Pumping Lemma for Context free languages :-

→ The theorem of pumping Lemma for Context free languages says that in any sufficiently long string in a CFL it is possible to find atmost 2 short nearby substrings that we can pump i.e., we may repeat both of strings i times for any integer i the resulting string will still be in the language.

→ Let L be a CFL. Then there exists a constant n such that if z is any string in L such that $|z|$ is atleast n, then we can write $z = uvwxy$ subject to the following conditions :-

i) $|vwx| \leq n$. That is the middle portion is not too long.

ii) $vx \neq \epsilon$. Since v & x are the pieces to be "pumped" this condition says that atleast one of the strings we pump must not be empty.

iii) For all $i \geq 0$, uv^iwx^iy is in L. That is the two strings v and x may be "pumped" any number of times including 0 & the resulting string will still be a member of L.

Eg :- show that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not Context Free Language.

Sol :- Assume $n = 4$

$z = a^4 b^4 c^4$

$= aaaa\,bbbb\,cccc$

Case i :- $\underbrace{aaaa}_{u}\,\underbrace{bbb}_{v}\underbrace{b}_{w}\underbrace{c}_{x}\underbrace{ccc}_{y}$

$|vwx| \leq n \qquad vx \neq \epsilon$

$4 \leq 4 \checkmark \qquad bb \neq \epsilon$

for $i = 0$

$z = aaaa\,bb\,cccc \notin L$

Case ii :- $\underbrace{aaa}_{u}\underbrace{a}_{v}\underbrace{b}_{w}\underbrace{b}_{x}\underbrace{bb\,cccc}_{y}$

$|vwx| \leq n \qquad vx \neq \epsilon$

$4 \leq 4 \checkmark \qquad aab \neq \epsilon$

for $i = 0$

$z = aaabbbcccc \notin L$

Eg:- Show that $L = \{ww \mid w \in \{0,1\}^*\}$ is not Context Free Language

Sol:- Assume $n = 3$ string $z = 0^n 1^n 0^n 1^n$.

$z = 000\,111\,000\,111$

$z = uv^i wx^i y$

Case i:- $z = \underbrace{000}_{u}\underbrace{111}_{v}\underbrace{000}_{w}\underbrace{111}_{x}\underbrace{}_{y}$

$|vwx| \leq n \qquad vx \neq \epsilon$

$3 \leq 3 \qquad 01 \neq \epsilon$

for $i = 0$

$z = 001\,000\,111 \notin L$

Case ii:- $z = \underbrace{000}_{u}\underbrace{111}_{v}\underbrace{}_{w}\underbrace{}_{x}\underbrace{000\,111}_{y}$

$|vwx| \leq n \qquad vx \neq \epsilon$

$3 \leq 3 \qquad \epsilon 1 \neq \epsilon$

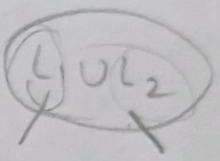
for $i = 0$

$z = 000\,1000111 \notin L$

Closure properties of Context-Free Language

Union

→ Let L_1 and L_2 be two CFL's with grammars $G_1 = (V_1, T_1, S_1, P_1)$ and $G_2 = (V_2, T_2, S_2, P_2)$ respectively.



Assume V_1 and V_2 are disjoint

Consider the Language $L(G_3)$ generated by Grammar $G_3 = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, S_3, P_3)$

where $P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1 | S_2\}$

Example :-

$L_1 = \{a^n b^n\}$

$S_1 \to aS_1b | ab$

~~$L_2 = \{ww^R\}$~~ $\{c^n d^n\}$

$S_2 \to cS_2d | cd$

~~$S_2 \to aS_2a | bS_2b | \epsilon$~~

Union : $L = \{a^n b^n\} \cup \{c^m d^m\}$

$S \to S_1 | S_2$

Concatenation

Let L_1 and L_2 be two CFL's with grammars $G_1 = (V_1, T_1, S_1, P_1)$ and $G_2 = (V_2, T_2, S_2, P_2)$ respectively

Assume V_1 and V_2 are disjoint

Consider language $L(G_4)$ generated by grammar $G_4 = (V_1 \cup V_2 \cup \{S_4\}, T_1 \cup T_2, S_4, P_4)$

where $P_4 = P_1 \cup P_2 \cup \{S_4 \rightarrow S_1 S_2\}$

Then $L(G_4) = L(G_1) \cdot L(G_2)$

$L(G_1)\, L(G_2)$

Kleen - closure

Let L_1 be the CFL's with grammars

$G_1 = (V_1, T_1, S_1, P_1)$

Consider language $L(G_5)$ generated by grammar

$G_5 = (V_1 \cup \{S_5\}, T_1, S_5, P_5)$

Where $P_5 = P_1 \cup \{S_5 \to S_1 S_5 | \in\}$

Then $L(G_5) = L(G_1)^*$

a^*

$S \to aSa | \varepsilon | a$

The Context Free Languages are not closed under intersection and complementation.

$L_1 = \{a^n b^m c^m : n \geq 0, m \geq 0\}$ $\quad a^n b^n c^m \; n$

$L_2 = \{a^m b^n c^n : n \geq 0; m \geq 0\}$ $\quad a^m b^n c^n$

$a^n b^n c^n$

Both L_1 & L_2 are context free Languages

Grammar

L_1 : $S \to S_1 S_2$

$S_1 \to a S_1 b \mid \epsilon$

$S_2 \to c S_2 \mid \epsilon$

$\cap$

Grammar

L_2 : $S \to S_1 S_2$

$S_1 \to a S_1 \mid \epsilon$

$S_2 \to b S_2 c \mid \epsilon$

$L_1 \cap L_2 = \{a^n b^n c^n : n \geq 0, m \geq 0\}$ is not Context Free Language

proved by Pumping Lemma already.

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

CFL is not closed under intersection and Complementation.

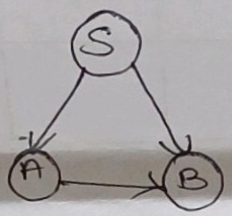
Decision Properties of Context Free languages :-

1) Finiteness & Infiniteness :-

→ To check whether a given CF language G is finite or infinite, Let us construct a CFG $G' = \{V', T, P', S\}$ in CNF that has no useless symbols & generates $L(G) - \varepsilon$.

→ Finiteness can be checked by constructing a directed graph for variables in the grammar & if the graph has no cycles then the language is said to be finite otherwise infinite.

Eg :- i) Is the grammar $S \to AB$, $A \to BB$, $B \to a$ is finite?



