

Unit - 3

Context-free grammars :- Definition of context free
Derivations using a Grammar, left most & Rightmost

The language of Grammar, sentential forms, parse
Applications of context-free grammars, Ambiguity
Grammars and languages

Push Down Automata : Definition of

CFG: is a formal grammar which is used to
generate all possible strings in a given
formal language

Context Free languages.

- A context free languages is a language generated by some context free grammar.
- The set of all CFL is identical to the set of languages accepted by pushdown Automata.

Context free grammar is defined by 4-tuples

as $G = \{V, \Sigma, S, P\}$ where

V = Set of variables or Non-terminal symbols

Σ = Set of Terminal symbols

S = start symbol

P = production Rules

CFG has production rule of the form
where $A \rightarrow \alpha$
 $\alpha \in \{V \cup \Sigma\}^*$ and $A \in V$

Example: for generating a language that generates equal no. of a's and b's in the form of $a^n b^n$
the context free grammar will be defined as

$$G = \{ (S, A), \{a, b\}, S, (S \rightarrow aAb, A \rightarrow aAb | \epsilon) \}$$

Sol

$$S \rightarrow aAb$$

$$\rightarrow a aAb b \text{ (by using } A \rightarrow aAb \text{)}$$

$$\rightarrow a a aAb b b \text{ (by using } A \rightarrow aAb \text{)}$$

→ $aaa \in bbb$ (by using $A \rightarrow \epsilon$)

→ $aaabbb$

→ $a^3b^3 \rightarrow a^n b^n$.

→ A CFG is a collection of 3 things

1. An alphabet Σ of letters called terminals from which we are going to make strings that will be the words of the language.
2. A set of symbols called non-terminals of one of which is a symbol 'S' standing for "start"
3. A finite set of productions of the form one non-terminal $\rightarrow$ finite set of terminals and/or non-terminal

Note

1. Every regular grammar is context free so a regular lang is also a context free one.
2. Family of regular languages is a proper subset of CFL's



Fig Relatⁿ b/w CFL & RL

Derivations using a Grammar

— The set of all strings that can be derived from a Grammar is said to be the Language generated from that Grammar.

Example 1: Consider the Grammar $G_1 = (\{S, A\}, \{a, b\}, S, S \rightarrow aAb, aA \rightarrow aaAb, A \rightarrow \epsilon)$

sol $S \rightarrow aAb$ (by $S \rightarrow aAb$).

~~$\rightarrow aaAb$~~ (~~$aA \rightarrow aaAb$~~)

~~$\rightarrow aa$~~ $S \rightarrow aaAb$ (by $aA \rightarrow aaAb$).

~~$S \rightarrow a aaAb b$~~ (by $aA \rightarrow aaAb$)

~~$S \rightarrow aaaa\epsilon bbb$~~ ($A \rightarrow \epsilon$)

~~$\rightarrow aaabb$~~

$S \rightarrow aAb$

$S \rightarrow aaAb$

$S \rightarrow a aaAb b$ ($aA \rightarrow aaAb$)

$S \rightarrow aa aaAb bbb$ ($A \rightarrow \epsilon$)

$S \rightarrow aaaa\epsilon bbbb$

Example 2: $G_2 = (\{S, A, B\}, \{a, b\}, \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\})$

Sol

$S \rightarrow AB$
 $\rightarrow ab.$

$L(G_2) = \{ab\}.$

Eg3:- $G_3 = (\{S, A, B\}, \{a, b\}, \{S \rightarrow AB, A \rightarrow aA|a, B \rightarrow bB|b\})$

Sol

$S \rightarrow AB$
 $\rightarrow a\underline{b}$

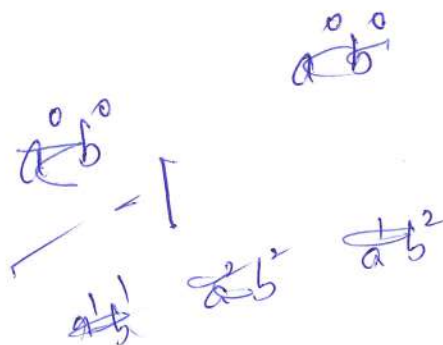
$S \rightarrow AB$
 $\rightarrow aA\underline{B}$
 $\rightarrow a\underline{A}b\underline{B}$
 $\rightarrow aab\underline{b}$
 $\rightarrow aabb.$

$S \rightarrow \underline{A}B$
 $\rightarrow a\underline{A}b$
 $\rightarrow aab$

$S \rightarrow AB$
 $\rightarrow a\underline{b}B$
 $\rightarrow abb$

$L(G_3) = \{ab, a^2b^2, a^2b, ab^2, \dots\}$

$= \{a^m b^n \mid m \geq 0 \text{ and } n \geq 0\}$



Derivation Tree / Parse Tree

A derivation tree or parse tree is an ordered rooted tree that graphically represents the semantic information of strings derived from a context free grammar.

Example :- For the Grammar $G = \{V, T, P, S\}$ where

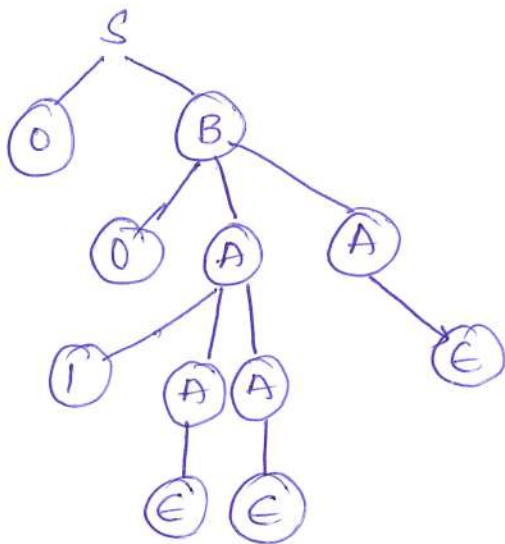
$S \rightarrow OB, A \rightarrow IAA|e, B \rightarrow OAA$

Sol

Root vertex $\rightarrow$ must be labelled by the start symbol

vertex $\rightarrow$ labelled by non-terminal symbols

leaves $\rightarrow$ labelled by terminal symbols or ϵ



Linear Grammar : Any context free grammar that has atmost one non-Terminal / variable in the right hand side of each production is called a linear Grammar.

Eg: $S \rightarrow a \underset{0}{s} b \mid b \underset{0}{s} a \mid a \underset{0}{b} \mid b \underset{0}{a}$

(atmost 1 or zero non terminal) Terminal - small
non terminal - Capital
So S is a linear Grammar

Right linear Grammar :-

The linear grammar in which the only one non terminal is on the rightmost end of production is called right linear Grammar

$S \rightarrow a \underline{A} \mid b \underline{B} \mid \epsilon$

$A \rightarrow a \underline{s} \mid a$

$B \rightarrow b \underline{s} \mid b$

Left linear Grammar :-

A linear grammar in which the only non terminal is on the leftmost ^{end} ~~end~~ of production is called left linear Grammar.

