Nullify the effect of ϵ in the Grammar.
Since X is nullable,

follow upwards.

$S \to Xb$	$Y \to X$	$X \to Y$ $\cancel{S \to Xb}$	$S \to aYa$	S
$\boxed{S \to b}$	~~dropped~~ $Y \to \epsilon \; \alpha$	$X \to \epsilon_{\alpha}$	$S \to aa$	

step 3 : ~~Rewriting Production rules from step2 as~~ Treat Unit Production rule.

$P'' = \{$
1. $S \to a$
2. $S \to Xb$
3. $S \to aYa$
4. $X \to Y$
5. $X \to b$
6. $S \to b$. $\}$

$X \to Y$

$Y \to X | b$

• Normal Forms.

To standardize the way CFG's have been written, Normal Forms been proposed by :-
ⓐ Chomsky Normal Form (CNF)
ⓑ Greibach Normal Form (GNF).

Ⓐ Chomsky Normal Form (CNF).

$$\left.\begin{array}{|l|}\hline A \rightarrow BC \\ A \rightarrow a \\ \hline A \rightarrow \epsilon \\ \hline\end{array}\right.$$

$A \rightarrow BC$, $A \rightarrow a$ } rule.

$A \rightarrow \epsilon$ ← exception.

*rule:- Only two Non-Terminal symbols or either only on terminal.

when ϵ is in $L(G)$.
we assume that S does not appear on RHS of any production.
Eg :- G is

$S \rightarrow AB \mid \epsilon$ } CNF
$A \rightarrow a$ } CNF
$B \rightarrow b$. } CNF

For eg :- $A \rightarrow aa$ ✗ CNF
$A \rightarrow ABC$ ✗

Construction of G in CNF

Step 1 : Elimination of null productions & unit productions using previous method.
Let G is (V, T, P, S).

step 2 : Elimination of terminals on R.H.S.
step 3 : Restricting number of variables on RHS.

Q. Consider, grammar as,

$S \to aSa$ }
$S \to bSb$ } Not in CNF
$S \to a$ — is CNF
$S \to b$ — is CNF
$S \to aa$ }
$S \to bb$ } Not in CNF

Soln : step 0 : $P = \{$ ① $S \to aSa$
② $S \to bSb$
③ $S \to a$
④ $S \to b$
⑤ $S \to aa$ }
⑥ $S \to bb$ }

Introduce new non terminal which is derivable to prev form & not change or alter G.

$S \to aSa$

∴ Introduce New NT A. such that $\boxed{A \to a}$
∴ So we get new production
∴ $S \to ASA$

∴ Introduce New NT such that
$\boxed{C \to SA}$
such that,
$\boxed{S \to AC}$
IN CNF

$S \to bSb$

∴ $\boxed{B \to b}$
∴ New production.
∴ $S \to BSB$

∴ Introduce New NT such that,
$\boxed{D \to SB}$
such that,
$\boxed{S \to BD}$
∴ IN CNF

$S \to aa$

Since $A \to a$
∴ $\boxed{S \to AA}$
IN CNF

$S \to bb$

Since $B \to b$
∴ $\boxed{S \to BB}$
IN CNF.

Now,

$G' = (\{S, A, B, C, D\}, \{a, b\}, P', S)$

$P' = \{$
1. $S \to AC$
2. $S \to BD$ — (2) moves
3. $S \to AA$
4. $S \to BB$
5. $S \to a$
6. $S \to b$
7. $A \to a$
8. $B \to b$
9. $C \to SA$
10. $D \to SB$ $\}$.

(B) **Greibach Normal Form (GNF).** (Alter Restriction of only two Non Terminal)

A context Free grammar is said to in GNF, if all productions are in following form :-

$$\cancel{A \to a} \quad A \to aX$$

A : is Non Terminal symbol

a : is terminal symbol

X : sequence of Non terminal symbol (it may be empty).

Eg :- $A \to a$, $A \to aAA$, $a \to bA$, $B \to b$, $A \to aABABAB$.