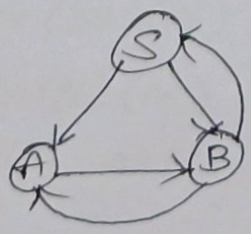
For $S \to AB$, place edge from S to A & S to B.

For $A \to BB$, place edge from A to B.

B derives only terminals there is no outgoing edge from B.

Since there is no cycle in the graph, the language is finite for the given grammar.

Eg :- ii) $S \to AB$, $A \to BB$, $B \to AS$.



There is cycle in graph. Hence the language is infinite for the grammar.

Membership :- using CYK algorithm

Eg :- $S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid a$

String :- "baaba"

Sol :-

	5	4	3	2	1
1	{A,S,C}	—	—	{A,S}	{B}
2	{A,S,C}	{B}	{B}	{AC}	
3	{B}	{SC}	{AC}		
4	{AS}	{B}			
5	{AC}				

1,2 ⇒ (11)(22)
B . AC
(BA)(BC)
A S

2,3 ⇒ (22)(33)
⇒ (AC)(AC)
⇒ (AC)(AA)(CC)
× × B

3,4 ⇒ (3,3)(4,4)
⇒ (AC)(B)
⇒ (AB)(CB)
⇒ S C

4,5 ⇒ (44)(55)
⇒ (B).(AC)
⇒ (BA)(BC)
⇒ A S

1,3 ⇒ (11)(2,3) or (1,2)(33)
BB or (AS)(AC)
× (AA)(SA)(AC)(SC)
× × × ×

2,4 ⇒ (22)(3,4) or (2,3)(4,4)
(AC)(SC) or (B)(B)
(AS)(AC)(CS)(CC) BB
× × × B ×

3,5 ⇒ (33)(45) or (3,4)(55)
(AC)(AS) or (SC)(AC)
(AA)(AS)(CA)(CS) or (SA)(SC)(CA)(CC)
× × × × × × × B

1,4 :-

(1 1) (2,4)
(B)(B)
X

(12) (34)
(AS)(SC)
(AS) (AC)(SS)(SC)
X X X X

(1 3) (44)
ϕ . (AS)
ϕ

2,5 :-

(22) (35)
(AC) B
(AB) (AC)
SC X

(23) (45)
(B)(AS)
(BA)(BS)
A X

(24) (55).
(B)(AC)
(BA)(BC)
A S

1,5 :-

(1 1)(25)
(B) (ASC)
(BA)(BS) (BC)
A X S

(1,2)(3,5)
(AS) (B)
(AB)(SB)
SC

(1,3)(4,5)
ϕ. AS
ϕ

(1,4)(5,5).
ϕ. AC
ϕ.

Emptiness :-

Eg :- $S \rightarrow AX / XY$
$X \rightarrow AA / BB$
$A \rightarrow a$
$B \rightarrow b$
$Y \rightarrow bY / BB$

Consider Terminals on RHS. $A \rightarrow a$, $B \rightarrow b$ substitute the values of a in A & b in B.

$S \rightarrow AX$

$\rightarrow aX$

$\rightarrow aAA$

$\rightarrow aaa$

$S \rightarrow XY$

$\rightarrow ABY$

$\rightarrow abY$

$\rightarrow abbY$

$\rightarrow abbBB$

$\rightarrow abbbb.$

Push Down Automata

Definition of PDA

→ A PDA is a way to implement a Context Free Grammar in a way similar to we design FA for Regular Grammar

→ It is more powerful than FSM

→ FSM has a very limited memory but PDA has more memory

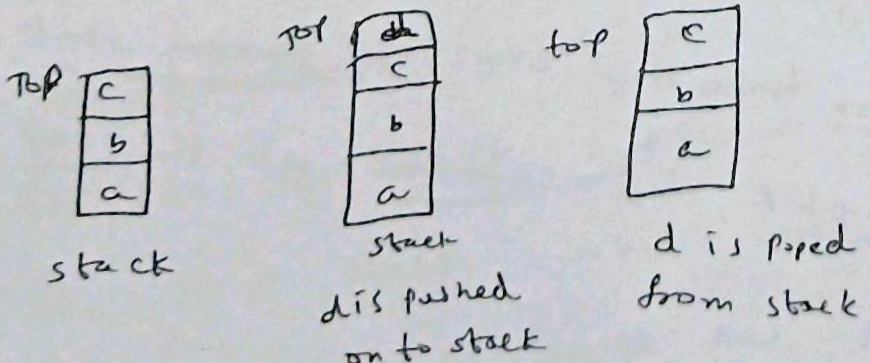
→ PDA = Finite state Machine (FSM) + A stack.

→ A stack is a way we arrange elements one on Top of another.

→ A stack does two basic operations:

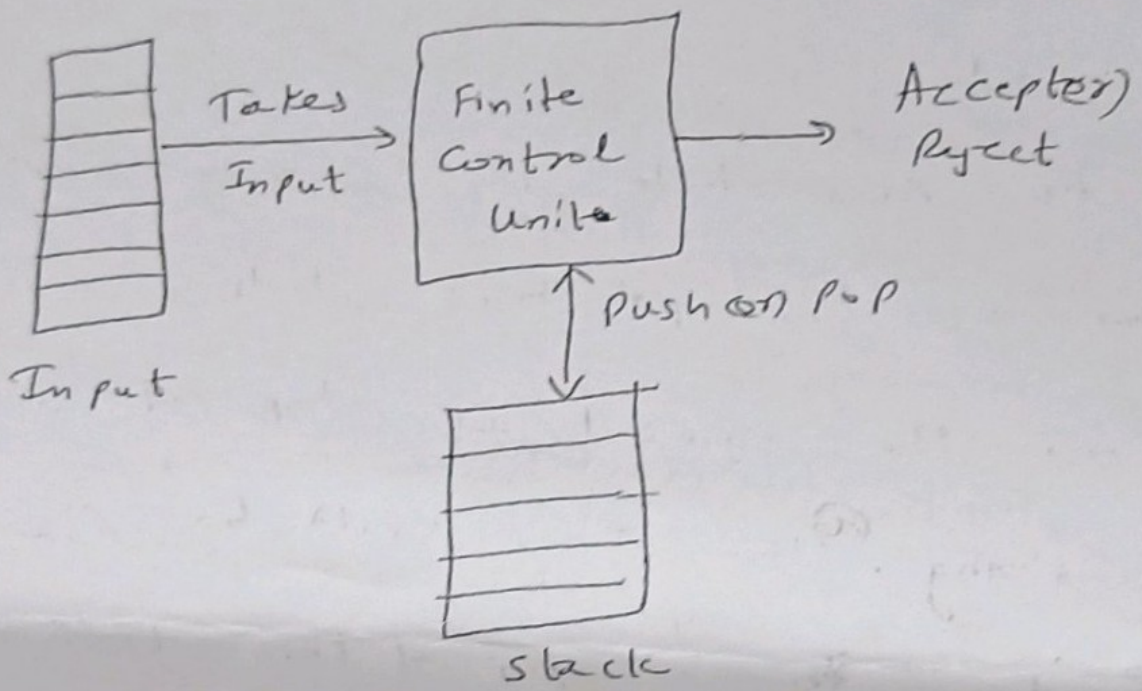
Push: A new element is added at the Top of stack.