$S \rightarrow \underline{A} a \mid \underline{B} b \mid \epsilon$ $A \rightarrow \underline{s} a \mid a$ $B \rightarrow \underline{s} b \mid b$

Right Recursive Grammar:-

Right recursive grammar is right linear grammar in which the non-terminal is same as that of left hand side of production rules.

$$\underline{S} \rightarrow a\underline{S} \mid b\underline{S} \mid a \mid b.$$

Left Recursive Grammar:-

It is left linear grammar in which the non terminal is same as that is in the left hand side of production rule.

$$\underline{S} \rightarrow \underline{S}a \mid \underline{S}b \mid a \mid b$$

Example for Right linear Grammar & Left linear Grammar

Q The grammar $G = (\{S\}, \{a, b\}, P, S)$ where P is given as $S \rightarrow abs \mid a$ is right linear

Sol

$$\begin{aligned} S &\rightarrow ab\underline{S} \\ &\rightarrow abab\underline{S} \quad \because S \rightarrow abs \\ &\rightarrow ababa \quad \because S \rightarrow a. \end{aligned}$$

Q2: The grammar $G = (\{S, S_1, S_2\}, \{a, b\}, P, S)$ with productions $S \rightarrow S_1ab$, $S_1 \rightarrow S_1ab \mid S_2$, $S_2 \rightarrow a$ is left linear

Sol

$$\begin{aligned} S &\rightarrow \underline{S}_1ab \\ &\rightarrow \underline{S}_1abab \quad \because S_1 \rightarrow S_1ab \\ &\rightarrow \underline{S}_2abab \quad \because S_1 \rightarrow S_2 \\ &\rightarrow aabab \quad \because S_2 \rightarrow a \end{aligned}$$

Q- Construct a LLG, to accept the string with even no. of a's followed by odd no. of b's

Sol

$L = \{b, aab, aabbb, aaaaab, \dots\}$

string should end with compersorily ~~end~~ 1 no. of b.

$S \rightarrow Ab$

string can have odd no. of b's to accept multiple no. of odd b's

$A \rightarrow Abb$

After reading all odd no. of b's even no. of a's should be accepted, Add new variable B to accept a.

$A \rightarrow B$

minimum possible string with a is ϵ

$B \rightarrow \epsilon$

To accept even no. of a's recursively

$B \rightarrow aa$

LL :- $S \rightarrow Ab, A \rightarrow Abb, A \rightarrow B, B \rightarrow \epsilon, B \rightarrow aa$

lets try to derive aabbbb

$S \rightarrow Ab$

$A \rightarrow Abb$ $\therefore A \rightarrow Abb$

$A \rightarrow Bbbb$ $\therefore A \rightarrow B$

$\rightarrow aabbbb \therefore B \rightarrow aa$

$\therefore$ we got derivable string.

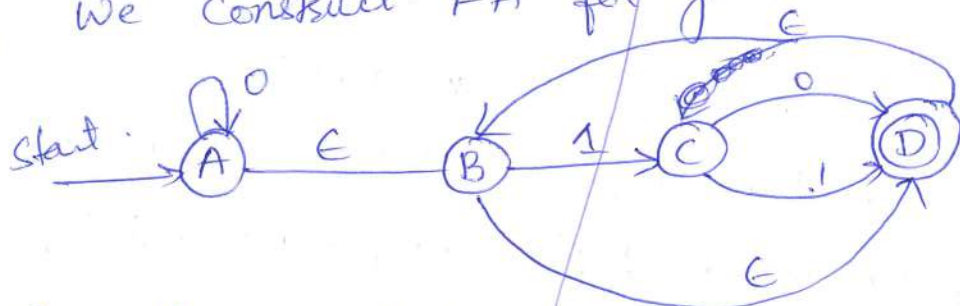
3 - (5)

Example 2 Construct a LLG to accept given Regular expression $R = (aa + ab(bb)^*ba)^*$

So

Construct left linear and right linear grammar for the R.E $0^*(1(0+1))^*$

So We construct FA for given R.E



Eliminating null i.e ϵ moves.

	0	1	ϵ
A	{A}	$\emptyset$	{B}
B	$\emptyset$	{C}	{D}
C	{D}	{D}	$\emptyset$
D	$\emptyset$	$\emptyset$	{B}

Fig: Transition table with ϵ -moves.

	0	1
A	{A, B, D}	{C}
B	$\emptyset$	{C}
C	{B, D}	{B, D}
D	$\emptyset$	{C}

Construction of Right-linear and left-linear grammar for a given R.E

1) Constructing right-linear grammar for a given R.E
the above method of construction of R.G from R.E yields the right linear grammar

2. Constructing left-linear grammar for a given R.E

Step 1: Reverse the give Reg. Exp

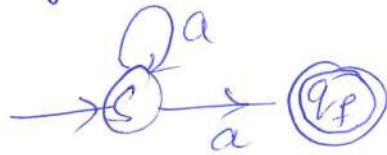
Step 2: Construct the right linear grammar for reversed regular expression using above method.

Step 3: After constructing the right linear grammar in the right hand side of the productions, the positions of terminal string and variable should be interchanged. We will get required left linear grammar.

Example 2:- If a regular grammar G is given by $S \rightarrow aS/a$ find M accepting $L(G)$

Sol Let q_0 correspond to S and q_f which is the new final state

Symbolically $M = (\{q_0, q_f\}, \{a\}, \delta, q_0, \{q_f\})$



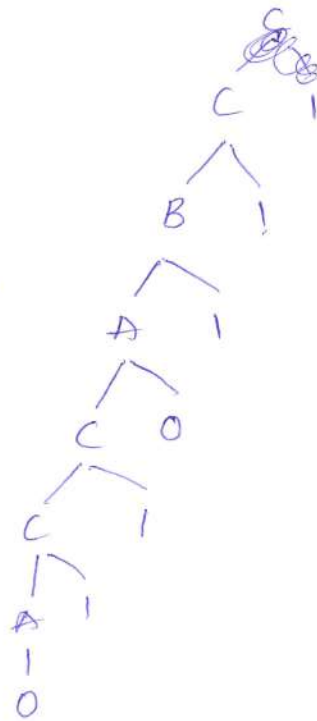
Reverse RLG

$$C \rightarrow C1 \mid B1$$

$$B \rightarrow A1 \mid 1$$

$$A \rightarrow A0 \mid B0 \mid C0 \mid 0$$

011011



001

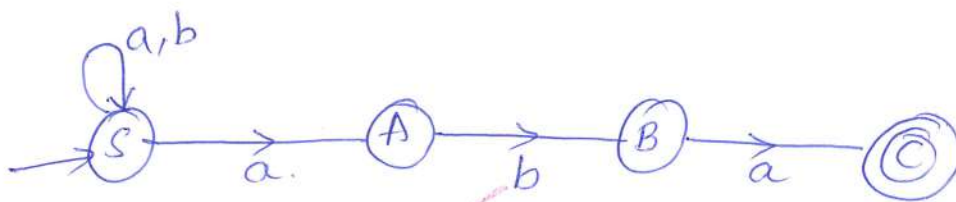


not accepting.