• PUSHDOWN AUTOMATA :

For Regular Language - Finite Automata (NFA or DFA)

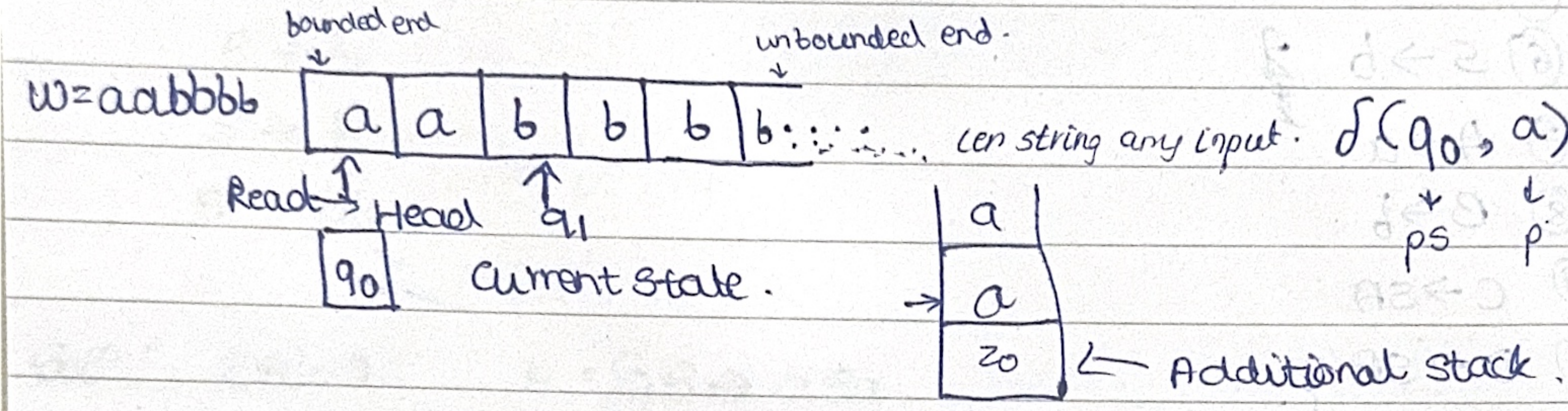
For CFL - Pushdown Automata → mathematical model.

Pushdown Machine

↓

7 - tuple.

Finite Control.



len string any input. $\delta(q_0, a) = q_2$.

ps p NS

• Formal definition of PDA.

The PDA is as :

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

Logic:

$A = (\{q_0\}, \{a, b\}, \overset{\Gamma}{\{a, z_0\}}, \delta, q_0, z_0, F)$.

Q : A finite set of states.

Σ : A finite set of input symbols.

Γ : A Finite stack alphabet or pushdown symbols.

δ : the transition function $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ to set of finite subsets of $Q \times \Gamma^*$

q_0 : the start state

Z_0 : the start symbol (pushdown symbol)

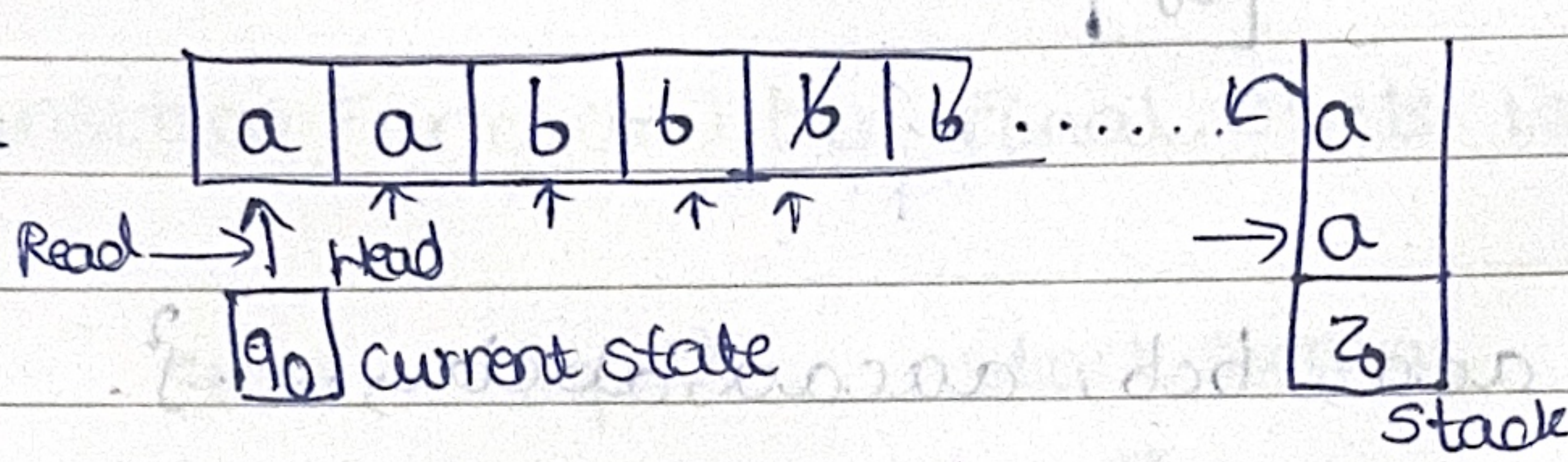
F : the set of accepting state or final states

keywords : Read
push
pop

• Move Table -

Problem : -

step 0 : -



step 1 : $A \{ \{q_0\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, F \}$.

