MODULE-3 (PART-I)

CONTEXT FREE GRAMMERS & LANGUAGES

Topic Learning objectives:- At the end of the Chapter You must be able to:

1. Define Context Free Grammer with an example
2. Explain derivations Using the grammer. Derive a string w using the grammer G.
3. Define (i) Left Most Derivation (LMD) (ii) Right Most Derivations (RMD) (iii) Parse tree (iv) Sentential form
4. Design Context Free Grammer (CFG) for given Languages (problems)
5. Construct LMD, RMD and Parse tree for any string using given grammer. (Problems)
6. Define Language of a grammer
7. Construct Parse tree for a given grammer and an input string. (Problems)
8. Define ambigious grammer
9. Show that the given grammer is ambigious (Problems)
10. Explain the aplications of Context Free grammers.

Possible Problems:-

1. Design Context Free Grammer for the following Languages.
   (i) Set of all strings of 0's and 1's which are Palindrome strings OR $L = \{ w = w^R \mid w \text{ is in } \{a,b\}^* \}$ R - reverse
   (ii) $L = \{0^n 1^n \mid n \geq 1\}$
   (iii) $L = \{a^{2n} b^m \mid n \geq 0, m \geq 0\}$
   (iv) $L = \{0^{n+2} 1^n \mid n \geq 1\}$
   (v) $L = \{a^i b^j c^k \mid i+j = k, i \geq 0, j \geq 0\}$
   (vi) $L = \{a^n b^m c^k \mid n+2m = k$
   (vii) $L = \{a^n b^m \mid m > n \text{ and } n \geq 0\}$

(viii) Set of all strings of equal number of a's ~~and b's~~ and b's OR $L = \{ w \mid n_a(w) = n_b(w) \}$

(ix) $L = \{ w \mid w \in \{0,1\}^*$ with atleast one occurence of 101 $\}$

(x) $L = \{ a^i b^j c^k \mid i = j + k \}$ over $\Sigma = \{a, b, c\}$

(xi) $L = \{ a^n b^n c^i \mid n \geq 0;\ i \geq 1 \} \cup \{ a^n b^n c^m d^m \mid n, m \geq 0 \}$

(xii) $L = \{ a^n b^m c^k \mid k = m + n,\ n, m \geq 0 \}$

2. Construct (i) Leftmost Derivation (ii) Right Most Derivation (iii) Parse tree for the string aaabab using the grammer: $S \rightarrow AbB \quad A \rightarrow aA \mid \epsilon \quad B \rightarrow aB \mid bB \mid \epsilon$

3. Construct (i) LMD (ii) RMD (iii) Parse Tree for the String $+ * - xy\, xy$ using the grammer
$E \rightarrow +EE \mid *EE \mid -EE \mid x \mid y$

4. Design Grammer (CFG) for Valid arithmetic expressions over operators + and −. The arguments of the expressions are Valid identifier over Symbols a, b, 0 and 1.

5. Show that the following grammer is ambigious.
$E \rightarrow E + E \mid E * E \mid I \mid (E)$
$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$. Find Un-ambigious grammer.

6 Consider the grammer. $S \rightarrow SbS \mid a$. This grammer is ambigious. Show in Particular that the string ababab a has two (i) Parse trees (ii) LMD's (iii) RMD's.

7. Construct (i) LMD (ii) RMD (iii) Parse tree for the String aabbaa Using the grammer $S \rightarrow AS \mid \epsilon$
$A \rightarrow aa \mid ab \mid ba \mid bb$.

8. Prove that the following grammer is ambigious $S \to aS \mid aSbS \mid \epsilon$. Show in Particular that the string 'aab' has two (i) LMD's (ii) RMD's (iii) Parse trees.

9. Show that the grammer $S \to AB \mid aaB$ $A \to a \mid Aa$ $B \to b$ is ambigious.

10. Discuss the applications of CFG.

11. Write (i) Leftmost Derivation (ii) Parse tree for the string $0-((1*0)-0)$ using the grammer $E \to E*T \mid T$ $T \to F-T \mid F$ $F \to (E) \mid 0 \mid 1$

Introduction to Context Free Grammer (CFG):-

Every language has its own Grammer, For example English has its own grammer. A grammer Consist of Set of Rules that are applied to form Valid Sentences. Following are some of the rules of an English grammer.

Rule 1: <Sentence> → <noun> <verb> <adverb>

Rule 2: <noun> → John | Robert

Rule 3: <Verb> → Spoke | ran | ate

Rule 4: <adverb> → well | slowly | quickly

Consider the Sentence 'John ate quickly'

Components of a Grammer
- Variables (Non-Terminal Symbols)
- Terminals
- Productions (Rules)
- Start Symbol

Variables: In above 4 Rules, <Sentence>, <noun> <verb> and <adverb> are Called Variables

Terminals:- the words 'John' and 'Robert' are terminals

Productions (Rules): The above four Rules are productions These Rules are applied to obtain the Sentence.

Start Variable: <Sentence> is a start variable since from here we start deriving the given sentence.

Let us derive 'John ate quickly'
Begin with Start Variable <sentence>. Rules applied ↓

<Sentence> (Start Variable)	⟹ <u><noun></u> <verb> <adverb>	(Rule 1)
	⟹ John <Verb> <adverb>	(Rule 2)
	⟹ John ate <adverb>	(Rule 3)
	⟹ <u>John ate quickly</u>	(Rule 4)

Since the given Sentence is derived from Start Variable the <u>Sentence is said to be Valid</u> <u>and is in the</u> language of grammer.