Pop: The top element of stack is read and removed.



A Pushdown Automata has 3 components:

1) An Input tape
2) A Finite Control unit
3) A stack with infinite size



→ A Finite-Control unit reads input, one symbol at a time.

→ The pushdown automaton is allowed to observe the symbol at top of stack and to base its transition on its current state, the input symbol, and symbol at the top of stack.

In one transition, the push down automaton:

(i) Consumes from the input, the symbol that it uses in the transition. If ϵ is used for the input, then no input symbol is consumed.

(ii) Goes to new state, which may or may not be same as the previous state.

(iii) Replaces the symbol at top of stack by any string. (a) The string could be ϵ, which corresponds to a pop of the stack.

(b) It could be the same symbol that appeared at the top of stack i.e no change to the stack.

(c) It could also replace the top stack symbol by another symbol, which in effect changes the top of stack but does not push or pop it.

(d) The top stack symbol could be replaced by two or more symbols, which has the effect of changing the top stack symbol, and then

pushing one or more new symbols onto the stack.

Formal Definition of PDA

→ PDA involves seven components.

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

Q: A Finite set of states

Σ: A Finite set of Input symbols

Γ: A Finite ~~set of~~ stack alphabet

δ: The Transition Function

$\delta(q, a, X)$

q: state in Q

a: Input symbol or ϵ

X: stack symbol in Γ

The output of δ is pair (p, γ)

p is a new state, γ is string of stack symbols that replaces X at the top of the stack.

if $\gamma = \epsilon$, then stack is popped

if $\gamma = X$, then stack is unchanged

if $\gamma = YZ$, then X is replaced by Z, and Y is pushed on to the stack.

q_0 : The start state

Z_0 : The start symbol

F : The set of accepting states or Final state.

Graphical Notation for PDA's

→ Transition Diagram for PDA's in which

(a) The nodes correspond to the states of the PDA.

(b) An arrow labeled 'start' indicates the start state, and doubly circled states are accepting, as for "Finite Automata"

(c) The arcs correspond to transitions of the PDA in the following sense. An arc labeled a, X/α from 'state q' to 'state p' means $\delta(q, a, X)$ contains the pair (p, α)

→ The start symbol of stack is

Instantaneous Descriptions of PDA

→ Configuration of PDA is represented by a triple (q, w, γ) (gamma)

$q \rightarrow$ state.
$w \rightarrow$ remaining input
$\gamma \rightarrow$ stack contents.

→ This triple is called "Instantaneous Description" or ID of Push Down Automata

(q, w, γ)

→ For suppose $\delta(q, a, X)$ contains (p, α).

Then for all strings w in Σ^* and β in Γ^*.

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

PDA Acceptance by Empty stack

→ For each PDA - $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$$N(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)\}$$ for any state q.

→ $N(S)$ is the set of inputs 'w' that 'P' can consume and at the same time empties the stack.

PDA Acceptance by Final state

→ Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA.

Then $L(P)$, the language accepted by P by 'final state' is

$$\{w \mid (q_0, w, Z_0) \vdash_P^* (q, t, \alpha)\}$$

for some state q in F and any stack string α.

→ starting in the initial state 'ID' with 'w' waiting on input, P consumes w from the input and enters an accepting state.

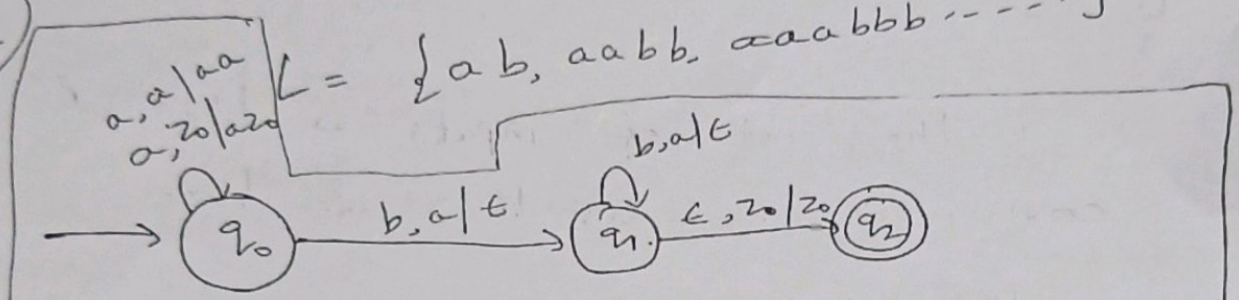
→ The contents of stack at that time is irrelevant.

PDA examples

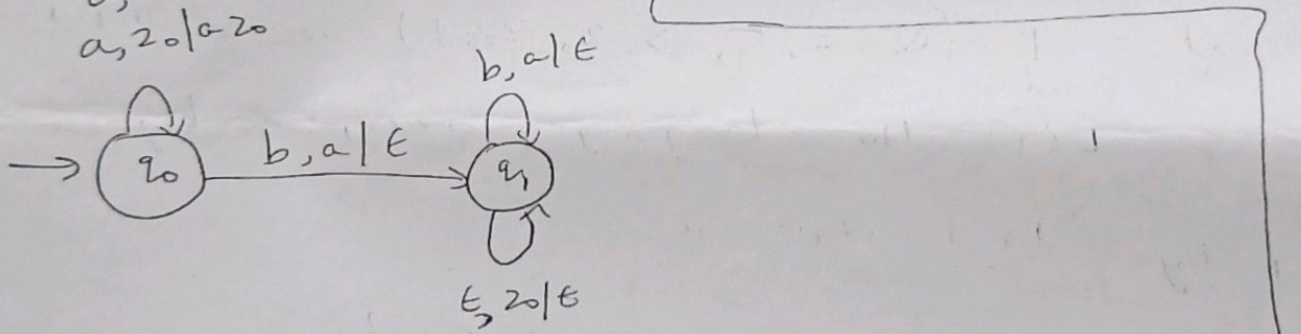
1) PDA accepts language $L = \{a^n b^n / n >= 1\}$