3-8 Continuation

Construct DFA equivalent to the grammar
 $S \rightarrow as \mid bs \mid aA$ $A \rightarrow bB$, $B \rightarrow aC$ $C \rightarrow \epsilon$

Sol



The given grammar is

$$S \rightarrow aS, S \rightarrow bS, S \rightarrow aA, A \rightarrow bB, B \rightarrow aC, C \rightarrow \epsilon$$

The DFA $M = (\{S, A, B, C\}, \{a, b\}, \delta, S, C)$ then the transition system is shown above.

Finite Automata and Left and Right linear Grammars

Example 1:-

Consider Given FA $\rightarrow$ 

So $L = \{ a, aa, ab, aab, \dots \}$

i.e. $L =$ set of strings start with a.

RLG :-

$A \rightarrow aB$

$B \rightarrow aB \mid bB \mid \epsilon$ $\rightarrow$ because B is final state.

Eliminate ϵ

$A \rightarrow aB \mid a$
 $B \rightarrow aB \mid bB \mid a \mid b$

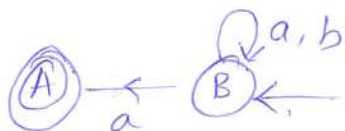
$\therefore$ substitute ϵ in production rules

If we reverse the grammar we get

$A \rightarrow Ba \mid a$

$B \rightarrow Ba \mid Bb \mid a \mid b$ $\left. \vphantom{\begin{matrix} A \rightarrow Ba \mid a \\ B \rightarrow Ba \mid Bb \mid a \mid b \end{matrix}} \right\} L^R \rightarrow$ reverse of the language.
 i.e. set of strings ends with a.

Reverse FA Convert to RLG and Reverse.



$\therefore$ Reverse of FA

$B \rightarrow aA$

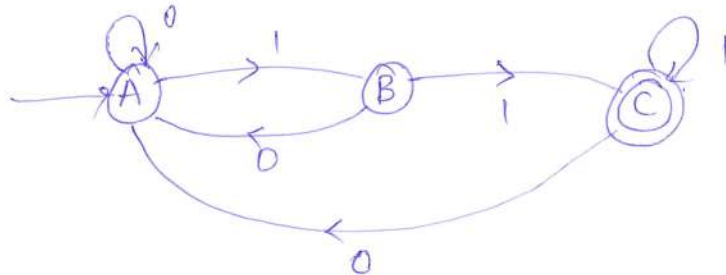
$B \rightarrow aB \mid bB$

$A \rightarrow \epsilon$ ($\because$ because A is final state)

Now Reverse

$B \rightarrow Aa \mid Ba \mid Bb$, $A \rightarrow \epsilon$ $\left. \vphantom{\begin{matrix} B \rightarrow Aa \mid Ba \mid Bb \\ A \rightarrow \epsilon \end{matrix}} \right\} L$ set of strings start with 'a'

Example 2 :- consider



$$L = \{011, 10011, 0111, 010011, \dots\}$$

L = set of strings ends with 11

$$A \rightarrow 0A \mid 1B$$

$$B \rightarrow 1C \mid 0A$$

$$C \rightarrow 0A \mid 1C \mid \epsilon$$

Eliminate ϵ production

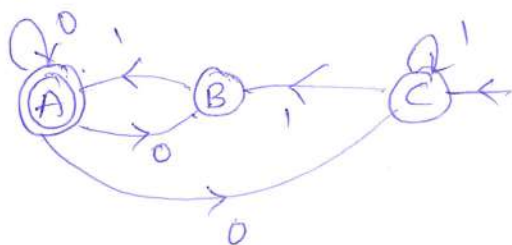
$$A \rightarrow 0A \mid 1B$$

$$B \rightarrow 1C \mid 0A \mid 1$$

$$C \rightarrow 0A \mid 1C \mid 1$$

to eliminate ϵ productions substitute ϵ in

Reverse FA



$$C \rightarrow 1C \mid 1B$$

$$B \rightarrow 1A$$

$$A \rightarrow 0A \mid 0B \mid 0C \mid \epsilon$$

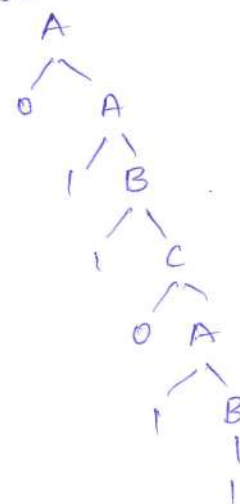
Remove ϵ

$$C \rightarrow 1C \mid 1B$$

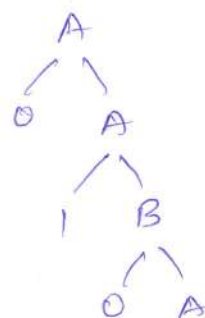
$$B \rightarrow 1A \mid 1$$

$$A \rightarrow 0A \mid 0B \mid 0C \mid 0$$

011011



010



Method to find whether a string belongs to a Grammar or not

1. Start with the start symbol and choose the ~~closest~~ production that matches to the given string.
2. Replace the variables with its most appropriate production. Repeat the process until the string is generated or until no other productions are left.

Example: Verify whether the Grammar

$S \rightarrow OB \mid IA, A \rightarrow O \mid OS \mid IAA \mid \epsilon, B \rightarrow I \mid IS \mid OBB$

generates the string 00110101

Sol

$S \rightarrow OB$	$\therefore B \rightarrow OBB$	$\rightarrow 0011010B$	$\therefore S \rightarrow OB$
$\rightarrow 00BB$	$\therefore B \rightarrow I$	$\rightarrow 0011010I$	$\therefore B \rightarrow I$
$\rightarrow 00IB$	$\therefore B \rightarrow IS$		
$\rightarrow 00IIS$	$\therefore S \rightarrow OB$		
$\rightarrow 0011OB$	$\therefore B \rightarrow IS$		
$\rightarrow 00110IS$			

