UNIT- IV

Push Down Automata (PDA)

PDA is useful in designing parsers or syntax analysers. A parser verifies the syntax of the text. Parsing is a part of the compilation process. So, knowledge of PDA is useful in Compiler Design.

Push Down Automata:

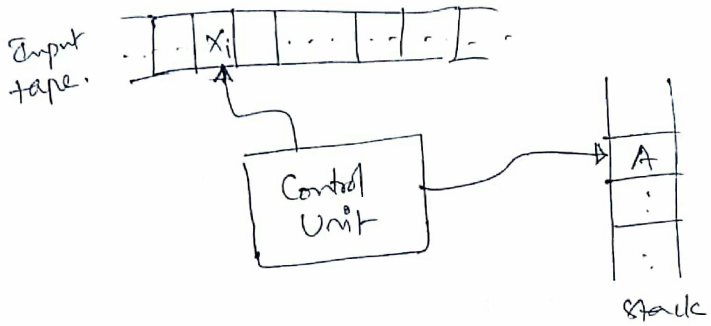


Fig: Conceptual model of Pushdown Automata.

PDA consists of three concepts components

(i) An input tape
(ii) finite control
(iii) Stack.

The input tape consists of a linear configuration of cells each of which contains a character from the input alphabet. This tape can be moved one cell at a time to the right.

The stack is also a sequential structure that has a first element and grows in either direction from the other end.

The control unit has some power which points the current symbol that is to be read. The head positioned over the current stack element can read and write special stack characters from that position. The current stack element is always the top element of stack.

The control unit contains both tape head and stack head finds itself at any moment in a particular state.

Def: PDA.

A finite state pushdown automaton is a 7-tuple

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

where

Q : a finite set of states alphabet

Σ : a finite set of input symbols

Γ : a finite set of stack alphabet

q_0 : the start state. $q_0 \in Q$

Z_0 : the start (initial) stack symbol $Z_0 \in \Gamma$

F : the set of final / Accepting states

δ : the transition function.

$$\delta(Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma) \longrightarrow Q \times \Gamma^*$$

PDA has 2 alphabets:

(i) An input alphabet Σ (for input string)

(ii) A stack alphabet Γ (stored on the stack)

A move on PDA may indicate:

(i) An element may be added to the stack $(q, a, Z_0) = (q, aZ_0)$

(ii) An element may be deleted from the stack

(iii) $(q, a, Z_0) = (q, \epsilon)$

(iv) There may or may not be change of state.

Ex:- $\delta(q, a, Z_0) = (q, aZ_0)$ indicates that in the state q on seeing a, a is pushed onto stack. There is no change of state.

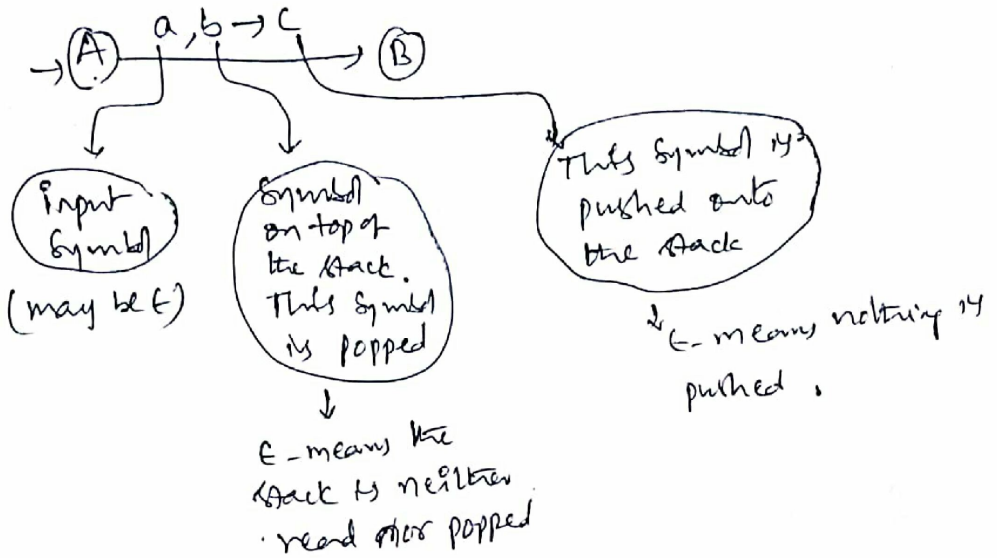
$\delta(q, a, Z_0) = (q, \epsilon)$ indicates that the state q on seeing an 'a', the current top symbol Z_0 is deleted from the stack.

$\delta(q, a, Z_0) = (q_1, aZ_0)$ indicates that a is pushed onto the stack and the state is changed to q_1.

Graphical Representation of Pushdown Automata.

PDA

$\delta(A, a, b) \rightarrow (B, c)$



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

$Q = \{P, q\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b\}$, $q_0 = q$, $F = \{P\}$.

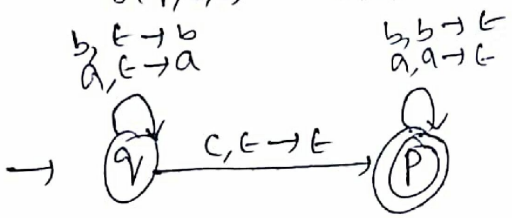
δ : $\delta(q, a, \epsilon) = (q, a)$

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

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

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

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



Pushdown Automata are two types

(i) Deterministic Pushdown Automata (DPDA)

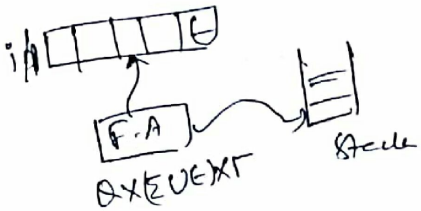
(ii) Non Deterministic Pushdown Automata (NDPDA)

DPDA:-

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

NDPDA:

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$$



Language Acceptance by PDA:

A language can be accepted by a PDA using two approaches.