<u>NOTE:</u> Rules are also called <u>Productions</u>

A Production (Rule) Consist of following general form

Variable → String of zero or more
(<u>Production head</u>) ↓ Terminals & Variables
production Symbol (<u>Production body</u>)

example <Sentence> → <noun> <verb> <adverb>

<u>Formal Definition of Context Free Grammer</u> (CFG)

A Grammer $G = (V, \Sigma, P, S)$ Consist of

(1) Finite Set of Variables (Non-Terminals) denoted by V.

(2) Finite Set of Terminal Symbols denoted by Σ.
these terminal symbols form the string

(3) Finite Set of <u>productions</u> (Rules) denoted by P. They represent recursive definition of Language.

A Production Consist of (i) Variable (head of the production) (ii) production Symbol → & (iii) String of zero or more terminals and Variables. This string is called as <u>body of production</u>, It represent one way to form strings in the language of Variable of head.

Example :- Following is the Context Free Grammer that Generates language Consisting of Set of all binary strings that are Palindromes.

$G = (V, T, P, S)$

$V = \{P\}$ — Set of Variables

$T = \{0, 1\}$ — set of terminal symbols

$P = \{P \to \epsilon,\ P \to 0,\ P \to 1,\ P \to 0P0,\ P \to 1P1\}$ — set of productions (Rules)

$S = P$ (Start variable)

<u>Note</u>: Each Variable in Set V generates (derives) a <u>language</u> (A specific class of strings)

For example P is a Variable that generates all Palindrome strings. In order to derive Palindrome Strings, We Begin with <u>Start Variable</u> of grammer, we apply appropriate <u>Productions</u> to replace the Variable by means of <u>production body</u> and finally we obtain <u>Palindrome String</u>

For <u>example</u>: Let us <u>derive</u> a string 01010

Begin with P (Start Variable)

$\underline{P} \Rightarrow 0\underline{P}0$ (Production $P \to 0P0$ is applied)

$\Rightarrow 01\underline{P}10$ (— " — $P \to 1P1$ — " —)

$\Rightarrow 01010$ (— " — $P \to 0$ — " —)

Let us derive another string $w = 10101$

$\underline{P} \Rightarrow 1\underline{P}1$ ($P \to 1P1$ is applied)

$\Rightarrow 10\underline{P}01$ ($P \to 0P0$ — " —)

$\Rightarrow 10101$ ($P \to 1$ is applied)

The Language generated by a grammer G is denoted by L(G). L(G) Contain the Set of all Strings w that Can be derived using Start Variable of G.

$L(G) = \{\epsilon, 0, 1, 010, 101, 01010, 10101, 1001, 0110, 1100$
$001100, \ldots\}$

Note: Productions (Rules) are the major component of a Grammer, Given a language, we have to Create productions in such a way that if we apply them we must be able to derive all the strings in the language.

Notational Conventions useful to write a grammer

These Symbols are Variables (Non-Terminals)

1. Upper case letter early in alphabet such as A, B, C, etc.
2. letter S when it appears is a Start Variable
3. Lower case italic letters such as <sentence>, <verb>, <adverb>

These Symbols are Terminals (Terminal Symbols)

1. Lower case letters early in alphabet such as a, b, c, etc
2. Operator symbols such as +, −, *, /, etc.
3. Punctuation symbols (,), {, }, ;, ,, etc
4. digits such as 0, 1, 2, ... etc.
5. Each Keyword such as if, else, int, for and each identifier such as id is a terminal symbol.

A Generic Production Can be Written as

$A \rightarrow \alpha$

(Production head) (Production body)

Ex: $A \rightarrow \underbrace{AaBa}_{\alpha}$

Set of productions with Common Production head such as $A \rightarrow \alpha_1$, $A \rightarrow \alpha_2$, ... $A \rightarrow \alpha_n$ Can be written as: $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ where $\mid$ is Union (+) operator.

Note: 1) α, β, γ, represent strings of terminals & Variables
For example $\alpha = AaBb$

2) u, v, w, x, y, z represent string of only terminal Symbols.
For example $w = ababb$

Derivations:- A Derivation is a process of infering or deriving that Certain strings are present in the language of a Variable.

We Begin with Start-Variable of grammer. At each step in derivation, We replace each Variable in Sentential form with Suitable production body.

For example, Consider the grammer $G = (V, P, T, S)$

where $V = \{S, A\}$

$T = \{a, b\}$

$$P = \{\overbrace{S \rightarrow A, \; A \rightarrow aAb \quad A \rightarrow bAa, \; A \rightarrow \epsilon \quad A \rightarrow aAa \quad A \rightarrow bAb}^{\text{Productions}}\}$$

$S = S$ (Start Variable)

Let us derive that String $W =$ abbaba is in language of grammer, Begin with Start Variable of grammer.

Note:- If Variable S is Specified in grammer, it is taken as start Variable. If S is not found in the grammer then production head of first production is the start Variable.

In general we must derive $S \overset{*}{\Rightarrow} w$

$*$ means String w is derived in zero or more steps.

Note: $\rightarrow$ (Production Symbol)

$\Rightarrow$ (Derivation Symbol)

$\underline{S} \Rightarrow A$ ($S \rightarrow A$ is applied)

$\Rightarrow a\underline{A}a$ ($A \rightarrow aAa$ —"—)

$\Rightarrow ab\underline{A}ba$ ($A \rightarrow bAb$ —"—)

$\Rightarrow abbAaba$ ($A \rightarrow bAa$)

$\Rightarrow abb\,aba$ ($A \rightarrow \epsilon$)

(Derivation of String abbaba from Start Variable S).