$L = \{ a^n b^n / n > 0 \}$

$L = \{ a^1 b^1, a^2 b^2, a^3 b^3, \ldots \}$

$L = \{ ab, aabb, aaabbb, \ldots \}$.

step 2 : Logic -

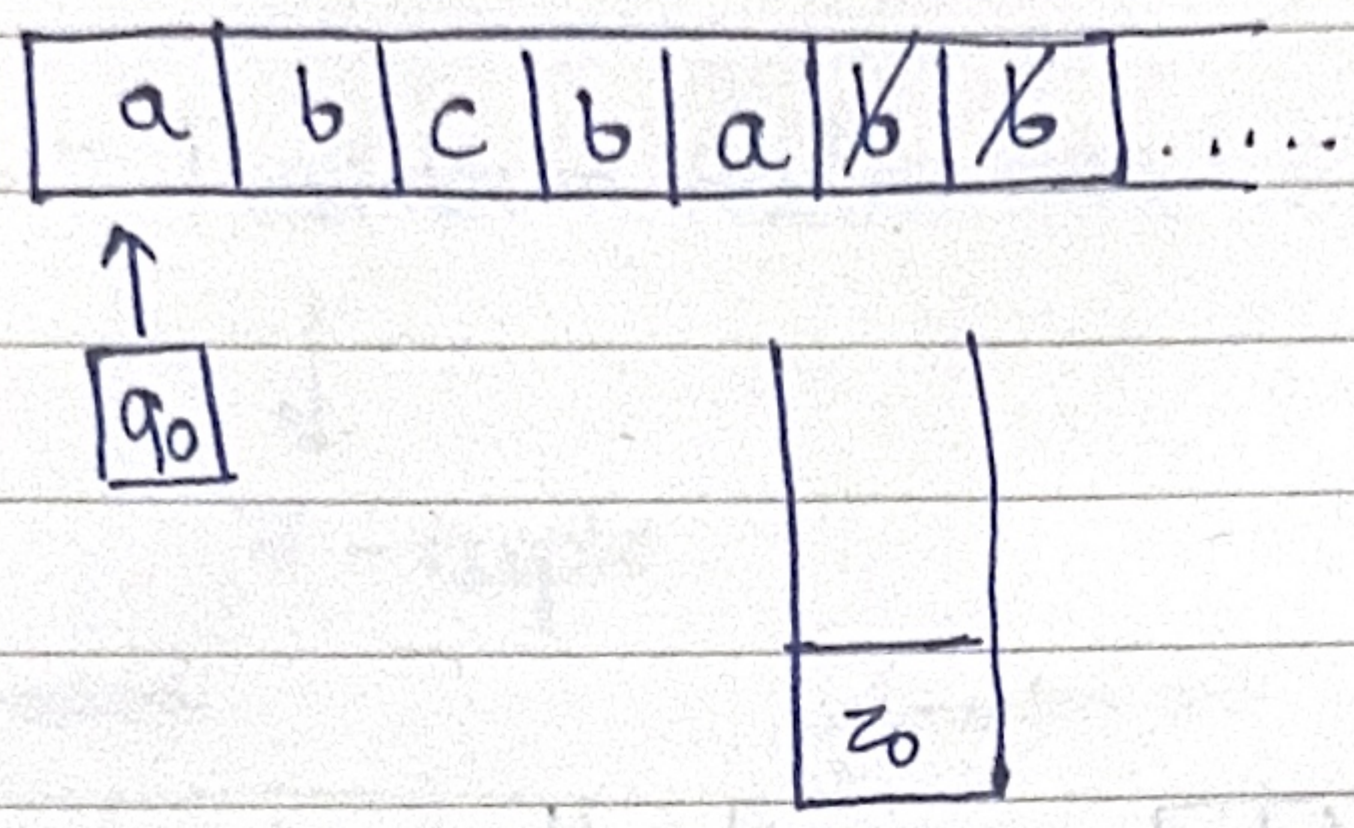
1. Read a & pushing a onto the stack
2. Read b & popping from the stack
3. Keep repeating step 1 & step 2 until (w is read).
   b from the tape.