3-8

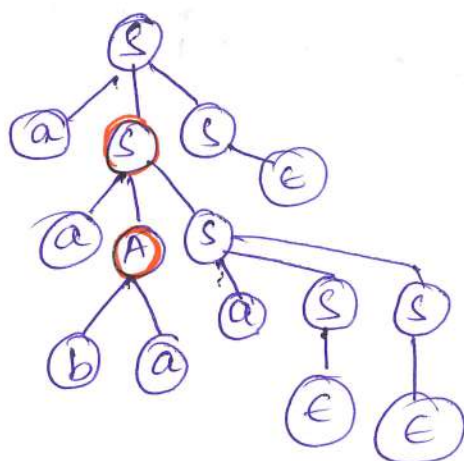
Example: Verify whether the Grammar $S \rightarrow aAb$
 $A \rightarrow aAb | \epsilon$ generates the string aabbb

ss $S \rightarrow aAb$
 $\rightarrow aaAb$ $\because A \rightarrow aAb$
 $\rightarrow aaaaAb$ $\because A \rightarrow aAb$
 $\rightarrow aaaaaAb$
 $\rightarrow aaaaa\epsilon bbb$ $\because A \rightarrow \epsilon$
 $\rightarrow aaaaa\epsilon bbb \Rightarrow \boxed{aaaa\epsilon bbb} \neq aabbb$
 $\therefore aabbb \notin$ to given grammar

Left Derivation Tree

A left derivation tree is obtained by applying production to the leftmost variable in each step.

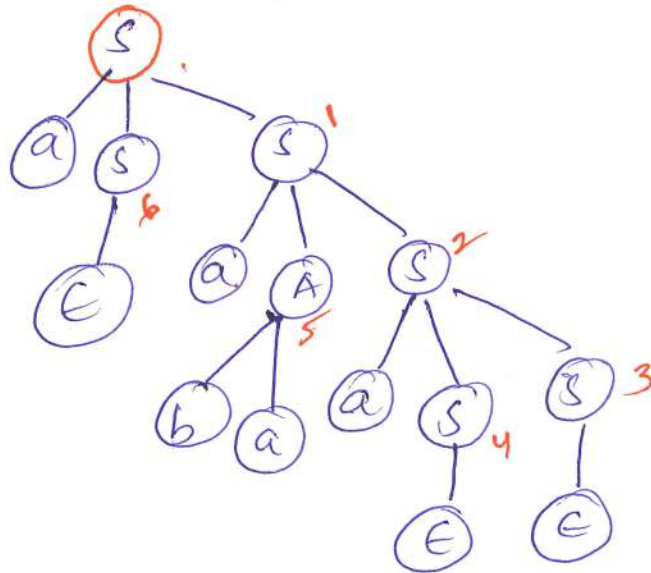
Eg:- for generating the string aabaa from the Grammar $S \rightarrow aAs | ass | \epsilon$, $A \rightarrow sbA | ba$



Right Derivation Tree

A Right Derivation tree is obtained by applying production to the rightmost variable in each step.

→ Consider same example of left derivation



Language of Grammar

Language of grammar is the set of all strings that can be generated from that grammar

→ If language consists of finite no. of strings then it is called as a finite language

→ If language consists of infinite no. of strings then it is called as an infinite language.

Sentential Forms : Right most and left most

Derivations of strings :

Left most derivation:- If a word w is generated by a CFG by a certain derivation and at each step in the derivation a rule of production is applied to the left most non terminal in the working string, then this derivation is called a left most derivation.

Example:- Consider the CFG $S \rightarrow aSA/b, A \rightarrow Ab/a$ for the string $aababa$.

Sol The following is a left most derivation

$S \rightarrow aSA$	$aababa$
$\rightarrow a aSA A$	
$\rightarrow a a \underline{b} A A$	$\because S \rightarrow b$
$\rightarrow a a b \underline{A} b A$	$\because A \rightarrow Ab$
$\rightarrow a a b a b \underline{A}$	$\because A \rightarrow a$
$\rightarrow a a b a b a$	$\because A \rightarrow a$

Note: At every stage in the derivation the non terminal replaced is the leftmost one.

Right most Derivation:

If a word w is generated by a CFG by a certain derivation and at each step in the derivation a rule of production is applied to the right most non terminal in the working string then this derivation is called a rightmost derivation.

Example Consider the CFG $S \rightarrow aSA/b, A \rightarrow Ab/a$ for the string $aababa$.

The following is a right most derivation

Sol

$S \rightarrow aSA$

$\therefore A \rightarrow Ab$

aababa

~~aSA~~

$\rightarrow aSa$

$\rightarrow aSaA$

$\therefore A \rightarrow Ab$

$\rightarrow aSaAb$

$\therefore A \rightarrow a$

$\rightarrow aSaaba$

$\therefore S \rightarrow b$

$\rightarrow aababa$

Note :- At every stage in the derivation the non terminal is the right most one.

Example:- Consider the grammar with production

$$S \rightarrow aAB$$

$$A \rightarrow bBb$$

$$B \rightarrow A/C$$

for the string 'abbb' and find out leftmost and rightmost derivation

2 Let G be the grammar

$$S \rightarrow aB/bA, A \rightarrow a/as/bAA, B \rightarrow b/bS/aBB$$

for the string aabbabab find the Rightmost, leftmost and derivation tree.

3 Let G be a grammar $S \rightarrow 0B/1A, A \rightarrow 0/0S/1AA$
 $B \rightarrow 1/1S/0BB$ for the string 00110101 find derivation tree, rightmost derivation, leftmost derivation

Ambiguous Grammar :-

A grammar is said to be ambiguous if there exists two or more derivation trees for a string w (ω) ~~is~~

(or)

A CFG G is said to be ambiguous if there exists some $w \in L(G)$ that has at least two distinct derivation trees.

→ i.e. Ambiguity implies the existence of two or more leftmost or rightmost derivations.

Example: $G = (\{S\}, \{a+b, +, *\}, P, S)$ where P consists of $S \rightarrow S+S \mid S*S \mid a \mid b$. The string $a+axb$ can be generated as.

<u>80</u>	$S \rightarrow S+S$	$S \rightarrow \underline{S} * S$
	$\rightarrow a+S$	$\rightarrow \underline{S} + S * S$
	$\rightarrow a + S * S$	$\rightarrow a + \underline{S} * S$
	$\rightarrow a + a * S$	$\rightarrow a + a * S$
	$\rightarrow a + a * b.$	$\rightarrow a + a * b$
	L.D.	L.D.

Thus this grammar is ambiguous.

Applications of CFG

→ CFG is of great practical importance

Applications

- * For defining programming language.
- * For parsing the program by constructing syntax tree
- * For translation of programming languages.
- * For describing arithmetic expression
- * For construction of compilers.