(i) Acceptance by final state: The PDA accepts its input by consuming it and finally it enters the final state

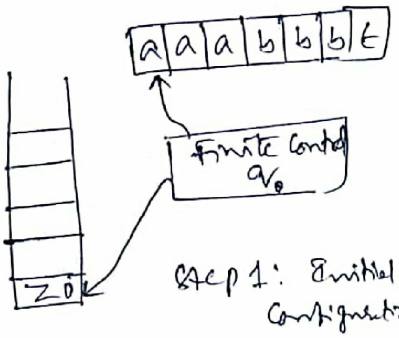
(ii) Acceptance by empty stack: on reading the input string from the initial configuration after some PDA, the stack of PDA becomes empty.

Design of Pushdown automata

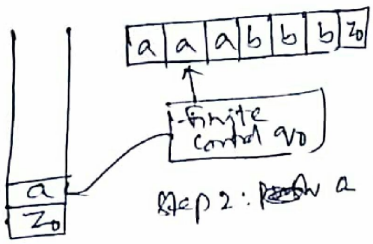
Ex:- Design PDA which accepts the language $L = \{a^n b^n \mid n \geq 1\}$.

Sol:
- Let q_0 be the initial state, q_f final state and Z_0 bottom of the stack.
- Read each 'a' push it onto the stack.
- Then read each b and pop out from stack for matching with a's.
- When all b's are read, if the stack is empty, then the string is valid. If any a's are left over on stack or b's on input tape, then the string is rejected.

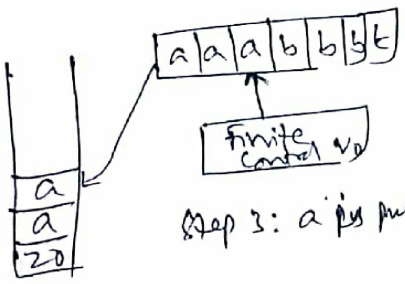
String: aaabbb.



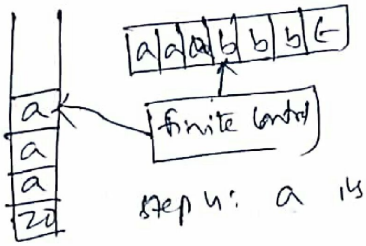
Step 1: Initial Configuration



Step 2: a is pushed.

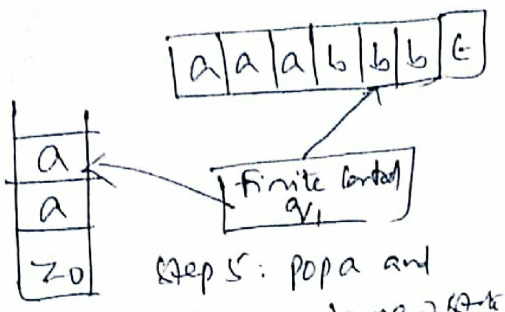


Step 3: a is pushed

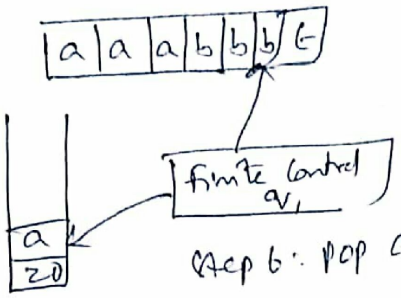


Step 4: a is pushed.

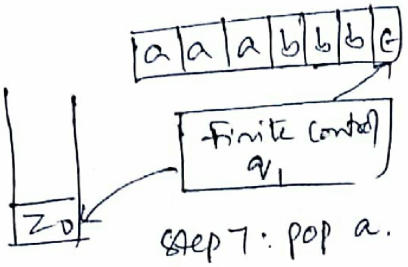
(4)



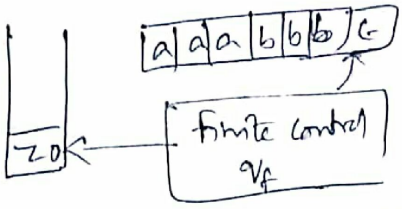
Step 5: pop a and change to new state



Step 6: pop a



Step 7: pop a.



Step 8: change to final state and halt

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ push a

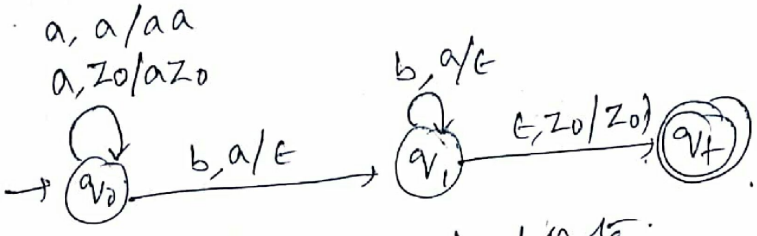
$\delta(q_0, a, a) = (q_0, aa)$ push a

$\delta(q_0, b, a) = (q_1, \epsilon)$ pop a and change the state

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

$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$ change to final state q_f and halt

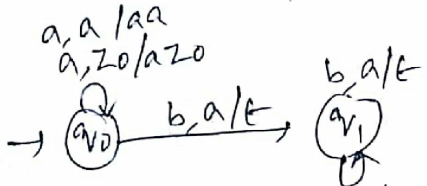
(q_1, ϵ)



Acceptance by final state.

Instantaneous description

$\delta(q_0, aaabbb\epsilon, Z_0)$
$= (q_0, aabbb\epsilon, aZ_0)$
$= (q_0, abbb\epsilon, aaZ_0)$
$= (q_0, bbb\epsilon, aaaZ_0)$
$= (q_1, bb\epsilon, aaZ_0)$
$= (q_1, b\epsilon, aZ_0)$
$= (q_1, \epsilon, Z_0)$
$= (q_f, Z_0)$
Accept

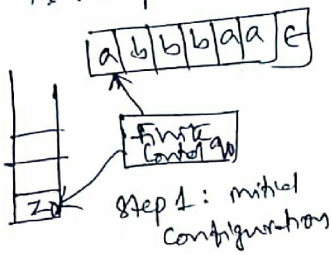


Acceptance by empty stack.

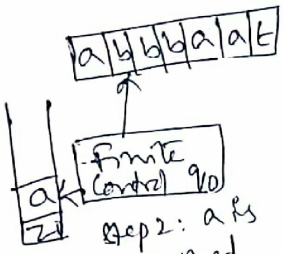
Design a PDA which accepts equal number of a and b's over $\Sigma = \{a, b\}$.