Note: We replace each Variable that is underlined with appropriate production body

Each intermediate step in a derivation (Such as A, aAa, $abAba$) is Called Sentential form.

(1) Leftmost Derivation (LMD) :- A derivation $S \Rightarrow w$ in which at each step we replace leftmost variable (Non Terminals) of a sentential form with appropriate production body is called Leftmost Derivation. We indicate that the derivation is leftmost by using $\xRightarrow{lm}$ at each step in a derivation.

(2) Rightmost Derivation (RMD) :- A Derivation in which at each step we replace the Rightmost Variable (N.T Symbol) of a sentential form with appropriate production body is called Rightmost derivation. We indicate Rightmost derivation using $\xRightarrow{rm}$ at each step in a derivation.

For example Consider the following grammer :-

$S \rightarrow AbB \qquad A \rightarrow aA \mid \epsilon \qquad B \rightarrow aB \mid bB \mid \epsilon$

Let us derive a string $w = aaabab$

(1) Left most Derivation (LMD)

Begin with Start Variable of given grammer.

$$
\begin{aligned}
S &\xRightarrow{lm} \underline{A}bB \quad (S \rightarrow AbB) \quad \text{production applied} \\
&\xRightarrow{lm} a\underline{A}bB \quad (A \rightarrow aA) \\
&\xRightarrow{lm} aa\underline{A}bB \quad (A \rightarrow aA) \\
&\xRightarrow{lm} aaa\underline{A}bB \quad (A \rightarrow aA) \\
&\xRightarrow{lm} aaab\underline{B} \quad (A \rightarrow \epsilon) \\
&\xRightarrow{lm} aaaba\underline{B} \quad (B \rightarrow aB) \\
&\xRightarrow{lm} aaababB \quad (B \rightarrow bB)
\end{aligned}
$$

(2) Right most Derivation : (RMD)

$$
\begin{aligned}
S &\xRightarrow{rm} Ab\underline{B} \quad (A \rightarrow AbB) \\
&\xRightarrow{rm} Aba\underline{B} \quad (B \rightarrow aB) \\
&\xRightarrow{rm} AbabB \quad (B \rightarrow bB) \\
&\xRightarrow{rm} \underline{A}bab \quad (B \rightarrow \epsilon) \\
&\xRightarrow{rm} a\underline{A}bab \quad (A \rightarrow aA) \\
&\xRightarrow{rm} aa\underline{A}bab \quad (A \rightarrow aA) \\
&\xRightarrow{rm} aaaAbab \quad (A \rightarrow aA) \\
&\xRightarrow{rm} aaabab \quad (A \rightarrow \epsilon)
\end{aligned}
$$

(3) Sentential form :-

Derivations from start symbol produce strings. These strings are called "sentential forms". That is if $G = (V, T, P, S)$ is a CFG, then any string α in $(V \cup T)^*$ such that $S \xRightarrow{*} \alpha$ is a Sentential form.

If $S \xRightarrow{lm} \alpha$ then α is a left sentential form

If $S \xRightarrow{rm} \alpha$ then α is a right sentential form

• Parse tree :- A Parse tree is a pictorial representation of the derivation $S \overset{*}{\Rightarrow} w$, It has following Properties

1. The root node is labelled with Start Variable of grammer.
2. Intermediate nodes are labelled with Variables of grammer
3. Leaf nodes are labelled with either Terminal Symbols or ϵ.

Construction of Parse tree for the string $w = aaabab$ Using the grammer of Previous example :-

1. Construct Root Node labelled with Start Variable of grammer.
2. Construct Subnodes using appropriate production of grammer. Here appropriate means the production body that matches the string aaabab to a maximum extent.

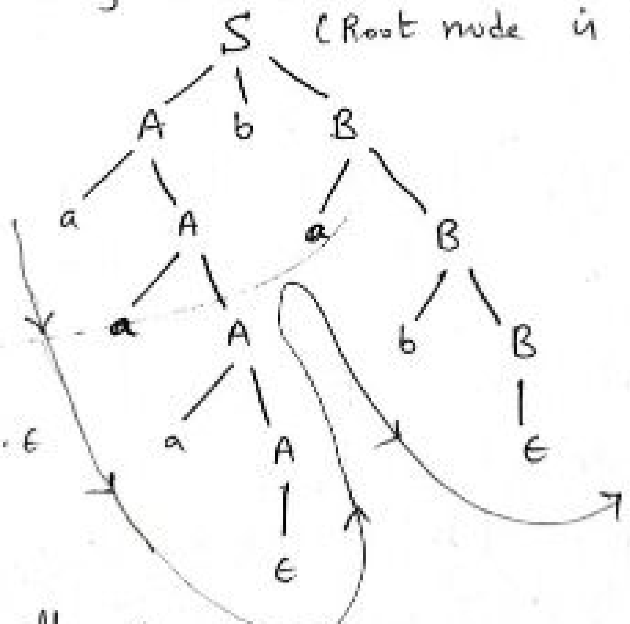
 Our objective is to derive $w = aaabab$.

3. Stop Constructing Subnodes when all the leaf nodes are terminal symbols.

Example: Grammer: $S \to AbB$ $A \to aA \mid \epsilon$ $B \to aB \mid bB \mid \epsilon$

input string $w = aaabab$

(Root node is Start Variable of G)



Yield $w = a.a.a.\epsilon.b.a.b.\epsilon$
$= aaabab$

If we Concatinate all the leaf nodes from left to right, We obtain a string of only terminals. This string is Called yield of the Parse tree. The Parse tree Constructed above Corresponds to Leftmost Derivation.