Push down Automata

A push down automata (PDA) is a way to implement a context free grammar in a similar way we design finite automata for Regular Grammar

Regular Grammar $\rightarrow$ Regular language $\rightarrow$ Finite State automation

Context free grammar $\rightarrow$ Context free language $\rightarrow$ Pushdown Automata

$\rightarrow$ It is more powerful than FSM

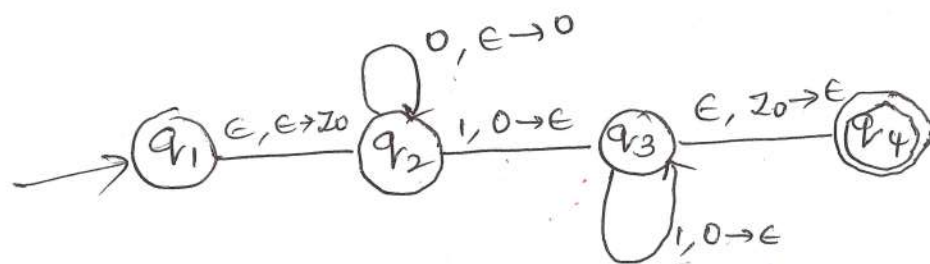
$\rightarrow$ FSM has a very limited memory but PDA has more memory

$\rightarrow$ PDA = Finite State Machine + A stack.

Stack $\begin{cases} \text{push} \\ \text{pop} \end{cases}$ + operations

70
Example 4. Construct a PDA that accepts

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



0 1 1 1

0	1
0	1
z0	

0	1
0	1
z0	

$$L = \{\epsilon, 01, 0011, 000111, \dots\}$$

Here in this example the no. of 'a' and 'b' have to be same

Initially we put a special symbol ' ϵ or z_0 ' into the empty stack.

Then at state q_2 if we encounter input 0 and top is Null, we push 0 into stack, This may iterate. And if we encounter input 1 and top is zero we pop this zero

Then at state q_3 if we encounter input 1 and top is zero we pop this zero. This may also iterate and if we encounter input 1 and top is 0, we pop the top element.

If special symbol z_0 is encountered at top of the stack, it is popped out and it finally goes to the accepting state q_4

8/1/21

Example 3:- formal definition of PDA

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

$$\delta = Q \times (\Sigma \cup \epsilon) \times \Gamma \rightarrow Q \times \Gamma^*$$

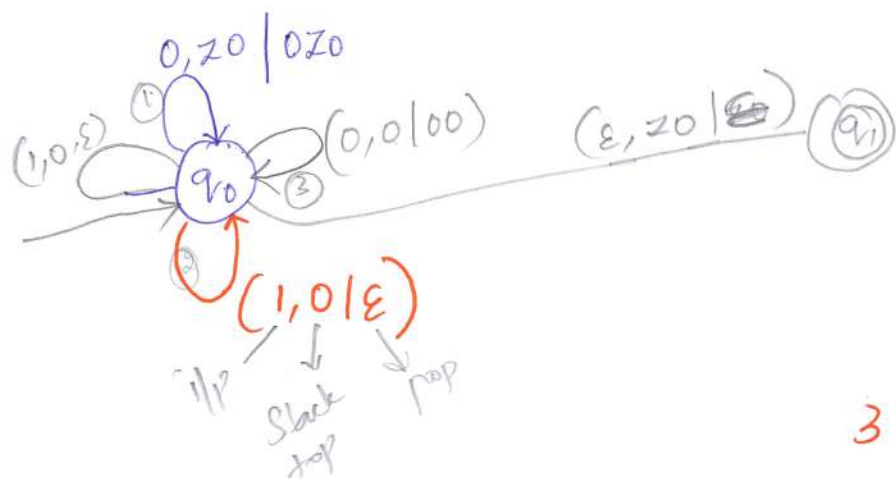
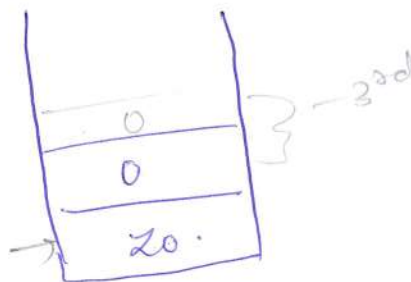
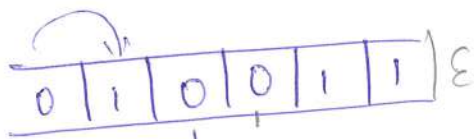
Consider language $L = \{w \in (0,1)^* \mid n_0(w) = n_1(w)\}$

Sol we can get language strings

$\{01, 10, 00, 11, 0011, 1100, 0101, 1010, \dots\}$

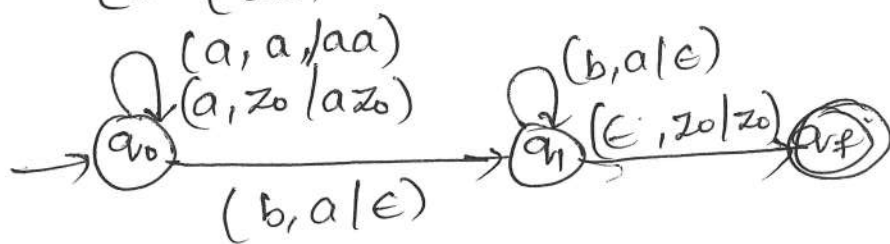
let us consider string 010011

no. of zeros = no. of ones



PDA $\rightarrow a^n b^n \mid n \geq 1$

$L = \{ab, aabb, \dots\}$



IP

$(q_0, a, z_0) = (q_0, az_0)$
 $(q_0, a, a) = (q_0, aa)$
 $(q_0, b, a) = (q_1, \epsilon)$
 $(q_1, b, a) = (q_1, \epsilon)$
 $(q_1, \epsilon, z_0) = (q_f, z_0)$

a | a | b | b



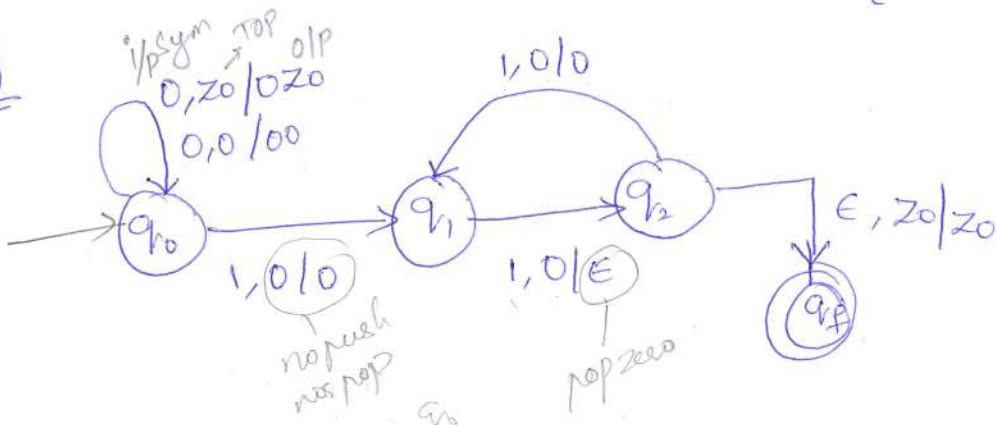
Example 2: PDA for designing $L = \{0^n 1^{2n}, n > 0\}$