Sol:- The input can start with either a or b, and they can occur in any order, as bbaaba or ababba or babbaa etc. and so on, are all valid in the language.

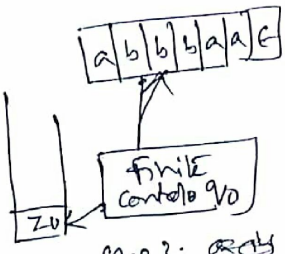
To design such a PDA, read the first symbol either a or b push it on the stack. Then read the next symbol. If the next symbol and top of the stack are same, push it onto the stack. Whenever top of stack and tape symbol are different, pop off the stack. When entire tape is read, if the stack becomes empty, then string is accepted.



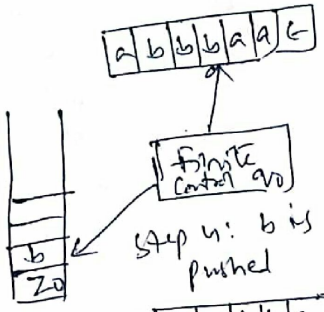
Step 1: initial configuration



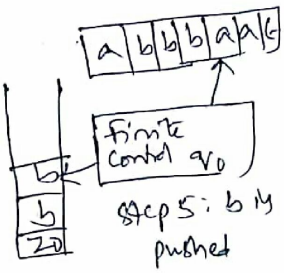
Step 2: a is pushed.



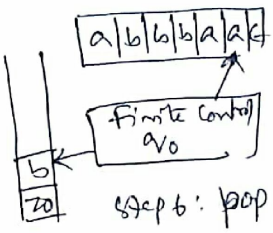
Step 3: pop a.



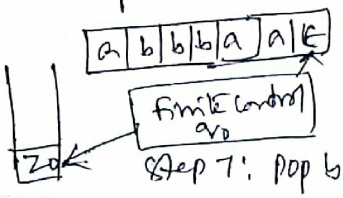
Step 4: b is pushed



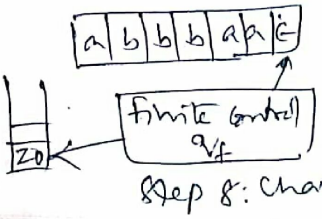
Step 5: b is pushed



Step 6: pop b.



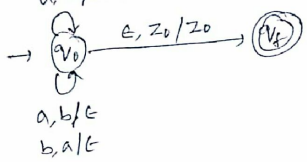
Step 7: pop b.



Step 8: Change to final state q_f and halt

Let q_0 be initial state, q_f be final state and Z_0 be the bottom of the stack ? (5)

b, b/bb
a, a/aa
b, Z_0/bZ_0
a, Z_0/aZ_0



a, b/ε
b, a/ε

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, b, Z_0) = (q_0, bZ_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, a, b) = (q_0, \epsilon)$
$\delta(q_0, b, a) = (q_0, \epsilon)$
$\delta(q_0, \epsilon, Z_0) = (q_f, Z_0)$

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

Instantaneous Description of PDA

During processing, the PDA moves from one configuration to another configuration. At any given instance, the configuration of PDA is expressed by the state of finite control, the content of stack and the input.

The configuration is expressed as a triple (q, x, γ) where

1. q is the state
2. x is the input string to be processed
3. γ is the content of the stack where the left most symbol corresponds to top of stack, and the rightmost is the bottom element.

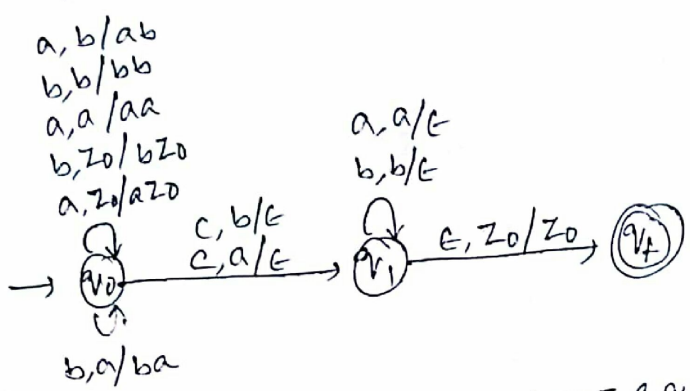
Instantaneous description for equal number of a's and b's over $\Sigma = \{a, b\}$

$(q_0, abbbaa\epsilon, Z_0) = (q_0, bbbaa\epsilon, aZ_0)$
$= (q_0, bbaa\epsilon, Z_0)$
$= (q_0, baa\epsilon, bZ_0)$
$= (q_0, aa\epsilon, bbZ_0)$
$= (q_0, a\epsilon, bZ_0)$
$= (q_0, \epsilon, Z_0) = (q_f, \epsilon, Z_0)$ = Accept

Design a PDA for $L = \{ wcw^r \mid w \in (a+b)^* \}$.

Sol:- Read the string w and push it onto the stack until it encounters c. After that, read each symbol, if it is matches the top of the stack, pop off the symbol. When input is read completely, if stack becomes empty, then the string is accepted.

q_0 be initial state, q_f be final state and Z_0 be the bottom of the stack.



The final PDA is given by $M = \{Q, \Sigma, \delta, q_0, F, Z_0, \Gamma\}$

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

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

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

δ: $\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, b, a) = (q_0, ba)$

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

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

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

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

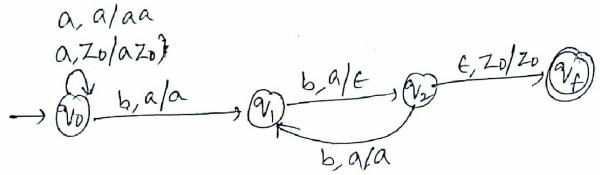
$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$

= Accept

Q.4) Design a PDA that accepts $L = \{a^n b^{2n} \mid n \geq 1\}$. (6)

Sol:- Here, we have to match each 'a' with two b's. So, Read the a's and push it on to the stack. After that, read one 'b' and change to different state. Then read second b. Now pop off from the stack and reset the state to initial state to continue the same process with each 'b'. When the input is read completely, if the stack becomes empty, then it is successful.

Let q_0 be initial state, q_f be final state and Z_0 become the bottom of the stack.