Problems: (1) Design Context free Grammer for the language

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

Solution:-

$L = \{01, 0011, 000111, 00001111, \ldots\}$

Therfore the language is set of all strings consisting of n no. of 0's followed by n number of 1's

Now create productions such that if we apply them in derivation, ∴ we must be able to derive all strings of L.

$S \rightarrow 01 \mid 0S1$

The grammer is $G = (V, T, P, S)$

$V = \{S\}$

$T = \{0, 1\}$

$P = \{S \rightarrow 01 \mid 0S1\}$

$S = S$ (Start variable

$w = 01$

$S \Rightarrow 01$ $(S \rightarrow 01)$

$w = 0011$

$S \Rightarrow 0S1$ $(S \rightarrow 0S1)$

$\Rightarrow 0011$

$w = 000111$

$S \Rightarrow 0\underline{S}1$

$\Rightarrow 00S11$

$\Rightarrow 000$

(2) Design CFG for $L = \{a^{2n} b^m \mid n \geq 0, m \geq 0\}$

Solution:- $L = \{ \underset{\substack{n=0\\m=0}}{\epsilon}, \underset{\substack{n=0\\m=1}}{b}, \underset{n=m=1}{\underline{aa}b}, \underset{n=2\ m=2}{\underline{aaaa}bb}, \underset{n=3\ m=3}{\underline{aaaaaa}bbb}, \ldots \ldots \}$

The language is set of all strings containing either ϵ or any even no. of a's followed by any no. of b's.

$S \rightarrow AB$

$A \rightarrow aaA \mid \epsilon$ A – generates even no of a's

$B \rightarrow bB \mid \epsilon$ B – " — any no. of b's

$$\therefore G = \left(\underbrace{\{S, A, B\}}_{V}, \underbrace{\{a, b\}}_{T}, \underbrace{\{S \rightarrow AB, A \rightarrow aaA \mid \epsilon, B \rightarrow bB \mid \epsilon\}}_{P}, S \right)$$

(3) Design CFG for $L = \{a^{2n} b^n \mid n \geq 0\}$

$L = \{ \underset{n=0}{\epsilon}, \underset{n=1}{aab}, \underset{n=2}{aaaabb}, \ldots \ldots \}$

$S \rightarrow aaSb \mid \epsilon$

$\therefore G = (V, T, P, S)$ where $V = \{S\}$

$T = \{a, b\}$ $P = \{aaSb \mid \epsilon\}$

$S = S$ (Start variable)

(4) Design CFG or Grammer for $L = \{ 0^{n+2} 1^n \mid n \geq 0 \}$

Solution:- $L = \{ \underset{n=0}{\underline{00}}, \underset{n=1}{\underline{00}01}, \underset{n=2}{\underline{00}0011}, \underset{n=3}{\underline{00}000111}, \ldots \}$

∴ Each string has 0's followed by 1's and two excess 0's Compared to 1's.

∴ Therefore the grammer is : $S \to 00 \mid 0S1$

$$G = (\underbrace{\{S\}}_{V}, \underbrace{\{0,1\}}_{T}, \underbrace{\{S \to 00 \mid 0S1\}}_{P}, \underbrace{S}_{S})$$

(5) Design CFG for $L = \{ a^n b^n c^n \mid n \geq 0 \}$

Solution:- $L = \{ \underset{n=0}{\epsilon}, \underset{n=1}{abc}, \underset{n=2}{aabbcc}, \underset{n=3}{aaabbbccc}, \ldots \}$

each w is of the form $a^n b^n$ followed by c^n.

(i) $a^n b^n$ is generated by $S \to A$
$A \to aAb \mid \epsilon$

(ii) c^n is generated by $C \to cC \mid \epsilon$

∴ $a^n b^n c^n$ is generated by $S \to AC \mid \epsilon$
$A \to aAb \mid \epsilon$
$C \to cC \mid \epsilon$

∴ $G = (V, T, P, S)$ where $V = \{S, A\}$, $T = \{a, b, c\}$
$P = \{ \underline{S \to AC \mid \epsilon}, \underline{A \to aAb \mid \epsilon}, C \to cC \mid \epsilon \}$

(6) Design CFG for $L = \{ a^i b^j c^k \mid i+j = k, i \geq 0, j \geq 0 \}$

Let $w = a^i b^j c^k$
substitute $i+j$ for k

$= a^i b^j c^{i+j}$

$= a^i b^j c^i c^j = a^i \underline{b^j c^j} c^i$

Every string has (i) equal number of a's and c's
(ii) equal number of b's and c's

(i) ∴ n number of a's followed by n no. of b's is generated by the following Productions:- $S \to aSc$

(ii) ∴ n number of b's followed by n no. of c's is generated by the Production
$A \to bAc$

Combine two productions and produce following grammer.

$S \to aSc \mid A$

$A \to bAc \mid \epsilon$

$G = (V, T, P, S)$ where $V = \{S, A\}$, $T = \{a, b, c\}$

$P = \{aSc \mid A,\ A \to bAc \mid \epsilon\}$

$S = S$ (Start Variable)

(7) Design CFG for $L = \{a^n b^m c^k \mid n + 2m = k\}$

<u>Solution:</u> Let $w = a^n b^m c^k$

Substitute $n+2m$ for K

$w = a^n b^m c^n c^{2m} = a^n b^m c^{2m} c^n$

The string w a has (i) n no. of a's followed by n no. of b's.