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

Sol) $L = \{ab, aabb, aaabbb, \ldots\}$

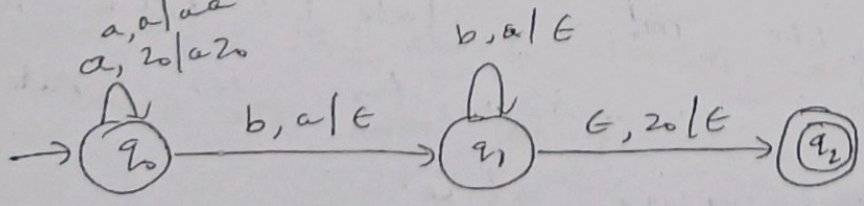


Acceptance by Final state



Acceptance by Empty stack

Acceptance by both Final state & Empty stack



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

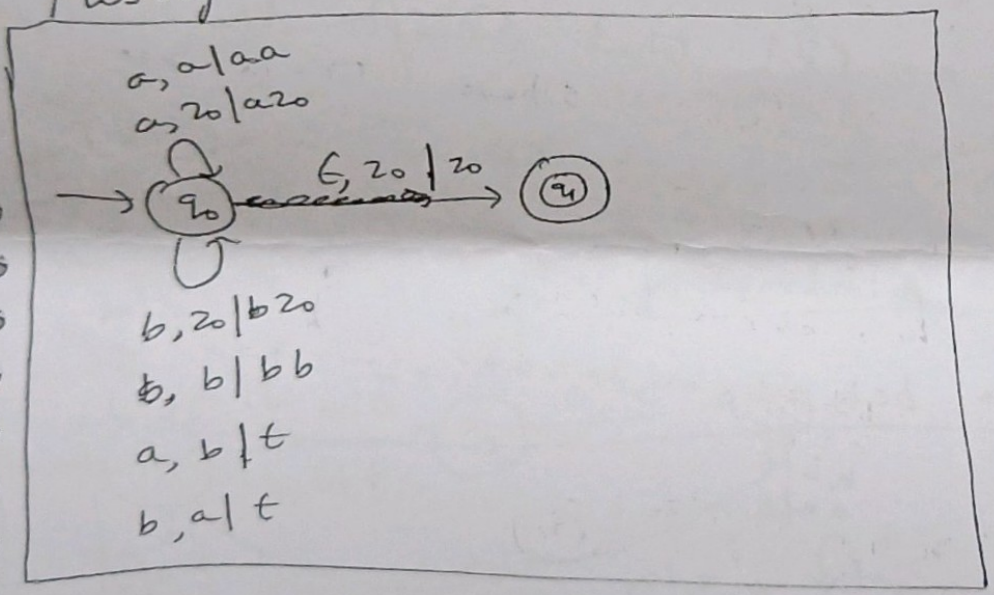
$\delta(q_0, a, a) = (q_0, aa)$ $\quad \delta(q_1, \epsilon, z_0) = (q_2, z_0)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

(2) PDA accepts Language where no. of a's are equal to no. of b's over $\Sigma = \{a, b\}$