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

$\{011, 001111, 00011111, \dots\}$

$\because 0=11$
Constraint

Sol



$$q_0, 0, z_0 \rightarrow 0z_0$$

$$q_0, 0, 0 \rightarrow 00 \text{ (push zero)}$$

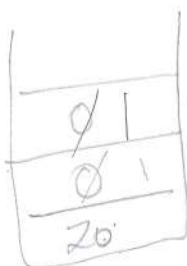
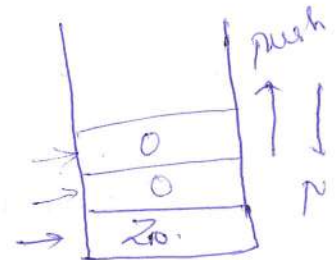
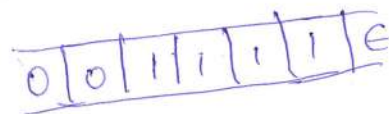
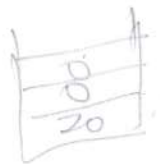
$$q_0, 1, 0 \rightarrow q_1, 0$$

$$q_1, 1, 0 \rightarrow q_2, \epsilon$$

$$q_2, \epsilon, z_0 \rightarrow q_0, z_0$$

$$q_2, 1, 0 \rightarrow q_1, 0$$

if ↓ stack



$$\delta(q_0, 0011100, z)$$

$$\vdash \delta(q_0, 011100, 0z)$$

$$\vdash \delta(q_0, 11100, 00z)$$

$$\vdash \delta(q_{\text{final}}, \epsilon)$$

$$a^n b^{2n}$$

$$aaa bbbbbb$$

$$\delta(q_0, aaa bbbbbb, z)$$

$$\vdash (q_0, aa bbbbbb, az)$$

Final State Acceptability

In final state acceptability, a PDA accepts a string when after reading the entire string, the PDA is in a final state. From the starting state, we can make moves that end up in a final state with any stack values. The stack values are irrelevant as long as we end up in a final state.

For a PDA $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ the language accepted by the set of final states F is -

$$L(PDA) = \{ w \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, x), q \in F \}$$

for any input stack string x .

Empty Stack Acceptability

Here a PDA accepts a string when, after reading the entire string, the PDA has emptied its stack.

For a PDA $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ the language accepted by the empty stack is

$$L(PDA) = \{ w \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon), q \in Q \}$$

Algorithm to find CFG corresponding to a given PDA

Input - A CFG $G = (V, T, P, S)$

o/p: Equivalent PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \Gamma, F)$ such that the non-terminal of the grammar G will be $\{X_{wx} \mid w, x \in Q\}$ and the start state will be A_{q_0}, F

Step 1: for every $w, x, y, z \in Q$ $m \in \Sigma$ and $a, b \in \Sigma$ if $\delta(w, a, \epsilon)$ contains (y, m) and (z, b, m) contain (x, ϵ) add the production rule $X_{wx} \rightarrow a X_{yz} b$ in the grammar

Step 2: for every $w, x, y, z \in Q$ add the production rule $X_{wx} \rightarrow X_{wy} X_{yx}$ in grammar G

Step 3: for $w \in Q$ add the production rule $X_{ww} \rightarrow \epsilon$ in grammar G .

Example:- Construct a PDA from the following

CFG $G = (\{S, x\}, \{a, b\}, P, S)$

where the productions are

$$S \rightarrow xS \mid \epsilon$$

$$x \rightarrow axb \mid \overset{x}{b} \mid ab$$

Sol :- Let the equivalent PDA

$$P = (\{q\}, \{a, b\}, \{a, b, x, \epsilon\}, \delta, q, \epsilon)$$

where $\delta(q, \epsilon, \epsilon) = \{(q, xS), (q, \epsilon)\}$

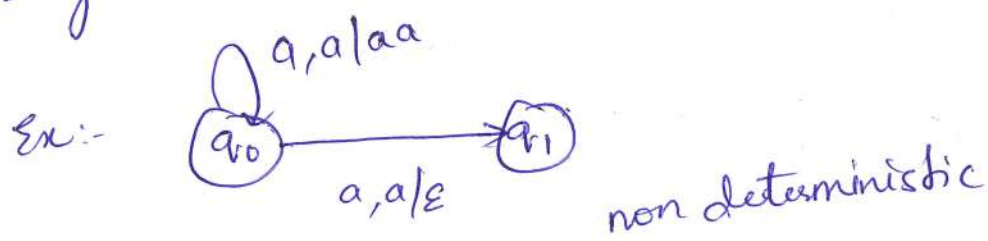
$$\delta(q, \epsilon, x) = \{(q, axb), (q, \overset{x}{b}), (q, ab)\}$$

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

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

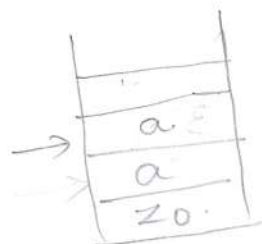
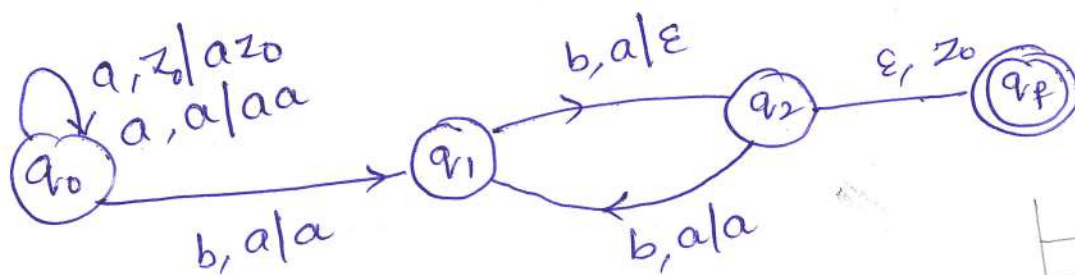
Deterministic Push down Automata (DPDA)

A PDA is said to be deterministic if all derivations in the design has to give only single move.