(ii) m number of b's followed by 2m number of c's

$a^n c^n$ is generated by $S \to aSc \mid A$

$b^m c^{2m}$ is generated by $A \to bAcc \mid \epsilon$

$\therefore a^n b^m c^{2m} c^n$ is generated by: $S \to aSc \mid A$

$A \to bAcc \mid \epsilon$

The grammer for L is $G = (V, T, P, S)$
where $V = \{S, A\}$ $T = \{a, b, c\}$
$P = \{S \to aSc \mid A \quad A \to bAcc \mid \epsilon\}$ $S = S$ (Start Variable)

(8) Design CFG for $L = \{a^n b^m \mid m > n$ and $n \geq 0\}$

<u>Solution:</u>- $L = \{\underset{n=0,\ m=1}{b}, \underset{n=0,\ m=2}{bb}, \underset{n=1,\ m=2}{abb}, \underset{n=2,\ m=3}{aabbb}, \underset{n=2,\ m=4}{aabbbb}, \ldots\}$

each String w is of the form $a^n b^n b^+$ ('b^+ means one or more b's)

$a^n b^n$ is generated by $S \to aSb$

b^+ is generated by $B \to bB \mid \epsilon$

$a^n b^n b^+$ is generated by $S \to aSb \mid B$

$B \to bB \mid \epsilon$

$G = V, T, P, S$ where $V = \{S, B\}$, $T = \{a, b\}$ $P = \{S \to aSb \mid B$
$B \to bB \mid \epsilon\}$
and $S = S$ (Start Variable of G)

(9) Design CFG for $L = \{ w \mid n_a(w) = n_b(w) \}$

L Consist of Set of all strings where number of a's equal to number of b's $L = \{ab, ba, abab, baab, abba, aabb, bbaa, \ldots\}$

$S \to A$, $A \to aAb \mid bAa \mid aAa \mid bBb \mid \epsilon$

$$\boxed{G = (\underbrace{\{S, A\}}_{V}, \underbrace{\{a, b\}}_{T}, \underbrace{S \to A,\ A \to aAb \mid bAa \mid aAa \mid bBb \mid \epsilon}_{P}, S)}$$

(10) Design CFG for $L = \{ w \mid w \in \{0,1\}^*$ with atleast one occurance of 101 $\}$

$L = \{101, 110\underline{101}, \underline{101}0, 0101\underline{01}, \ldots\}$

$S \to A101A$
$A \to 0A \mid 1A \mid \epsilon$ grammer for L is

$$\boxed{G = (\underbrace{\{S, A\}}_{V}, \underbrace{\{0, 1\}}_{T}, \underbrace{S \to A101A,\ A \to 0A \mid 1A \mid \epsilon}_{P}, S)}$$

(11) Design CFG for the language $L = \{ a^i b^j c^k \mid i = j + k \}$ over $\Sigma = \{a, b, c\}$

Solution: Consider the G string $w = a^i b^j c^k$
Substitute $j+k$ for i in w

$= a^{j+k} b^j c^k$

$= a^j a^k b^j c^k$

$= a^k \underline{a^j b^j} c^k$

$a^k c^k$ is generated by $S \to aSc$
$a^j b^j$ is generated by $A \to aAb$
$a^k a^j b^j c^k$ is generated by $S \to aSc \mid A$
∴ Grammer for given L is $A \to aAb \mid \epsilon$

$$\boxed{G = (\underbrace{\{S, A\}}_{V}, \underbrace{\{a, b, c\}}_{T}, \underbrace{\{S \to aSc \mid A,\ A \to aAb \mid \epsilon\}}_{P}, S)}$$

(12) Design CFG for the Language $L = \{a^n b^n c^i \mid n \geq 0, i \geq 1\}$
$\cup$
$\{a^n b^n c^m d^m \mid n, m \geq 0\}$

$a^n b^n$ is generated by $A \to aSb \mid \epsilon$
c^i —— " —— $C \to cC \mid \epsilon$

grammer for L is $= \{a^n b^n c^i \mid n \geq 0, i \geq 1\}$ is

$$\boxed{\begin{array}{l} S \to AC \\ A \to aSb \mid \epsilon \\ C \to cC \mid \epsilon \end{array}} \quad \text{—— (1)}$$

Now Consider $\{\underline{a^n b^n}\ \underline{c^m d^m} \mid n, m \geq 0\}$

$a^n b^n$ is generated by $\boxed{\begin{array}{l} A \to aAb \mid \epsilon \\ B \to cBd \mid \epsilon \end{array}}$ —— (2)
$c^m d^m$ —— " —— " ——
$(m \geq 0)$

The grammer for given Problem is:

$P = \{\underbrace{S \to AC,\ A \to aSb \mid \epsilon,\ C \to cC \mid \epsilon}_{\text{taken from (1)}},\ \underbrace{A \to aAb \mid \epsilon,\ B \to cBd \mid \epsilon}_{\text{taken from (2)}}\}$

$V = \{S, A, B\}$
$T = \{a, b, c, d\}$
$S = S$ (Start Variable)

(13) Design Context free Grammer for $L = \{a^n b^m c^k \mid k = m + n,\ n, m \geq 0\}$

<u>Solution</u>:- Consider $W = a^n b^m c^k$
Substitute $m+n$ for K
$= a^n b^m c^{m+n}$
$= a^n b^m c^m c^n$
$= a^n b^m c^n c^m$
$= a^n \underline{b^m c^m} c^n$