The final PDA $M = \{Q, \Sigma, \delta, q_0, F, Z_0, \Gamma\}$

$Q = \{q_0, q_1, q_2, q_f\}$ $\quad q_0 = \{q_0\}$

$\Sigma = \{a, b\}$ $\quad F = \{q_f\}$

$\Gamma = \{a, Z_0\}$ $\quad Z_0$ = the initial symbol in the stack.

δ: $\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

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

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

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

$\delta(q_2, \epsilon, Z_0) = (q_f, Z_0)$ Accept

Types of PDA's

There are two types of PDA's.

(1) Deterministic PDA (DPDA)

(2) Non Deterministic PDA (NDPDA)

Deterministic PDA (DPDA):

(1) A PDA that has at most one choice of move in any state is called a deterministic PDA.

- DPDA's are very useful in programming languages. For example, parsers used in Yet Another Compiler Compiler (YACC) are deterministic PDA's (DPDA).

Def: A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if and only if

(1) $\delta(q, a, X)$ has at most one move for $q \in Q$, $a \in \Sigma \cup \{\epsilon\}$ and $X \in \Gamma$ $\quad Q \times (\Sigma \cup \epsilon) \times \Gamma \to Q \times \Gamma^*$

(2) If $\delta(q, a, X)$ is not empty for some $a \in \Sigma$, then $\delta(q, \epsilon, X)$ must be empty.

DPDA is less powerful than NPDA.

Non Deterministic PDA's

- Non-deterministic PDA (NPDA) provides non-determinism in the moves defined.
- The PDA that has more than one choice of moves in any state is called non-deterministic PDA (NPDA).
- The context free languages could be recognized by NPDA's. $\quad Q \times (\Sigma \cup \epsilon) \times \Gamma \to 2^{(Q \times \Gamma^*)}$

(7)

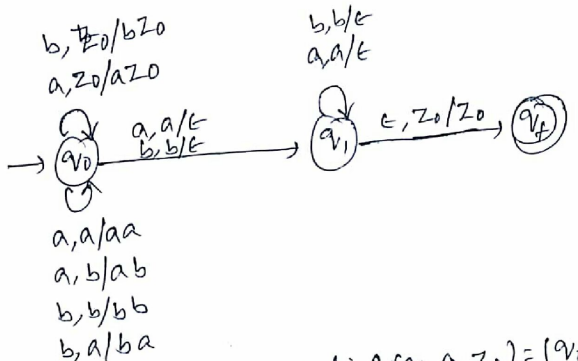
Design a NPDA which accepts $L = \{ ww^R \mid w \in (a+b)^* \}$

Sol:- Read the string w and push it on to the stack. After that, read each symbol. If it matches with top of the stack, pop off the symbol. When input is read completely, if stack becomes empty, it is successful.

The PDA can be given as follows.

Let q_0 be initial state, q_f be final state, Z_0 the bottom of the stack

abaaba



$M = \{Q, \Sigma, \delta, q_0, F, Z_0, \Gamma\}$ &

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

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

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

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, b, a) = (q_0, ba)$

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

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

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

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

$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$

UNIT-IV

Two-Stack PDA

Let us define a two-stack pushdown automata to be a six tuple.

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

Q = is a finite set of states

Σ = is an input alphabet

Γ = is a stack of symbols

$q_0 \in Q$ is the initial state.

$F \subseteq Q$ is the set of final states.

δ is transition relation, is a finite subset of

$$(Q \times \{\Sigma \cup \in\} \times \Gamma^* \times \Gamma^*) \Rightarrow (Q \times \Gamma^* \times \Gamma^*)$$

- Q: state
- $\{\Sigma \cup \in\}$: input symbol
- first Γ^*: first stack symbol
- second Γ^*: second stack symbol
- first Γ^*: operation on first stack
- second Γ^*: operation on second stack.

There are certain languages which cannot be accepted by push down automata as there is only one stack. This is specially, when we want to perform two checks on the input string. But if we use two stacks there, then these languages can be accepted by PDA.

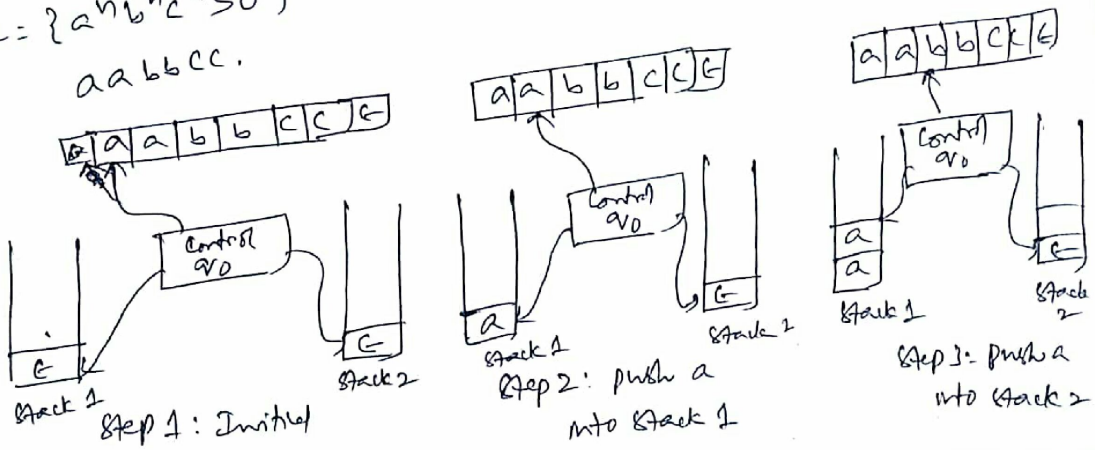
Design a two-stack PDA for languages $L=\{a^n b^n c^n \mid n>0\}$