Sol: $L = \{\epsilon, ab, ba, abab, baba, \ldots\}$

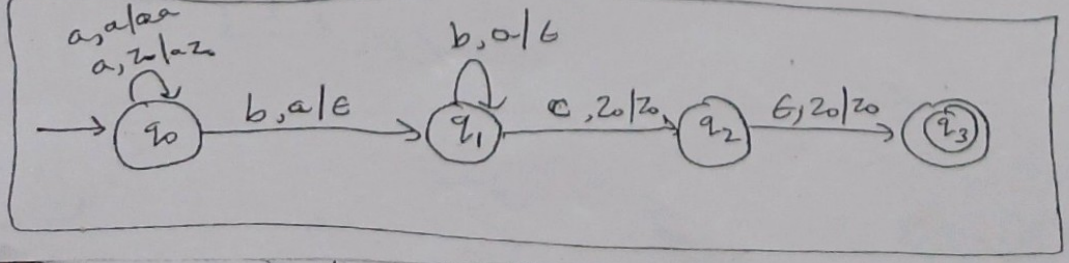
Mostly used Acceptance by Final State



(3) PDA accepts Language $L = \{a^n b^n c \mid n >= 1\}$

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

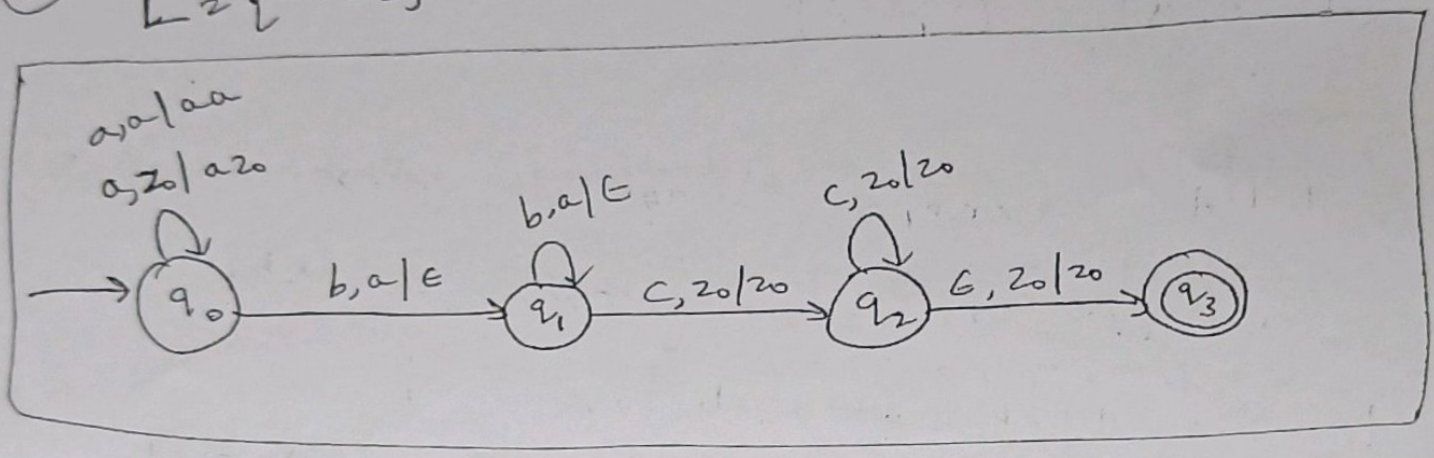
$L = \{abc, aabbc, aaabbbc, \ldots\}$



(4) Construct a PDA that accepts the Language

$L=\{a^n b^n c^m / m, n \geq 1\}$ where $\Sigma=\{a,b,c\}$

Sol) $L=\{abc, aabbc, aaabbbc, \ldots\}$

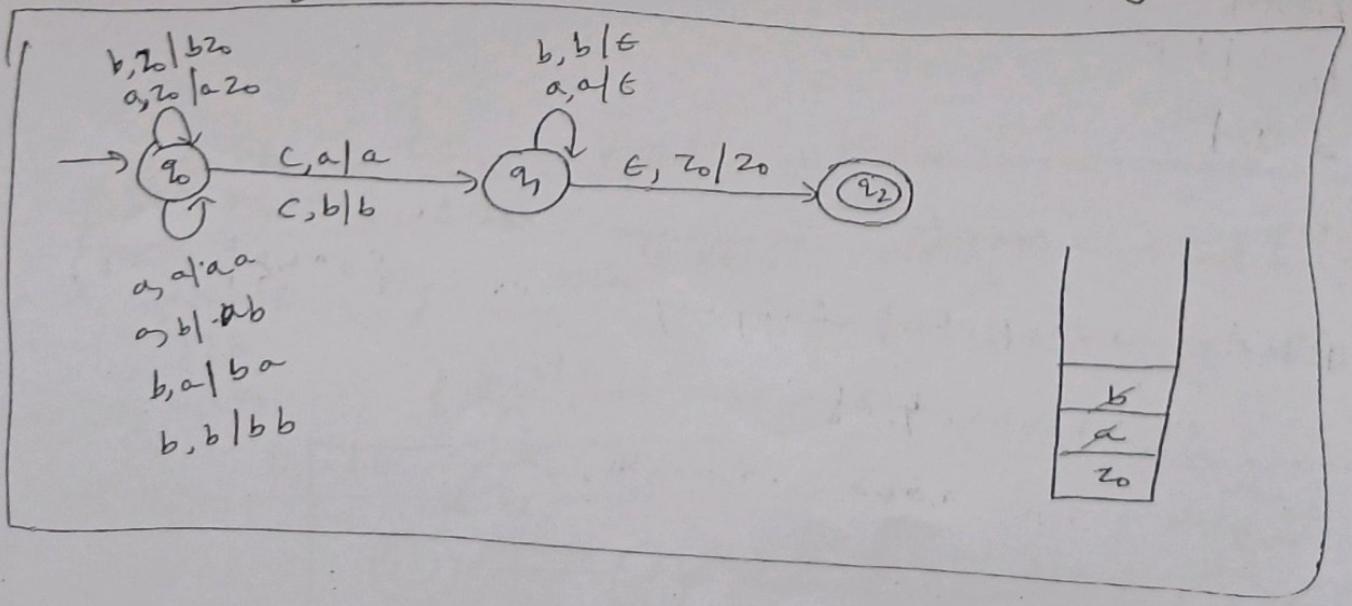


(5) Construct a PDA that accepts the Language

$L=\{wcw^R\}$ over $\Sigma=\{a,b,c\}$ where $w=(a+b)^*$

Odd length palindrome

$L=\{aca, bcb, abcba, bacab, \ldots\}$

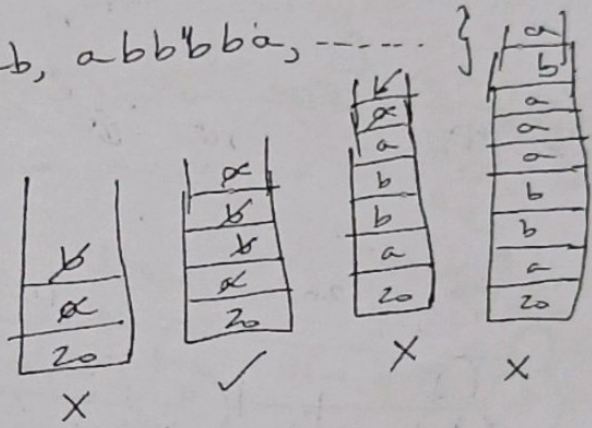


6) Construct a PDA that accepts the language $L = \{ww^R\}$ over $\Sigma = \{a, b\}$ where $w = (a+b)^*$