$a^n c^n$ is generated by $S \to aSc$
$b^m c^m$ is generated by $A \to bAc$
$a^n b^m c^m c^n$ is generated by : $S \to aSc \mid A$
$A \to bAc \mid \epsilon$

The Grammer is:-

$\therefore G = \{\underbrace{\{S, A\}}_{V}, \underbrace{\{a, b, c\}}_{T}, \underbrace{\{S \to aSc \mid A,\ A \to bAc \mid \epsilon\}}_{P}, S\}$

Q.14) Construct (i) Leftmost Derivation (LMD)
(ii) Rightmost Derivation (RMD)
(iii) Parse tree for the input string $+ * - xyxy$
Using the grammer $E \to +EE \mid *EE \mid -EE \mid x \mid y$

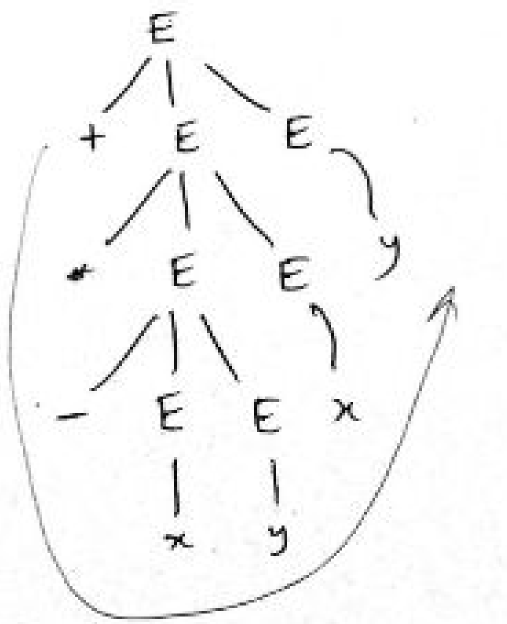
(i) LMD : $w = + * - xyxy$

$E \overset{lm}{\Longrightarrow} +EE \quad (E \to +EE)$
$\overset{lm}{\Longrightarrow} + * EEE \quad (E \to *EE)$
$\overset{lm}{\Longrightarrow} + * - EEEE \quad (E \to -EE)$
$\overset{lm}{\Longrightarrow} + * - xEEE \quad (E \to x)$
$\overset{lm}{\Longrightarrow} + * - xyEE \quad (E \to y)$
$\overset{lm}{\Longrightarrow} + * - xyxE \quad (E \to x)$
$\overset{lm}{\Longrightarrow} + * - xyxy \quad (E \to y)$

(ii) RMD :- $w = + * - xyxy$

$E \overset{rm}{\Longrightarrow} +EE \quad (\text{using } E \to +EE)$
$\overset{rm}{\Longrightarrow} +Ey \quad (\text{—"— } E \to y)$
$\overset{rm}{\Longrightarrow} + * EEy \quad (E \to *EE)$
$\overset{rm}{\Longrightarrow} + * Exy \quad (\text{using } E \to x)$
$\overset{rm}{\Longrightarrow} + * - EExy \quad (\text{using } E \to -EE)$
$\overset{rm}{\Longrightarrow} + * - Eyxy \quad (\text{using } E \to y)$
$\overset{rm}{\Longrightarrow} + * - xyxy \quad (\text{using } E \to y)$

(iii) Parse tree :



Yield $w = + * - xyxy$

(15) Design Grammer (CFG) for Valid arithmetic expressions over operators + and − . The arguments of the expressions are Valid identifiers over symbols a, b, 0, & 1.

Let I is a Variable that generates Set of identifers as strings and E generates set of strings that are expressions.

$$\boxed{\text{The grammer (G) is: } \begin{array}{l} E \to E+E \mid E*E \mid (E) \mid I \\ I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \end{array}}$$

$L(G) = \{a+b,\ a*b,\ a+b*a,\ b*a+b,\ a0+b1*b1,$
$ba1*ab0+aab\}$

(16) Show that the above grammer is ambigious. Find Un-ambigious grammer.

Ambigious grammer :- Definition

A CFG $G=(V,T,P,S)$ is ambigious if there exist atleast one string w in T^* for which we can find two different parse trees each with root labelled S and yield w.

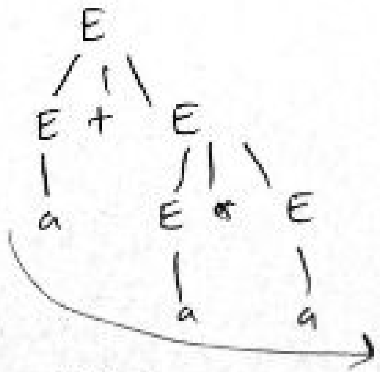
If each string w has atmost one parsetree then the grammer is Un-ambigious

Step ①: Derive Some string w from grammer

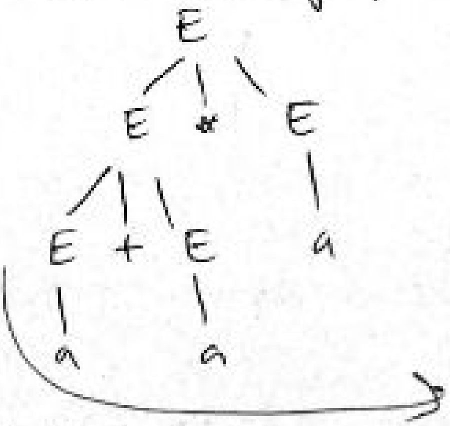
$$E \xRightarrow{lm} E+E \xRightarrow{lm} a+E \xRightarrow{lm} a+E*E \xRightarrow{lm} a+a*E \Rightarrow a+a*a$$

Step ②: Construct two different parse trees whose Yield is $w = a+a*a$

Start Variable (E) of G is root node of parse tree



Yield = a+a*a



Yield $w = a+a*a$

The un-ambigious grammer can be obtained as follows:
Generate two new Variables F (Factor) and T (Term).