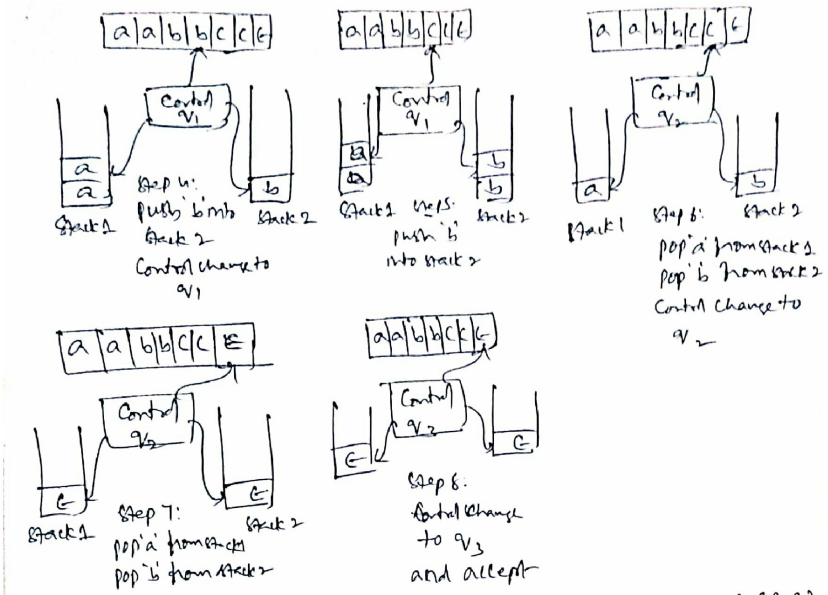
Sol: If we use one stack here then the language can not be accepted by PDA. Suppose with one stack we will push all a's onto the stack on reading them. then on reading b's we will pop each a. This helps in making a making a check on equal number of a's and b's. But now when we read out c's, the single stack is empty and we can't make out the equality of a's and b's with c's. Thus it is not possible for us to design a PDA with one stack which accepts language $L = a^n b^n c^n$.

But if we use two stacks, one for storing all a's and other for storing all the b's, then on reading each 'c' we can pop a each 'a' from first stack and each 'b' from second stack. This helps in identifying whether or not there are equal number of a's, b's and c's comming in order. This also means that PDA with two stack has more power than PDA with single stack.

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

aabbcc.





PDA M is defined by $M = \{\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \{a, b\}, \delta, q_0, q_3\}$

where δ is defined by

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

$\delta(q_0, a, a, \epsilon) = (q_0, aa, \epsilon)$

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

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

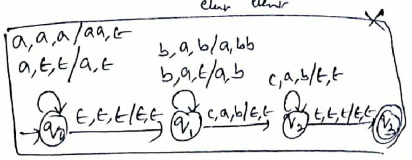
$\delta(q_1, b, a, b) = (q_1, a, bb)$

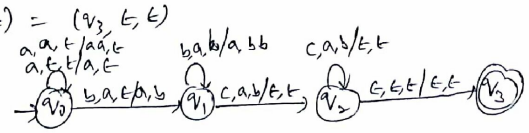
$\delta(q_1, c, a, b) = (q_2, \epsilon, \epsilon)$

$\delta(q_2, c, a, b) = (q_2, \epsilon, \epsilon)$

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

$\delta(q_0, a, \epsilon, \epsilon) = (q_0, a, \epsilon)$ — present control state, input symbol, first stack clear, second stack clear, control state, first stack operation, second stack operation





$\delta(q_0, aabbcc, \epsilon, \epsilon) \vdash \delta(q_0, abbcc, a, \epsilon)$

$\vdash \delta(q_0, bbcc, aa, \epsilon)$

$\vdash \delta(q_1, bcc, aa, b)$

$\vdash \delta(q_1, cc, aa, bb)$

$\vdash \delta(q_2, c, a, b)$

$\vdash \delta(q_2, c, \epsilon, \epsilon)$

$\vdash \delta(q_3, \epsilon, \epsilon)$ Accept

Pushdown Automata to Context Free Grammar (1)

CFG to PDA

Theorem: If L is a CFL, then we can construct a PDA 'P' accepting L by an empty stack.

$L = N(P)$.

Method: Let $L = L(G)$, where $G = (V, T, P, S)$ is a Context Free Grammar in Greibach Normal Form (GNF).

We construct PDA P as

$P = (\{q\}, \Sigma, (V \cup \Sigma), \delta, q, S, \emptyset)$ [$\emptyset$ is set of final state. S: top of stack element]

where δ is defined by the following rules

$R_1 = \delta(q, \epsilon, A) = \{(q, \alpha) \mid A \to \alpha \text{ is in } P\}$

$R_2 = \delta(q, a, a) = \{(q, \epsilon)\}$ for every a in Σ.

Ex:- Construct a PDA P which is equivalent to the following CFG

$S \to OCC, C \to OS, C \to 1S, C \to 0$.

Test whether 0104 is accepted by this PDA.

Sol:- Let PDA $P = (\{q\}, (0,1), \{S, C, 0, 1\}, \delta, q, S, \emptyset)$

The production rules δ is can be given as

R_3: $\delta(q, \epsilon, S) = (q, OCC)$ by using Rule R_1

R_4: $\delta(q, \epsilon, C) = \{(q, OS), (q, 1S), (q, 0)\}$ by using Rule R_1

$R_5 : \delta(q, 0, 0) = (q, \epsilon)$ by using Rule R_2

$R_6 : \delta(q, 1, 1) = (q, \epsilon)$ by using Rule R_2.

the string 01^04

$S \Rightarrow 0CC \Rightarrow 01SC \Rightarrow 010CCC \Rightarrow 0100CC \Rightarrow 01000C$
$\Rightarrow 010000.$

$\delta(q, 010000, S) \vdash \delta(q, 010000, 0CC)$: R_3
$\vdash \delta(q, 10000, CC)$: R_5
$\vdash \delta(q, 10000, 1SC)$: R_4
$\vdash \delta(q, 0000, SC)$: R_6
$\vdash \delta(q, 0000, 0CCC)$: R_3
$\vdash \delta(q, 000, CCC)$: R_5
$\vdash \delta(q, 000, 0CC)$: R_4
$\vdash \delta(q, 00, CC)$: R_5
$\vdash \delta(q, 00, 0C)$: R_4
$\vdash \delta(q, 0, C)$: R_5
$\vdash \delta(q, 0, 0)$: R_4
$\vdash \delta(q, \epsilon)$ R_4
Accepted.

Thus, 01^04 is accepted by the PDA.

Convert the grammar $S \to 0AA$, $A \to 0S \mid 1S \mid 0$ to PDA that accepts the same language by empty stack.