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ - Push	$\delta(q_0, b, aa) = (q_1, Z_0)$ Pop.

Q. Design PDA for the language $L = \{ xcx^r \mid x \in \{a,b\}^* \}$.

↳ Odd Palindrome string problem.

Sol^n: Step 0: $\Sigma = \{a, b, c\}$ ← marker.



$L(A) = \{ c, aca, bcb, aacaa, abcba, \ldots \}$.

xabcacxacccb

x acab.

Step 1: Let $A = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is a PDA.

where,

$Q = \{q_0, q_1, q_f\}$,

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

$\Gamma = \{a, b, z_0\}$,

$F = \{q_f\}$

Step 2: Logic.

a). Read a ~~and~~ or b and push onto stack.

b) When the machine ~~is ready~~ reads c change state from q_0 to q_1, there by skipping c.

* KLP Mishra → PDA

c) when pda is going to reads 'a' from x^r, it must pop 'a' and if machine reads "b" from x^r, it must pop "b".

d)

δ (Transition Function) by Final State is:

Step 3:

Move No.	State	input	Stack symbol	move.
1	q_0	a	Z_0	(q_0, aZ_0)
2	q_0	b	Z_0	(q_0, bZ_0)
3	q_0	a	a	(q_0, aa)
4	q_0	b	b	(q_0, bb)
5	q_0	a	a	(q_0, ab)
6	q_0	b	b	(q_0, ba)
7	q_0	c	Z_0	(q_1, Z_0)
8	q_0	c	a	(q_1, a)
9	q_0	c	b	(q_1, b)
10	q_1	a	a	(q_1, ϵ)
11	q_1	b	b	(q_1, ϵ)
12	q_1	ϵ	Z_0	(q_f, Z_0)

step 4:

$(q_0, abcba, Z_0)$

$\vdash (q_0, bcba, \underline{a}Z_0)$

$\vdash (q_0, cba, \underline{ba}Z_0)$

$\vdash (q_1, ba, \underline{ba}Z_0)$

$\vdash (q_1, a, \underline{a}Z_0)$. pop b

$\vdash (q_f, \epsilon, Z_0)$

$\vdash (q_F, Z_0)$.

- Deterministic and Non - Deterministic PDA.

DPDA :

transitfunction is :

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

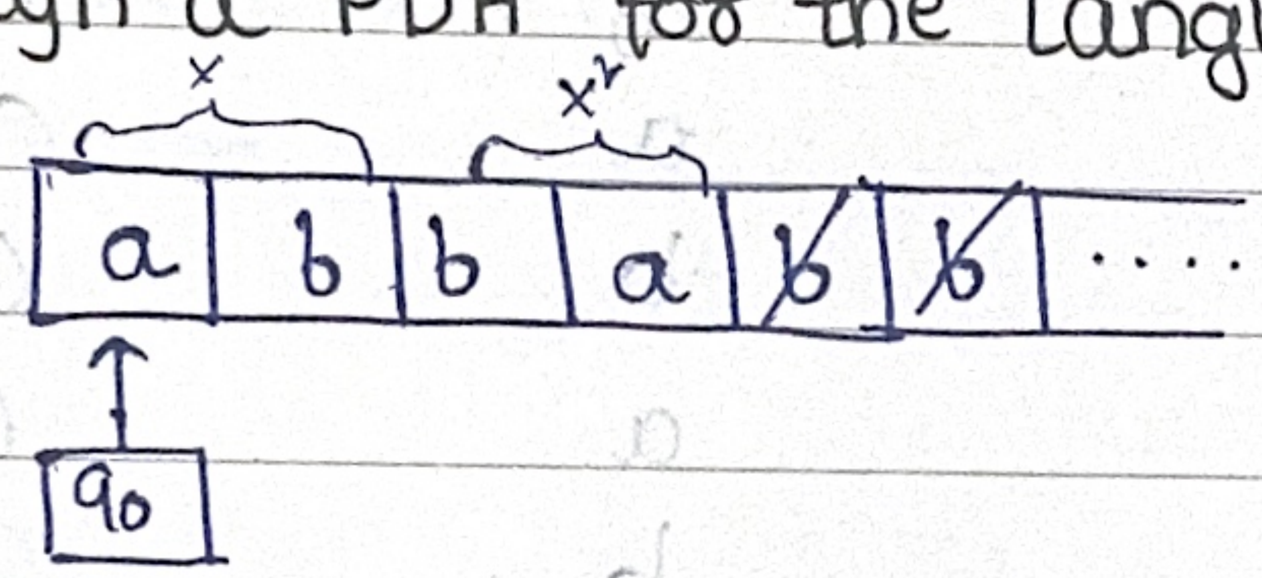
e.g. $\delta(q, a, z)$ is either empty or singleton.