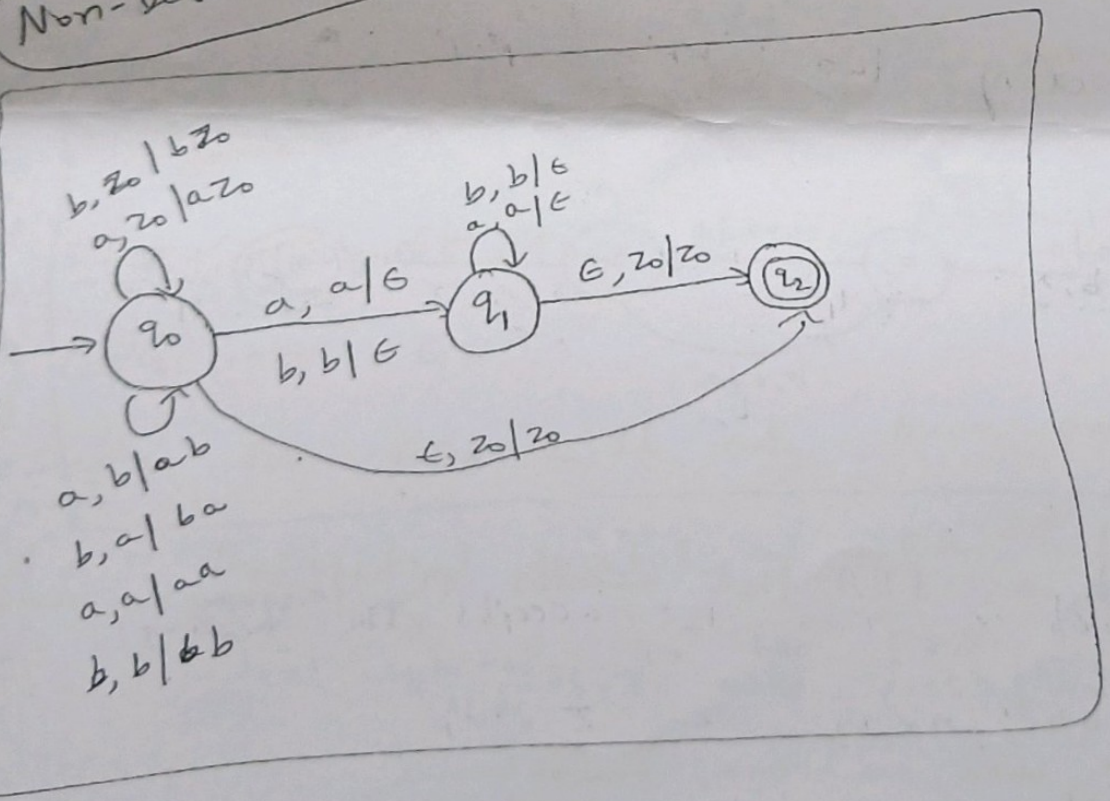
Even length palindrome

$L = \{aa, bb, abba, baab, abbbbba, \ldots\}$

abba|abba



Non-Deterministic PDA

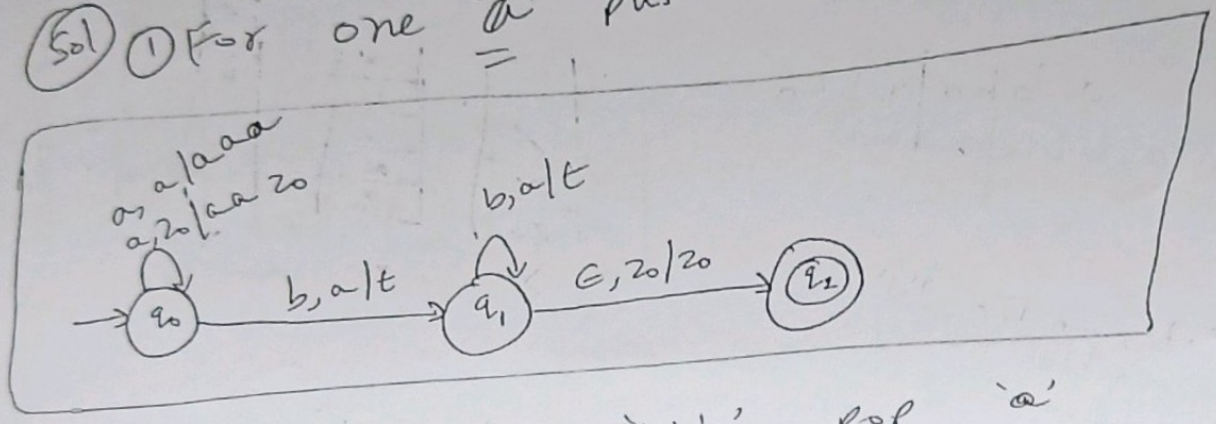


(7) Construct a PDA that accepts the Language

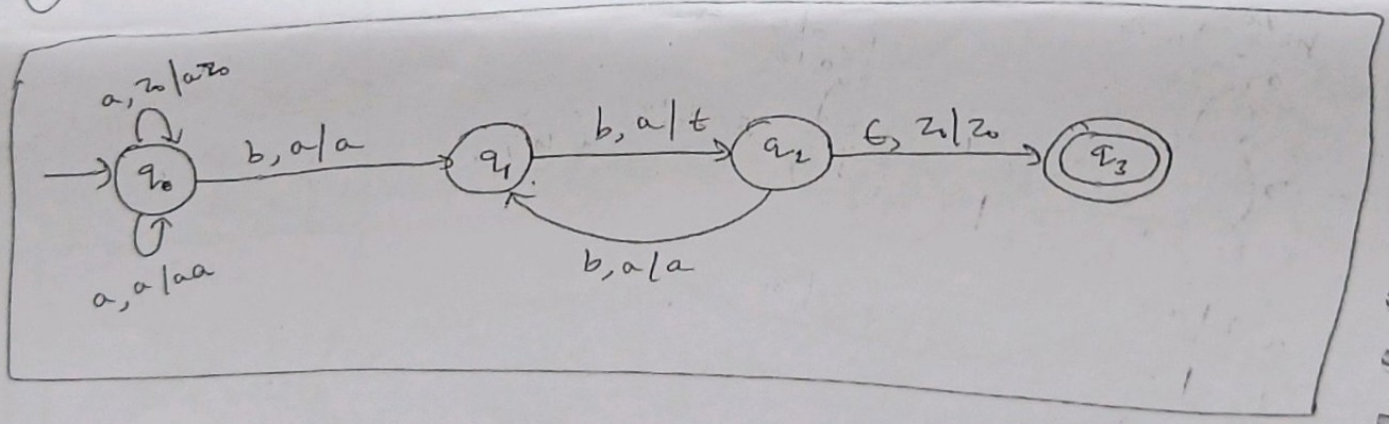
$L = \{a^n b^{2n} / n >= 1\}$ where $\Sigma = \{a, b\}$

$L = \{abb, aabbbb, aaabbbbbb, \ldots\}$

Sol (1) For one 'a' push two a's



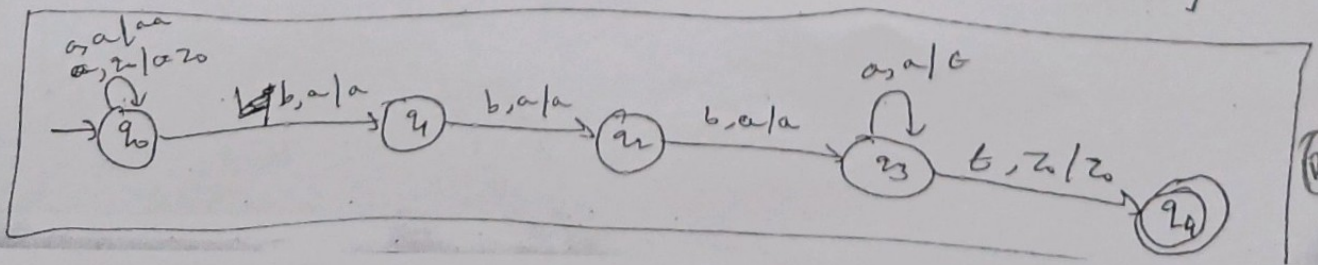
(2) For every two 'bb's' pop 'a'