Sol: PDA $P = (\{q\}, \Sigma, (V \cup \Sigma), \delta, q, S, \phi)$

δ: $R_1 = \delta(q, \epsilon, A) = (q, \alpha) \mid A \to \alpha$ is in P

$R_2 = \delta(q, a, a) = (q, \epsilon)$ for every a in Σ.

The given productions are

$S \to 0AA$

$A \to 0S \mid 1S \mid 0$

PDA $P = (\{q\}, \{0,1\}, \{S, A, 0, 1\}, \delta, q, S, \phi)$

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

$\delta(q, \epsilon, A) = \{(q, 0S), (q, 1S), (q, 0)\}$

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

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

String 000

$S \Rightarrow 0AA \Rightarrow 00A \Rightarrow 000$

$\delta(q, 000, S) \vdash \delta(q, 000, 0AA)$

$\vdash \delta(q, 00, AA)$

$\vdash \delta(q, 00, 0A)$

$\vdash \delta(q, 0, A)$

$\vdash \delta(q, 0, 0)$

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

Ex:- Obtain the PDA for the following grammar

$S \to AS|\epsilon$

$A \to 0A1|A1|01.$

Sol:- In the given grammar, there is ϵ-productions.

So, remove ϵ-productions.

then $S \to AS|A$

$A \to 0A1|A1|01.$

The production $S \to A$ is a Unit production. So, remove Unit production. then

$S \to AS|0A1|A1|01$

$A \to 0A1|A1|01.$

But the productions are not in CNF, then

$S \to AS|0B|A1|01$

$A \to 0B|A1|01$

$B \to A1.$

The grammar is in CNF. But the grammar is not in GNF. Convert that it into GNF.

. $S = A_1$, $A = A_2$, $B = A_3$, then

$A_1 \to A_2A_1|0A_3|A_21|01$

$A_2 \to 0A_3|A_21|01$

$A_3 \to A_21.$

Here, $A_2 \to A_2 1 | 0A_3 | 01$ is in left recursion; ③

So, remove it, then

$$Z \to 1 | 1Z$$
$$A_2 \to 0A_3 | 01 | 0A_3Z | 01Z$$

then

$$A_1 \to 0A_3A_1 | 01A_1 | 0A_3ZA_1 | 01ZA_1 | 0A_3 | 0A_3Z1 | 011 | 0A_3Z1 | 01Z1 | 01$$

$$A_2 \to 0A_3 | 01 | 0A_3Z | 01Z$$

$$A_3 \to 0A_31 | 011 | 0A_3Z1 | 01Z1$$

$$Z \to 1 | 1Z.$$

The above productions are in the form of GNF.

then: PDA $= \{\{q\}, \{0,1\}, \{A_1, A_2, A_3, Z\}, \delta, q, A_1, \phi\}$

$\delta(q, \epsilon, A_1) = \{(q, 0A_3A_1), (q, 01A_1), (q, 0A_3ZA_1), (q, 01ZA_1), (q, 0A_3), (q, 0A_3Z1), (q, 011), (q, 0A_3Z1), (q, 01Z1), (q, 01)\}$

$\delta(q, \epsilon, A_2) = \{(q, 0A_3), (q, 01), (q, 0A_3Z), (q, 01Z)\}$

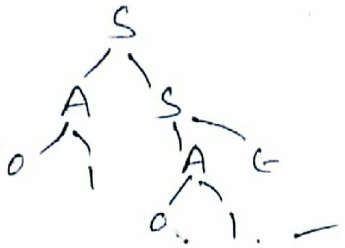
$\delta(q, \epsilon, A_3) = \{(q, 0A_31), (q, 011), (q, 0A_3Z1), (q, 01Z1)\}$

$\delta(q, \epsilon, Z) = \{(q, 1), (q, 1Z)\}$

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

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

String = 0101



$\delta(q, 0101, A_1) \vdash \delta(q, 0101, 01A_1)$

$\vdash \delta(q, 101, 1A_1)$

$\vdash \delta(q, 01, A_1)$

$\vdash \delta(q, 01, 01)$

$\vdash \delta(q, 1, 1)$

$\vdash \delta(q, \epsilon)$ = Accept

Ex:- Convert the following Grammar G to PDA that accepts the same language by empty stack.

$S \to 0S1 \mid A$

$A \to A0 \mid S \mid \epsilon$.

Sol:- In the given grammar, ϵ-productions are there. So, remove ϵ-productions. Then

$S \to 0S1 \mid A$

$A \to A0 \mid S \mid 0$

But, the grammar contains, Unit productions $S \to A$ and $A \to S$. So, remove it. Then the productions are

$S \to 0S1 \mid A0 \mid S \mid 0$

~~$A \to 0S1$~~

$A \to A0 \mid 0S1 \mid A \mid 0$

then $S \to 0S1 \mid A0 \mid 0$

$A \to A0 \mid 0S1 \mid 0$.

But, the grammar productions are not in CNF.
So, Convert it into CNF.

$S \to 0B \mid A0 \mid 0$
$A \to A0 \mid 0B \mid 0$
$B \to S1$

the grammar is in CNF. But the grammar productions are not in GNF. Then, Convert it into GNF by following way.

$S = A_1$, $B = A_2$, $A = A_3$,

then $A_1 \to 0A_2 \mid A_3 0 \mid 0$
$A_3 \to A_3 0 \mid 0A_2 \mid 0$
$A_2 \to A_1 1$

there, the production $A_3 \to A_3 0$ in left recursion, then remove left recursion.

$A_3 \to A_3 0 \mid 0A_2 \mid 0$

$\Rightarrow Z \to 0 \mid 0Z$
$A_3 \to 0A_2 \mid 0 \mid 0A_2 Z \mid 0Z.$

Then the productions are

$A_1 \to 0A_2 \mid A_3 0 \mid 0$
$A_3 \to 0A_2 \mid 0 \mid 0A_2 Z \mid 0Z$
$A_2 \to A_1 1$
$Z \to 0 \mid 0Z$

A_3 production is in GNF. then substitute, this in A_1.

Then $A_1 \to 0A_2 \mid 0A_20 \mid 00 \mid 0A_2Z0 \mid 0Z0 \mid 0$

$A_3 \to 0A_2 \mid 0 \mid 0A_2Z \mid 0Z$

$A_2 \to 0A_21 \mid 0A_201 \mid 001 \mid 0A_2Z01 \mid 0Z01 \mid 01$

$Z \to 0 \mid 0Z.$

The grammar is in GNF.