Ex: Consider $L = \{a^n b^{2n} \mid n > 0\}$

$L = \{abb, aabbbb, \dots\}$



→ For DPDA Every input with the current state there is only one move

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

$$\delta : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma$$

Definition :- A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is deterministic if for every $q \in Q, a \in \Sigma \cup \{\lambda\}, b \in \Gamma$

- ① $\delta(q, a, b)$ contains atmost one element.
- ② if $\delta(q, \lambda, b)$ is not empty then $\delta(q, c, b)$ must be empty for every $c \in \Sigma$

Definition: L is a deterministic Context free language (DCFL) if and only if there exists a deterministic PDA M such that $L = L(M)$

Example: $L = \{a^n b^n \mid n \geq 0\}$ is DCFL

Proof: PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ with $Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b\}, \Gamma = \{0, 1\}, \delta, q_0$

$$\delta(q_0, a, 0) = \{(q_1, 10)\}$$

$$\delta(q_1, a, 1) = \{(q_1, 11)\},$$

$$\delta(q_1, b, 1) = \{(q_2, \lambda)\}$$

$$\delta(q_2, b, 1) = \{(q_2, \lambda)\}$$

$$\delta(q_2, \lambda, 0) = \{(q_0, \lambda)\}.$$

accepts given language. It satisfies the conditions for being deterministic

This PDA is a non deterministic PDA because finding the mid for the given string and reading the string from left and matching it with from right (reverse) direction leads non deterministic moves Here is the TD

$$1. \delta(q_0, a, z) = (q_1, az)$$

$$\delta(q_0, b, z) = (q_1, bz)$$

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

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

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

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

pushing the symbols onto the stack.

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

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

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

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

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

popping the symbols on reading the same kind of symbol

NPDA : is very much similar to NFA.

The CFG which accepts deterministic PDA accepts NPDA as well. Similarly there are some CFGs which can be accepted only by NPDA and not by DPDA. Thus NPDA is more powerful than DPDA.

Ex: Design PDA for palindrome strings.

Suppose $L = \{aba, aa, bb, bab, bbabb, aabaa, \dots\}$

This string can be odd or even palindrome.

→ The logic for constructing PDA is that we will push a symbol onto the stack till half of the string then we will read each symbol and then perform the pop operation.

We will compare to see whether the symbol which is popped is similar to the symbol which is read. Whether we reach to end of the input, we expect the stack to be empty.

CFG to PDA

1. The first symbol on R.H.S production must be a terminal symbol. The following steps are used to obtain PDA from CFG is

Step 1: Convert the given productions of CFG into RLQ / CNF (Grewich Normal form)

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

Step 3: The initial symbol of CFG will be initial symbol in the PDA

Step 4: For non terminal symbol add the following rule

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

where the production rule is $A \rightarrow \alpha$

Step 5: For each terminal symbols add the following rule

$$\delta(q, a, a) = (q, \epsilon) \text{ for every terminal symbol}$$

Example 1: Convert the following grammar to PDA that accepts the same language

$$S \rightarrow 0S1 \mid A$$

$$A \rightarrow 1A0 \mid \epsilon \mid \epsilon$$

Sol The CFG can be first simplified by eliminating unit productions.

$$S \rightarrow 0S1 \mid 1S0 \mid \epsilon$$

Now we will convert CFG to RL

$$S \rightarrow 0Sx \mid 1Sy \mid \epsilon$$

$$x \rightarrow 1$$

$$y \rightarrow 0$$

The PDA can be.

$$R_1: \delta(q, \epsilon, S) = \{(q, \epsilon, 0sx) \mid (q, \epsilon, 1sy) \mid (q, \epsilon)\}$$

$$R_2: \delta(q, \epsilon, x) = \{(q, 1)\}$$

$$R_3: \delta(q, \epsilon, y) = \{(q, 0)\}$$

$$R_4: \delta(q, 0, 0) = \{(q, \epsilon)\}$$

$$R_5: \delta(q, 1, 1) = \{(q, \epsilon)\}$$

~~30/10~~

88
 Example 2: - Construct PDA for the Given CFG and test whether 010^4 is acceptable by this PDA

$$S \rightarrow OBB$$

$$B \rightarrow 0S \mid 1S \mid 0.$$

Sol PDA can be given as.

$$A = \{q, (0, 1) (S, B, 0, 1), \delta, q, s, ?\}$$

The production rules can be.

$$R_1: \delta(q, \epsilon, S) = \{(q, OBB)\}$$

$$R_2: \delta(q, \epsilon, B) = \{(q, 0S) \mid (q, 1S) \mid (q, 0)\}$$

$$R_3 = \delta(q, 0, 0) = \{(q, \epsilon)\}$$

$$R_4 = \delta(q, 1, 1) = \{(q, \epsilon)\}$$

Testing 010^4 i.e. 010000 against PDA

$$\delta(q, 010000, S) \vdash \delta(q, 010000, \underline{OBB})$$

$$\vdash \delta(q, 10000, \underline{BB}) \rightarrow R_1$$

$$\therefore B \rightarrow 1S$$

$$\vdash \delta(q, 10000, 1SB)$$

$$\vdash \delta(q, 0000, SB)$$

$$\vdash \delta(q, \underline{0000}, \underline{OBBB})$$

$$\therefore S \rightarrow OBB$$

$$\vdash \delta(q, 000, BBB)$$

$$\vdash \delta(q, 000, 0BB)$$

$$\therefore B \rightarrow 0$$

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

$$B \rightarrow 0$$

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

$$B \rightarrow 0$$

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

$\therefore$ Accepted 010^4

Example 3 :- Draw Convert the following
PDA CFG to PDA

$$S \rightarrow aSb$$

$$S \rightarrow a|b|\epsilon$$

Sol PDA can be given as

$$P = \{(q_i) (a|b), (s, a, b, z_0), \delta, q, z_0, v\}$$

The mapping function will be δ :

$$R_1 = \delta(q, \epsilon, \epsilon) = \{(q, aSb)\}$$

$$R_2 = \delta(q, \epsilon, \epsilon) = \{(q, a) | (q, b) | (q, \epsilon)\}$$

$$R_3 = \delta(q, a, a) = \{(q, \epsilon)\}$$

$$R_4 = \delta(q, b, b) = \{(q, \epsilon)\}$$

$$R_1 : \delta(q, \varepsilon, z_0) = \{(q, \varepsilon)\}.$$

Consider the string aaabb.

$$\delta(q, aaabb, \varepsilon) \vdash \delta(q, \cancel{a}aabb, \cancel{a}\varepsilon b)$$

$$\vdash \delta(q, aabb, \varepsilon b) \quad \therefore$$

$$\vdash \delta(q, aabb, \cancel{a}\varepsilon bb) \quad \varepsilon \rightarrow a\varepsilon b$$

$$\vdash \delta(q, abb, \varepsilon bb)$$

$$\vdash \delta(q, \cancel{a}bb, \cancel{a}\varepsilon b) \quad \varepsilon \rightarrow a\varepsilon$$

$$\vdash \delta(q, abb, abb) \quad \varepsilon \rightarrow a$$

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

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

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

$\therefore$ string aaabb is accepted

Conversion of PDA to CFG

Let PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0)$ Σ - set of symbols

Construct CFG $G = (V, \Sigma, R, S)$ Σ - set of terminals.