$\delta(q, a, z) \neq \emptyset$

DPDA $\neq$ NPDA.

NPDA $\Rightarrow$ Q. Design a PDA for the language $L = \{ x x^r \mid x \in \{a, b\}^* \}$.

Soln : step0 :



Step I : Let,

$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA.

where,

$Q = \{q_0, q_1, q_f\}$,

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

$\Gamma = \{a, b, Z_0\}$

$F = \{q_f\}$

Step 2: Logic :

Step 3: δ (Transition Function) by Final State is :

Move No.	State	input:	stack symbol	move
1	q_0	a	Z_0	$\{(q_0 . aZ_0), (q_1 . Z_0)\}$
2	q_0	b.	Z_0	$\{(q_0 . bZ_0), (q_1 . Z_0)\}$
3	q_0	a	a	$\{(q_0 . aa), (q_1, a)\}$
4	q_0	b	a	$\{(q_0 . ba), (q_1, a)\}$
5	q_0	a	b	$\{(q_0 . ab), (q_1, b)\}$
6	q_0	b	b	$\{(q_0 . bb), (q_1, b)\}$
7	q_0	ϵ	Z_0	$\{(q_1, Z_0)\}$
8	q_0	ϵ	a	$\{(q_1, a)\}$
9	q_0	ϵ	b	$\{(q_1, b)\}$
10	q_1	a	a	$\{(q_1, \epsilon)\}$
11	q_1	b	b	$\{(q_1, \epsilon)\}$
12	q_1	ϵ	Z_0	$\{(q_f, Z_0)\}$.

Step 4 : $(q_0, abba, Z_0)$

$\vdash ($

~~Context Sensitive Language~~

- CFG to PDA.

- $G = (V, T, P, S)$ to $A = (Q, \Sigma, \Gamma, \delta, q, Z_0, F)$

Rule 1: $A \to \alpha$.

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

Rule 2: $A \to a$ {for terminal symbol}

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

1. Construct a PDA for the CFG {NPDA}

$S \to 0BB$

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

Test whether 010^4 is in $N(A)$.

Solⁿ: We construct PDA A as :

$A = (Q, \Sigma, \Gamma, \delta, q, Z_0, F)$

$Q = \{q\}$

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

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

$Z_0 = S$

$F = \phi$

①

∴ Move Table.

Move No.	State	input	stack symbol	move
1	q	ϵ	S	$\{(q, 0BB)\}$
2	q	ϵ	B	$\{(q, 0S), (q, 1S), (q, 0)\}$
3	q	0	0	$\{(q, \epsilon)\}$
4	q	1	1	$\{(q, \epsilon)\}$

∴ Instantaneous Description (ID)

$w = 0\ 1\ 0000\ b$

∴ $(q, 010000, S)$
$\vdash (q, 010000, 0BB)$
$\vdash (q, 10000, BB)$
$\vdash (q, 10000, 1SB)$
$\vdash (q, 0000, SB)$.
$\vdash (q, 0000, 0BBB)$
$\vdash (q, 000, BBB)$
$\vdash (q, 000, 0BB)$
$\vdash (q, 00, BB)$
$\vdash (q, 00, 0B)$
$\vdash (q, 0, B)$
$\vdash (q, 0, 0)$
$\vdash (q, \epsilon)$

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

$\delta(q, 0, S)$ ϵ N.T.
↳ results $(q, 0BB)$.

$\delta(q, 0, 0)$ → pop (read 0)
↳ (q, ϵ).

$\delta(q, \epsilon, B)$
↳ $(q, 1S)$

$\delta(q, 1, 1)$ → pop (read 1).
↳ (q, ϵ)

pop pop

0	1	0	0	0	0	b	...

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

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

- Rule 1 : $A \to \alpha$
$\delta(q, \epsilon, A) = (q, \alpha)$
- Rule 2 : $A \to a$
$\delta(q, a, a) = (q, \epsilon)$

Example 2: Convert grammar.

$S \to aSb \mid A$
$A \to bSa \mid S \mid \epsilon$

To a PDA that accepts the same language by empty stack.