Then, the equivalent PDA $P = \{Q, \Sigma, \delta, q_0, f, Z, \Gamma\}$ are

$P : (\{q\}, \{0,1\}, \{A_1, A_2, A_3, Z, 0, 1\}, \delta, q, A_1, \emptyset)$

$\delta(q, \in, A_1) = \{(q, 0A_2), (q, 0A_20), (q, 00), (q, 0A_2Z0), (q, 0Z0), (q, 0)\}$

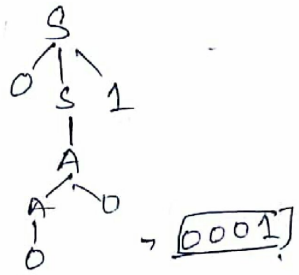
$\delta(q, \in, A_3) = \{(q, 0A_2), (q, 0), (q, 0A_2Z), (q, 0Z)\}$

$\delta(q, \in, A_2) = \{(q, 0A_21), (q, 0A_201), (q, 001), (q, 0A_2Z01), (q, 0Z01), (q, 01)\}$

$\delta(q, \in, Z) = \{(q, 0), (q, 0Z)\}$

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

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



Ques: 0001

$\delta(q, 0001, A_1) \vdash \delta(q, 0001, 0A_2)$
$\vdash \delta(q, 001, A_2)$
$\vdash \delta(q, 001, 001)$
$\vdash \delta(q, 01, 01)$
$\vdash \delta(q, 1, 1)$
$\vdash \delta(q, \in)$ ⇒ Accept

Constructing CFG for the given PDA:

Given PDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset\}$, we will construct a grammar G such that $L(G) = L(M)$.

To convert the PDA to CFG, use the following three rules.

R_1: The productions for start symbol S are given by

$S \rightarrow [q_0, Z_0, q]$ for each state q in Q.

R_2: Each move that pops a symbol from stack with transition as

$$\delta(q, a, Z_i) = (q_1, \epsilon)$$

induces a production as

$$[q, Z_i, q_1] \rightarrow a \text{ for } q_1 \text{ in } Q.$$

R_3: Each move that does not pop symbol from stack with transition as

$$\delta(q, a, Z_0) = (q_1, Z_1 Z_2 Z_3 Z_4 \cdots)$$

induces a production as

$$[q, Z_0, q_m] \rightarrow a[q_1, Z_1, q_2][q_2, Z_2, q_3][q_3, Z_3 q_4] \cdots [q_{m-1}, Z_m, q_m]$$

for each q_i in Q, where $1 \leq i \leq m$.

After defining all the rules, apply simplifications on grammar to get reduced grammar.

Ex:- Give the equivalent CFG for the following PDA (3)

$M = \{\{q_0, q_1\}, \{a, b\}, \{Z, Z_0\}, \delta, q_0, Z_0, \phi\}$ where δ is given by

$\delta(q_0, b, Z_0) = (q_0, ZZ_0)$

$\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$

$\delta(q_0, b, Z) = (q_0, ZZ)$

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

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

$\delta(q_1, a, Z_0) = (q_0, Z_0)$

Sol:- The states are q_0 and q_1, and stack symbols are Z and Z_0.

The states are $\{S, [q_0, Z_0, q_0], [q_0, Z_0, q_1], [q_0, Z_0, q_0], [q_1, Z_0, q_1], [q_0, Z, q_0], [q_0, Z, q_1], [q_1, Z, q_0], [q_1, Z, q_1]\}$

Start productions are given by rule 1.

ie $S \to [q_0, Z_0, q_0] \mid [q_0, Z_0, q_1]$

(i) The context free grammar for $\delta(q_0, b, Z_0) = (q_0, ZZ_0)$ is obtained by rule (3)

$[q_0, Z_0, q_0] \to b[q_0, Z_0, q_0][q_0, Z_0, q_0]$

$[q_0, Z_0, q_0] \to b[q_0, Z, q_1][q_1, Z_0, q_0]$

$[q_0, Z_0, q_1] \to b[q_0, Z, q_0][q_0, Z_0, q_1]$

$[q_0, Z_0, q_1] \to b[q_0, Z, q_1][q_1, Z_0, q_1]$

(ii) The context free grammar for $\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$ is obtained by rule (2)

$[q_0, Z_0, q_1] \to \epsilon$

(3) The CFG for $\delta(q_0, b, Z) = (q_0, ZZ)$ is obtained by rule (1)

$[q_0, Z, q_1] \to b[q_0, Z, q_0][q_0, Z, q_0]$
$[q_0, Z, q_0] \to b[q_0, Z, q_1][q_1, Z, q_0]$
$[q_0, Z, q_1] \to b[q_0, Z, q_0][q_0, Z, q_1]$
$[q_0, Z, q_1] \to b[q_0, Z, q_1][q_1, Z, q_1]$

(4) The CFG for $\delta(q_0, a, Z) = (q_0, Z)$ is obtained by rule (3)

$[q_0, Z, q_0] \to a[q_0, Z, q_0]$ ×
$[q_0, Z, q_1] \to a[q_1, Z, q_1]$

(5) The CFG for $\delta(q_1, b, Z) = (q_1, \epsilon)$ is obtained by rule (2)

$[q_1, Z, q_1] \to b$

(6) The CFG for $\delta(q_1, a, Z_0) = (q_0, Z_0)$ is obtained by rule (3)

$[q_1, Z_0, q_0] \to a[q_0, Z_0, q_0]$
$[q_1, Z_0, q_0] \to a[q_0, Z_0, q_1]$ ✓

Simplifying grammar: In the above grammar, first identify the non-terminals that are not defined and eliminate the productions that refer to these productions. Similarly, use the process procedure of eliminating useless symbols and useless productions. Hence the complete grammar is as follows.

$S \to [q_0, Z_0, q_0]$
$[q_0, Z_0, q_0] \to b[q_0, Z, q_1][q_1, Z_0, q_0]$
$[q_0, Z_0, q_0] \to \epsilon$
$[q_0, Z, q_1] \to b[q_0, Z, q_1][q_1, Z, q_1]$
$[q_0, Z, q_1] \to a[q_1, Z, q_1]$
$[q_1, Z, q_1] \to b$
$[q_1, Z_0, q_0] \to a[q_0, Z_0, q_0]$ =

Construct PDA to accept the [illegible] in the CFG.