(8) Construct a PDA that accepts the language

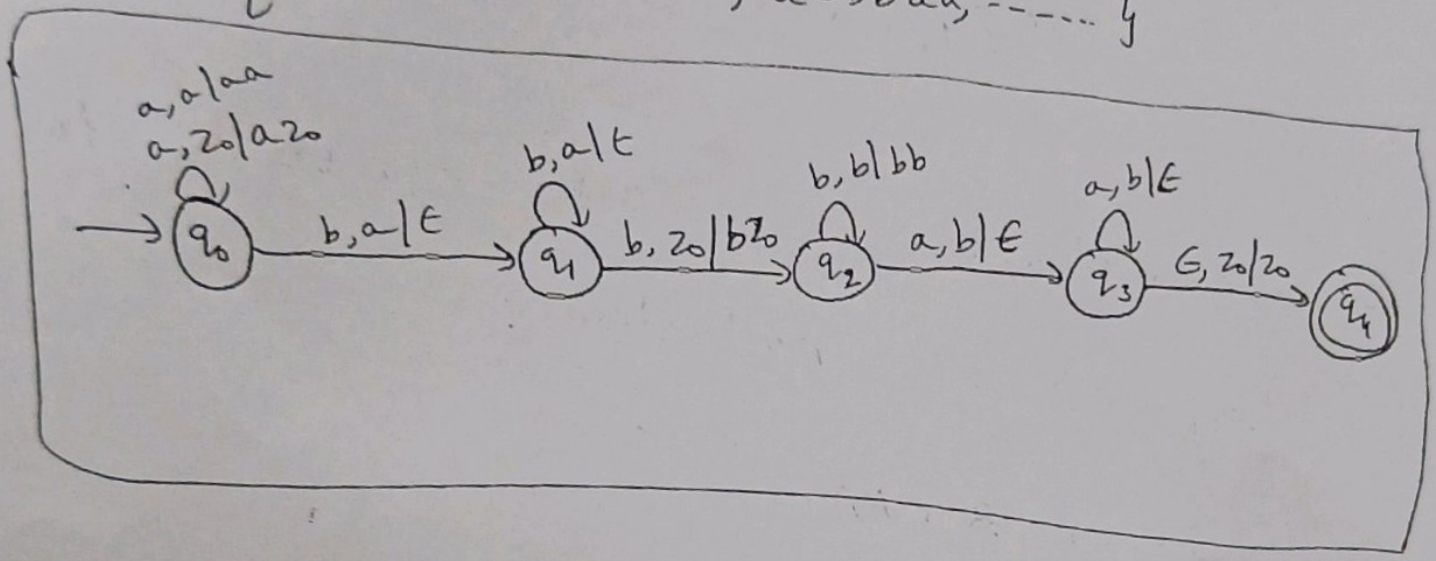
$L = \{a^n bbba^n / n >= 1\}$ where $\Sigma = \{a, b\}$

$L = \{abbba, aabbbaa, aaabbbaaa, \ldots\}$



(4) Construct a PDA that accepts the language $L = \{a^n b^{n+m} a^m / m, n >= 1\}$ where $\Sigma = \{a, b\}$

$L = \{abba, aabbba, abbbaa, \ldots\}$



Equivalence of CFG and PDA

Conversion of CFG to PDA

Step 1: Convert the given production of CFG into GNF

Step 2: The PDA will only have one state {q}

Step 3: The initial symbol of CFG will be (S) Initial symbol in the PDA.

step 4: For non-Terminal symbols, add the following rule: $\delta(q, \epsilon, A) = (q, a)$

where $A \rightarrow a$

Conversion of CFG to PDA

$A \to aA, \cdots$
$A \to a$

1) Convert the given production of CFG into GNF.
2) The PDA will only have one state $\{q\}$
3) The initial symbol of CFG will be initial symbol in the PDA.
4) For non-Terminal symbols, add the following rule:-

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

where $A \to a$

5) For each terminal symbols add the following rule:-

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

Conversion of PDA to CFG:-

→ The productions in P are induced by move of PDA as follows:-

1) 'S' productions are given by $S \to [q_0 z_0 q]$ for every $q \in Q$
2) For every popping move: $\delta(q, a, z) = (q', \epsilon)$ induces production

$$[q, z, q'] \to a.$$

3) For every pushing move :-

$\delta(q, a, z) = (q_1, z_1 \cdots z_m)$ induces many productions

$$[q, z, q_1^*] \to a\,[q_1 z_1 q_2]\,[q_2 z_2 q_3] \cdots [q_m z_m q_1]$$

Q) Generate CFG for given PDA

$M = \{\{q_0, q_1\}, \{0, 1\}, \{x, z_0\}, \delta, q_0, z_0, q_1\}$

① $\delta(q_0, 1, z_0) = (q_0, x z_0)$

② $\delta(q_0, 1, x) = (q_0, xx)$

③ $\delta(q_0, 0, x) = (q_0, x)$

④ $\delta(q_0, \epsilon, x) = (q_1, \epsilon)$

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

⑥ $\delta(q_1, 0, x) = (q_1, xx)$

⑦ $\delta(q_1, 0, z_0) = (q_1, \epsilon)$.

Step 1 :- $S \rightarrow [q_0 z_0 q_0] \mid [q_0 z_0 q_1]$

Step 2 :- Consider ① $\delta(q_0, 1, z_0) = (q_0, x z_0)$

$[q_0 z_0 q_0] = 1 [q_0, x, q_0] [q_0, z_0, q_0]$

$[q_0 z_0 q_0] = 1 [q_0, x, q_1] [q_1 z_0 q_0]$

$[q_0 z_0 q_1] = 1 [q_0, x, q_1] [q_1 z_0 q_1]$

$[q_0 z_0 q_1] = 1 [q_0, x q_0] [q_0 z_0 q_1]$

Step 3 :- Consider ② $\delta(q_0, 1, x) = (q_0, xx)$

$[q_0 x q_0] = 1 [q_0, x, q_0] [q_0 x q_0]$

$[q_0 x q_0] = 1 [q_0, x q_1] [q_1 x q_0]$

$[q_0 x q_1] = 1 [q_0 x q_0] [q_0 x q_1]$

$[q_0 x q_1] = 1 [q_0 x q_1] [q_1 x q_1]$

step (4) :- Consider (3)

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

$[q_0 x q_0] = 0[q_0 x q_0]$

$[q_0 x q_1] = 0[q_0 x q_1]$

step (5) :- Consider (4)