(i) <u>Factor</u>:- is an expression that can not be broken by any adjacent operators. Identifier is a Factor. Expression within paranthesis is a factor.

$$F \to I \mid (E)$$

(ii) <u>Term</u> is an expression that cannot be broken by + but *. Term is a Factor and is also product of two Factors

$$T \to F \mid T * F$$

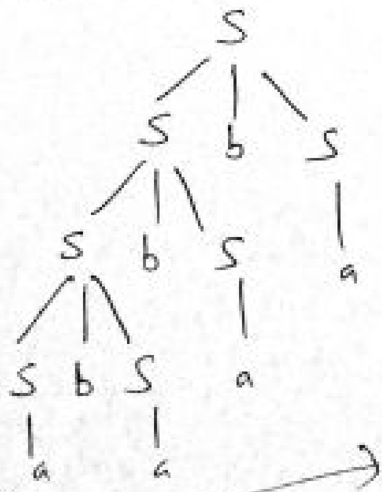
(iii) <u>Expression</u> is one that can be broken by any operator. Expression is Sum of two terms ~~and Product of two terms~~ or a term.

$$E \to E + T \mid T$$

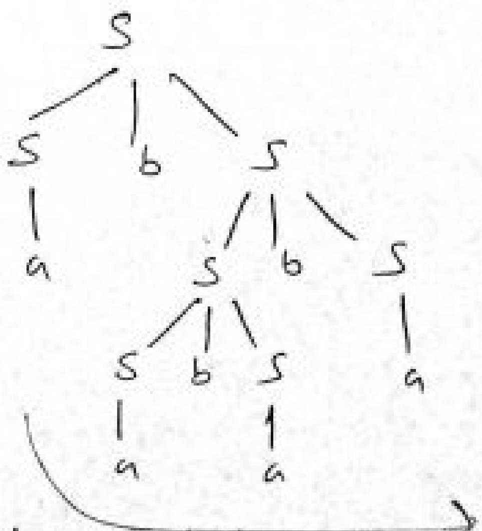
The resultant Un-ambigious grammer is:-

$$\boxed{\begin{array}{l} E \to E + T \mid T \\ T \to F \mid T * F \\ F \to I \mid (E) \end{array}}$$

(16) Consider the grammer $S \to SbS \mid a$. This grammer is ambigious. Show in particular that the string abababa has two (i) Parse trees (ii) LMD's (iii) RMD's



Yield W = abababa



Yield = abababa

(ii) LMD :- $w = abababa$

$S \xRightarrow{lm} \underline{S}bS$ (Using $S \to SbS$)

$\xRightarrow{lm} \underline{S}bSbS$ (-"- $S \to SbS$)

$\xRightarrow{lm} \underline{S}bSbSbS$ (Using $S \to SbS$)

$\xRightarrow{lm} ab\underline{S}bSbS$ (-"- $S \to a$)

$\xRightarrow{lm} abab\underline{S}bS$ (-"- $S \to a$)

$\xRightarrow{lm} ababab\underline{S}$ (-"- $S \to a$)

$\xRightarrow{lm} abababa$ (-"- $S \to a$)

(iii) RMD! $w = abababa$

$S \xRightarrow{rm} Sb\underline{S}$ (Using $S \to SbS$)

$\xRightarrow{rm} \underline{S}ba$ (Using $S \to a$)

$\xRightarrow{rm} Sb\underline{S}ba$ (-"- $S \to SbS$)

$\xRightarrow{rm} \underline{S}baba$ (-"- $S \to a$)

$\Longrightarrow Sb\underline{S}baba$ (-"- $S \to SbS$)

$\Longrightarrow \underline{S}bababa$ (-"- $S \to a$)

$\Longrightarrow abababa$ (Using $S \to a$)

(17) Construct (i) LMD (ii) RMD & (iii) parse tree for the string aabbaa using the grammer: $S \to AS \mid \epsilon$
$A \to aa \mid ab \mid ba \mid bb$

(i) LMD :- $w = aabbaa$

$S \xRightarrow{lm} \underline{A}S$ (Using $S \to AS$)

$\xRightarrow{lm} aa\underline{S}$ (-"- $A \to aa$)

$\xRightarrow{lm} aa\underline{A}S$ (-"- $S \to AS$)

$\xRightarrow{lm} aabb\underline{S}$ (-"- $A \to bb$)

$\xRightarrow{lm} aabb\underline{A}S$ (-"- $S \to AS$)

$\xRightarrow{lm} aabbaa\cancel{S}$ (Using $S \to \epsilon$)

(ii) RMD :- $w = aabbaa$

$S \xRightarrow{rm} A\underline{S}$ (Using $S \to AS$)

$\xRightarrow{rm} AA\underline{S}$ (Using $S \to AS$)