Variables V

1. special symbol S . $\because S$ is one of variable.
2. $[P \times q]$ where P, q are states in Q & x is in Γ
i.e. all other variables are formed by states P and q on x .
(i.e. stack sym)
Combination of

Productions, R

1. for all states P ,

$$S \rightarrow [q_0 z_0 P]$$

2. Let $\delta(\overset{\text{current state}}{q}, \overset{\text{top of stack}}{a}, x) = (r, y_1, y_2, \dots, y_k)$

$$[q \ x \ r_k] \rightarrow a [r \ y_1 \ r_1] [r_1 \ y_2 \ r_2] \dots (r_{k-1} \ y_k \ r_k)$$

for all states $r_1, r_2, \dots, r_k$
new states

$$(3) \delta(q, a, x) = (r, \epsilon)$$

$$[q, x r] \rightarrow a$$

$$(4) \delta(q, \epsilon, x) = (r, \epsilon)$$

$$[q \ x r] \rightarrow \epsilon$$

Example: Convert the following PDA to CFG

$$P = (\{q, p\}, \{0, 1\}, \{x, z\}, \delta, q, z, \phi)$$

$$\begin{aligned} \delta(q, 1, z) &= (q, xz) & \delta(q, 0, x) &= (p, x) \\ \delta(q, 1, x) &= (q, xx) & \delta(p, 1, x) &= (p, \epsilon) \\ \delta(q, \epsilon, x) &= (q, \epsilon) & \delta(p, 0, z) &= (q, z) \end{aligned}$$

$q \times q$
 $q \times p$
 $p \times p$
 $p \times q$
 $q \times z$
 $q \times p$
 $p \times z$
 $p \times p$

Sol

$$V = \{s, [q \times q], [q \times p], [p \times q], [p \times p], [q \times z], [q \times p], [p \times z], [p \times p]\}$$

i, $s \rightarrow [q \times q]$

$s \rightarrow [p \times p]$

ii, $\delta(q, 1, z) = (q, xz)$

$[q \times q] \Rightarrow 1 [q \times q] [q \times q]$

$[q \times q] \rightarrow 1 [q \times p] [p \times q]$

$[q \times p] \rightarrow 1 [q \times q] [q \times p]$

$[q \times p] \rightarrow 1 [q \times p] [p \times p]$

iii, $\delta(q, 1, x) = (q, xx)$

$[q \times q] \rightarrow 1 [q \times q] [q \times q]$

$[q \times q] \rightarrow 1 [q \times p] [p \times q]$

$[q \times p] \rightarrow 1 [q \times q] [q \times p]$

$[q \times p] \rightarrow 1 [q \times p] [p \times p]$

④ $\delta(q, \epsilon, x) = (q, \epsilon)$

$[q \times q] \rightarrow \epsilon$

⑤ $\delta(q, 0, x) = (p, x)$

$[q \times q] \rightarrow 0 [p \times q]$

$[q \times p] \rightarrow 0 [p \times p]$

⑥ $\delta(p, 1, x) = (p, \epsilon)$

$[p \times p] \rightarrow 1$

⑦ $\delta(p, 0, z) = (q, z)$

$[p \times q] \rightarrow 0 [q \times q]$

$[p \times p] \rightarrow 0 [q \times p]$

Example 2:

Convert the following PDA to CFG

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

δ is given as follows:

$$\delta(q_0, 1, z_0) = (q_0, xz_0)$$

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

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

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

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

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

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

Sol produce & production.

$$S \rightarrow [q_0 z_0 q_0]$$

$$S \rightarrow [q_1 z_0 q_1]$$

assume

$$A = [q_0 z_0 q_0] \quad B = [q_0 z_0 q_1]$$

$$S \rightarrow A|B$$

for 1st transition.

$$\delta(q_0, 1, z_0) = (q_0, xz_0) \quad \text{4 production}$$

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

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

$$\begin{array}{l} [q_0 z_0 q_1] = 1 [q_0 x q_0] [q_0 z_0 q_1] \\ [q_0 z_0 q_1] = 1 [q_0 x q_1] [q_1 z_0 q_1] \end{array}$$

for 2nd transition.

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

$$[q_0 x q_0] \rightarrow 1 [q_0 x q_0] [q_0 x q_0]$$

$$[q_0 x q_0] \rightarrow 1 [q_0 x q_1] [q_1 x q_0]$$

$$[q_0 x q_1] \rightarrow 1 [q_0 x q_0] [q_0 x q_1]$$

$$[q_0 x q_1] \rightarrow 1 [q_0 x q_1] [q_1 x q_1]$$

for 3rd transition

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

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

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

for 4th transition

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

$$[q_0 x q_1] \rightarrow \epsilon$$

for 5th transition.

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

$$[q_1 x q_1] \rightarrow \epsilon$$

for 6th transition.

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

$$[q_1, a, q_0] \rightarrow 0 [q_1, a, q_0] [q_0, a, q_0]$$

$$[q_1, a, q_0] \rightarrow 0 [q_1, a, q_1] [q_1, a, q_0]$$

$$[q_1, a, q_1] \rightarrow 0 [q_1, a, q_0] [q_0, a, q_1]$$

$$[q_1, a, q_1] \rightarrow 0 [q_1, a, q_1] [q_1, a, q_1]$$

for 1st transition

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

$$(q_1, z_0, q_1) \rightarrow 0.$$

Example 3: - $\delta(q_0, a, z_0) = (q_0, a, z_0)$

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

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

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

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

$$M = \{q_0, q_1\}, \{a, b\}, \delta, (z_0, a), q_0, z_0, \phi)$$

sol

$$V = \{ \epsilon, q_0 z_0 q_1, q_0 z_0 q_0, q_0 a q_0, q_0 a q_1, q_1 z_0 q_1, q_1 z_0 q_0, q_1 a q_1, q_1 a q_0 \}$$

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

4th transition.

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

$$[q_0 z_0 q_0] = a [q_0 a q_0] [q_0 z_0 q_0]$$

$$[q_0 z_0 q_0] = a [q_0 a q_1] [q_1 z_0 q_0]$$

$$[q_0 z_0 q_1] = a [q_0 a q_0] [q_0 z_0 q_1]$$

$$[q_0 z_0 q_1] = a [q_0 a q_1] [q_1 z_0 q_1]$$

$$\delta(q_0, a, x) = (q_0, xx)$$

$$[q_0, x, q_0] \rightarrow a [q_0, x, q_0] [q_0, x, q_0]$$

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

$$[q_0, x, q_p] \rightarrow a [q_0, x, q_0] [q_0, x, q_p]$$

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

$$3^{rd} \quad \delta(q_0, b, x) = (q_1, \epsilon)$$

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

$$4^{th} \quad \delta(q_1, b, x) = (q_1, \epsilon)$$

$$[q_1, x, q_1] \rightarrow b$$

$$5^{th} \quad \delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$$

$$[q_1, z_0, q_1] \rightarrow \epsilon$$