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

$[q_0 x q_1] \longrightarrow \epsilon$

step 6 :- Consider (5)

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

$[q_1 x q_1] \rightarrow \epsilon$

step 7 :- Consider (6)

$\delta(q_1, 0, x) = (q_1, xx)$

$[q_1 x q_0] \longrightarrow 0[q_1 x q_0][q_0 x q_0]$

$[q_1 x q_0] \longrightarrow 0[q_1 x q_1][q_1 x q_0]$

$[q_1 x q_1] \longrightarrow 0[q_1 x q_0][q_0 x q_1]$

$[q_1 x q_1] \longrightarrow 0[q_1 x q_1][q_1 x q_1]$

step 8 :- Consider (7)

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

$[q_1 z_0 q_1] \longrightarrow 0$

PDA to CFG Conversion :-

$M = \{\{q_0, q_1\}, \{a, b\}, \{z, z_0\}, \delta, q_0, z_0, \phi\}$ where δ is

① $\delta(q_0 b, z_0) = (q_0 z z_0)$

② $\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$

③ $\delta(q_0, b, z) = (q_0, zz)$

④ $\delta(q_0, a, z) = (q_1, z)$

⑤ $\delta(q_1, b, z) = (q_1, \varepsilon)$

⑥ $\delta(q_1, a, z_0) = (q_0, z_0)$

Sol :- $S \rightarrow [q_0 z_0 q_0] \mid [q_0 z_0 q_1]$

② $\Rightarrow \delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$.

$[q_0, z_0, q_0] \rightarrow \varepsilon$

⑤ $\Rightarrow$ ~~$\delta(q_0, a, z) = (q_1, z)$~~.

$[q_0 z q_1] \rightarrow a$

⑤ $\Rightarrow \delta(q_1, b, z) = (q_1, \varepsilon)$

$[q_1, z, q_1] \rightarrow b$.

① $\Rightarrow \delta(q_0, b, z_0) = (q_0, z z_0)$.

$[q_0, z_0, q_0] \rightarrow b\,[q_0 z q_0]\,[q_0 z_0 q_0]$

$[q_0, z_0, q_0] \rightarrow b\,[q_0 z q_1]\,[q_1 z_0 q_0]$

$[q_0, z_0, q_1] \rightarrow b\,[q_0 z q_0]\,[q_0 z_0 q_1]$

$[q_0, z_0, q_1] \rightarrow b\,[q_0 z q_1]\,[q_1 z_0 q_1]$

③ $\Rightarrow \delta(q_0, b, z) = (q_0, zz)$

$[q_0, b, q_0] \rightarrow b\,[q_0, z, q_0]\ [q_0, z, q_0]$

$[q_0, b, q_0] \rightarrow b\,[q_0, z, q_1]\ [q_1, z, q_0]$

$[q_0, b, q_1] \rightarrow b\,[q_0, z, q_0]\ [q_0, z, q_1]$

$[q_0, b, q_1] \rightarrow b\,[q_0, z, q_1]\ [q_1, z, q_1]$

④ $\Rightarrow \delta(q_0, a, z) = (q_1, z)$.

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

$[q_0, z, q_1] \rightarrow a\,[q_1\ z\ q_1]$.

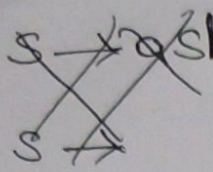
⑥ $\Rightarrow \delta(q_1, a, z_0) = (q_0, z_0)$.

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

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

Equivalence of PDA's & CFG's :-

Conversion of CFG to PDA :-



Eg :- $S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid bS \mid aBB$

$S \to aB \mid bA \Rightarrow \delta(q, \epsilon, S) = \{(q, aB), (q, bA)\}$

$A \to a \mid aS \mid bAA \Rightarrow \delta(q, \epsilon, A) = \{(q, a), (q, aS), (q, bAA)\}$

$B \to b \mid bS \mid aBB \Rightarrow \delta(q, \epsilon, B) = \{(q, b), (q, bS), (q, aBB)\}$

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

$\Rightarrow \delta(q, b, b) = (q, \epsilon)$.

String :- "aabbabba"

$\delta(q, aabbabba, S) \vdash \delta(q, aabbabba, aB)$

$\vdash \delta(q, abbabba, aBB)$

$\vdash \delta(q, bbabba, BB)$

$\vdash \delta(q, bbabba, bSB)$

$\vdash \delta(q, babba, bAB)$

$\vdash \delta(q, abba, aB)$

$\vdash \delta(q, bba, bS)$

$\vdash \delta(q, ba, bS)$

$\vdash \delta(q, ba, bA)$

$\vdash \delta(q, a, A)$

$\vdash \delta(q, \epsilon)$.

Accepted.

Eg :- $S \rightarrow aAA$

$A \rightarrow as \mid bs \mid a$

Sol :- Already grammar is in GNF

$S \rightarrow aAA \Rightarrow \delta(q_1, \varepsilon, S) = \{(q_1, aAA)\}$

$A \rightarrow as \mid bs \mid a \Rightarrow \delta(q_1, \varepsilon, A) = \{(q_1, as), (q_1, bs), (q_1, a)\}$

$\Rightarrow \delta(q_1, a, a) = (q_1, \varepsilon)$

$\Rightarrow \delta(q_1, b, b) = (q_1, \varepsilon)$

Step :- "abaaaaa"

$\delta(q_1, abaaaaa, S) \Rightarrow \delta(q_1, abaaaaa, aAA)$

$\vdash \delta(q_1, baaaaa, bSA)$

$\vdash \delta(q_1, aaaaa, aAAA)$

$\vdash \delta(q_1, aaaa, aAA)$

$\vdash \delta(q_1, aaa, aA)$

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

$\vdash \delta(q_1, a, \varepsilon)$

$\vdash \delta(q_1, \varepsilon)$ Accepted.

Deterministic PDA :-

→ A PDA is deterministic if there is never a choice of move in any situation. If $\delta(q, a, X)$ contains more than one pair then the PDA is nondeterministic because we can choose among these pairs when deciding the next move.

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ to be deterministic if & only if following conditions are met:-

→ $\delta(q, a, X)$ has at most one member for any q in Q, & X in Γ.

→ If $\delta(q, a, X)$ is nonempty for some a in Σ then $\delta(q, \epsilon, X)$ must be empty.

→ Even if $\delta(q, a, X)$ is always a singleton we could still have choices b/w read input & making a move with ϵ.

Eg :- $wcw^R \mid w \text{ in } (0+1)^*$