The PDA M as given below $A = (\{q_0, q_1\}, \{0,1\}, \{X, A\}, \delta, q_0, S, \phi)$

where δ is as given below

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

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

$\delta(q_0, 1, \cdot$

Let $M = (\{q_0, q_1\}, \{0,1\}, \{X, Z_0\}, \delta, q_0, Z_0, \phi)$ where δ is given by

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$

$\delta(q_0, 0, X) = \{(q_0, XX)\}$

$\delta(q_0, 1, X) = \{(q_1, \epsilon)\}$

$\delta(q_1, 1, X) = \{(q_1, \epsilon)\}$

$\delta(q_1, \epsilon, X) = \{(q_1, \epsilon)\}$

$\delta(q_1, \epsilon, Z_0) = \{(q_1, \epsilon)\}$

Construct CFG $G = (V, T, P, S)$ generating $N(M)$.

Sol: We have to construct CFG $G = (V, T, P, S)$ for the given PDA. The V represents set of non-terminals such that

$V = \{S, [q_0, X, q_0], [q_0, X, q_1], [q_1, X, q_0], [q_1, X, q_1], [q_0, Z_0, q_0], [q_0, Z_0, q_1], [q_1, Z_0, q_0], [q_1, Z_0, q_1]\}$

The T represents set of terminals Here.

$T = \{0, 1\}$

The production rules can be constructed from the δ function.

For the start state production S, we can write as

$S \rightarrow [q_0, Z_0, q_0]$

$S \rightarrow [q_0, Z_0, q_1]$

Now can write the productions for $[q_0, Z_0, q_0]$ and $[q_0, Z_0, q_1]$.

$\delta(q_0, 0, Z_0) = (q_0, XZ_0)$ -

$[q_0, Z_0, q_0] \to 0\,[q_0, X, q_0][q_0, Z_0, q_0]$

$[q_0, Z_0, q_0] \to 0\,[q_0, X, q_1][q_1, Z_0, q_0]$

$[q_0, Z_0, q_1] \to 0\,[q_0, X, q_0][q_0, Z_0, q_1]$

$[q_0, Z_0, q_1] \to 0\,[q_0, X, q_1][q_1, Z_0, q_1]$

for $\delta(q_0, 0, X) = (q_0, XX)$

$[q_0, X, q_0] \to 0\,[q_0, X, q_0][q_0, X, q_0]$

$[q_0, X, q_0] \to 0\,[q_0, X, q_1][q_1, X, q_0]$

$[q_0, X, q_1] \to 0\,[q_0, X, q_0][q_0, X, q_1]$

$[q_0, X, q_1] \to 0\,[q_0, X, q_1][q_1, X, q_1]$

for the rule $\delta(q_0, 1, X) = (q_1, \epsilon)$, we get

$[q_0, X, q_1] \to 1$

For the rule $\delta(q_1, 1, X) = (q_1, \epsilon)$, we get

$[q_1, X, q_1] \to 1$

For the rule $\delta(q_1, \epsilon, X) = (q_1, \epsilon)$, we get

$[q_1, X, q_1] \to \epsilon$

For the rule $\delta(q_1, \epsilon, Z_0) = (q_1, \epsilon)$, we get

$[q_1, Z_0, q_1] \to \epsilon$.

Ex: The PDA is as given below

$$A = (\{q_0, q_1\}, \{0,1\}, \{S, A\}, \delta, q_0, S, \emptyset)$$

where δ is as given below

$\delta(q_0, 1, S) = \{(q_0, AS)\}$
$\delta(q_0, \epsilon, S) = \{(q_0, \epsilon)\}$
$\delta(q_0, 1, A) = \{(q_0, AA)\}$
$\delta(q_0, 0, A) = \{(q_1, A)\}$
$\delta(q_1, 1, A) = \{(q_1, \epsilon)\}$
$\delta(q_1, 0, S) = \{(q_0, S)\}$. Construct the CFG equivalent to this PDA

Sol:- CFG $G = (V, T, P, S)$

$T = \{0, 1\}$, $V = \{S, [q_0, A, q_0], [q_0, A, q_1], [q_0, S, q_0], [q_0, S, q_1], [q_1, A, q_0], [q_1, A, q_0], [q_1, S, q_0], [q_1, S, q_1]\}$

Production rules

Using rule 1.

$P_1 : S \to [q_0, S, q_0]$
$P_2 : S \to [q_0, S, q_1]$

For $\delta(q_0, 1, S) = \{(q_0, AS)\}$, we get [using rule 3]

$P_3 : [q_0, S, q_0] \Rightarrow 1\,[q_0, A, q_0]\,[q_0, S, q_0]$
$P_4 : [q_0, S, q_0] \to 1\,[q_0, A, q_1]\,[q_1, S, q_0]$
$P_5 : [q_0, S, q_1] \to 1\,[q_0, A, q_0]\,[q_0, S, q_1]$
$P_6 : [q_0, S, q_1] \to 1\,[q_0, A, q_1]\,[q_1, S, q_1]$

For $\delta(q_0, \epsilon, S) = \{(q_0, \epsilon)\}$ using rule 2

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

For $\delta(q_0, 1, A) = \{(q_0, AA)\}$, we get [using rule 3]

$P_8 : [q_0, A, q_0] \Rightarrow 1\,[q_0, A, q_0]\,[q_0, A, q_0]$
$P_9 : [q_0, A, q_0] \to 1\,[q_0, A, q_1]\,[q_1, A, q_0]$
$P_{10} : [q_0, A, q_1] \to 1\,[q_0, A, q_0]\,[q_0, A, q_1]$
$P_{11} : [q_0, A, q_1] \to 1\,[q_0, A, q_1]\,[q_1, A, q_1]$

for $\delta(q_0, 0, A) = \{(q_1, A)\}$ we get [using rule 3]

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

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

for $\delta(q_1, 1, A) = \{(q_1, \epsilon)\}$, we get [using rule 2]

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

for $\delta(q_1, 0, S) = \{(q_0, S)\}$ gives

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

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