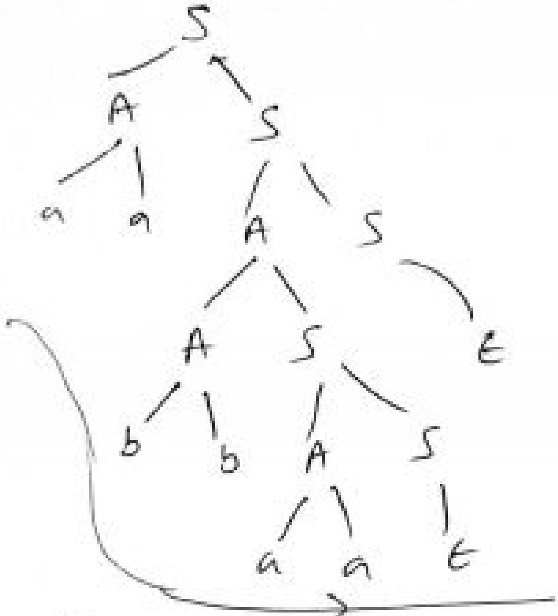
$\xRightarrow{rm} AAA\underline{S}$ (-"- $S \to AS$)

$\xRightarrow{rm} AA\underline{A}$ (Using $S \to \epsilon$)

$\xRightarrow{rm} A\underline{A}aa$ (Using $A \to aa$)
$\xRightarrow{rm} \underline{A}bbaa$ (—"— $A \to bb$)
$\xRightarrow{rm} \underline{aabbaa}$ (—"— $A \to aa$)

(iii) <u>Parse tree</u>

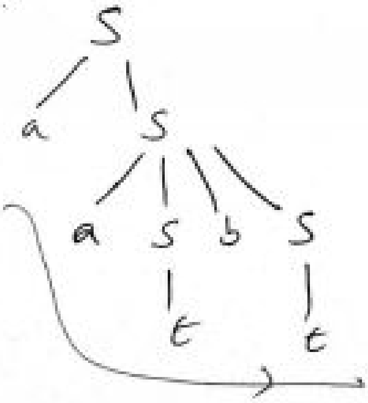
$w = aabbaa$



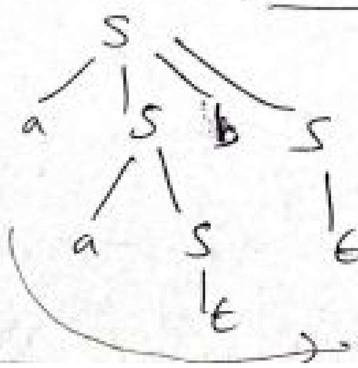
Yield $\underline{w = aabbaa}$

(18) Prove that the following grammer is ambigious. $S \to aS \mid aSbS \mid \epsilon$. Show in particular that the string <u>'aab'</u> <u>has two</u> (i) LMD's (ii) RMD's (iii) Parse trees.

(i) <u>Parse tree</u> :- $w = aab$



Yield $\underline{w = aab}$



Yield = $\underline{aab}$

(ii) <u>LMD</u>:
$S \xRightarrow{lm} a\underline{S}bS$ (Using $S \to aSbS$)
$\xRightarrow{lm} a\,aSbS$ (Using $S \to aS$)
$\Rightarrow aa\,bS$ (—"— $S \to \epsilon$)
$\Rightarrow aab$ (—"— $S \to \epsilon$)

(ii) <u>LMD</u>:
$S \xRightarrow{lm} a\underline{S}$ (Using $S \to aS$)
$\xRightarrow{lm} aa\underline{S}bS$ (Using $S \to aSbS$)
$\xRightarrow{lm} aab\underline{S}$ ($S \to \epsilon$)
$\xRightarrow{lm} \underline{aab}$ (Using $S \to \epsilon$)

(iii) **RMD's:**

$w = aab$

$S \xRightarrow{rm} aSbS$ (Using $S \to aSbS$)

$\xRightarrow{rm} aSb$ (Using $S \to \epsilon$)

$\xRightarrow{rm} aaSb$ ($S \to aS$)

$\xRightarrow{rm} aab$ ($S \to \epsilon$)

(iv) **RMD:**

$S \xRightarrow{rm} aS$ (Using $S \to aS$)

$\xRightarrow{rm} aaSbS$ (Using $S \to aSbS$)

$\xRightarrow{rm} aabS$ (—"— $S \to \epsilon$)

$\xRightarrow{rm} aab$ (—"— $S \to \epsilon$)

Applications of Context Free Grammers:-

① Context Free Grammers are useful in the development of Parsers of Compiler.

Parser is a module of Compiler that takes stream of tokens as input and groups tokens into expressions and statements as per the grammer.

② Grammers are also used in the design of Parsers for XML (Extensible Mark-Up languages).

③ grammers are used in YACC programs.

⑱ Write (i) Leftmost Derivation (ii) Parse tree for the String $w = 0-((1*0)-0)$ Using the grammer

$E \to E+T \mid T \quad T \to F-T \mid F \quad F \to (E) \mid 0 \mid 1$

(i) **LMD:-** $E \xRightarrow{lm} F-T \xRightarrow{lm} 0-T \xRightarrow{lm} 0-F \xRightarrow{lm} 0-(E)$

$\xRightarrow{lm} 0-(F-T) \Rightarrow 0-((E)-T) \Rightarrow 0-((E*T)-T)$

$\xRightarrow{lm} 0-((T*T)-T) \xRightarrow{lm} 0-((F*T)-T)$

$\xRightarrow{lm} 0-((1*F)-T) \xRightarrow{lm} 0-((1*0)-T)$

$\xRightarrow{lm} 0-((1*0)-F)$

$\xRightarrow{lm} 0-((1*0)-0)$