Segment 1

CFG → Context Free Gramer

A CGF is a set of rule / productions used to generate pattern of strings.

→ set of recursive rules

→ used to describe CFG languages

→ can describe all regular languages but not all possible "

Formal Defination : 4 tuple (V, Σ, R, S) ; N ; T ; P

i) V → Variable (N → Non terminal)

ii) Σ → finite set disjoint from V (T → terminals)

iii) R → rules (P → Production)

iv) S → start variable

AU22

Any components have a similarity with any of the components of regular language or finite automata.

→ Production rules in CFGs are somewhat similar to the transition function in Finite automata, as both defines how symbols can be combined or changed.

→ Terminals in CFGs are similar to the Alphabet in Regular languages and Finite Automata. Both represent the basic symbols used to form strings.

* L = { strings contain 0 and 1 }

= { 01, 001, 011, 0011, 0111, 00011 ... }

Production rule

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

n = 1 , $0^1 1^1 = 01$

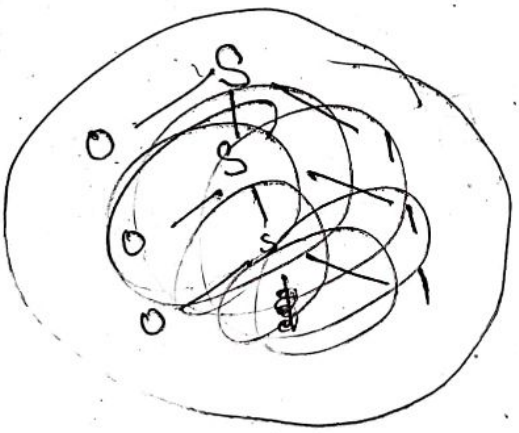
n = 2 , $0^2 1^2 = 0011$

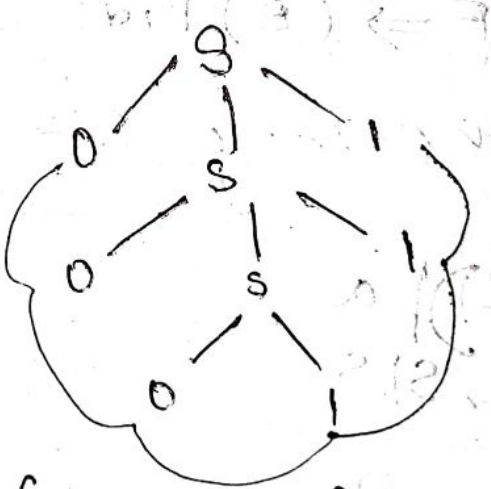
n = 3 , $0^3 1^3 = 000111$

S → 0S1 or S → 01 | 0S1

S → 0S1

pattern ⇒ 000111





parse tree (000111)

Write formal defination / Identify terminals, non terminals, start variable.

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

Variable (Non terminal) $\rightarrow \{E, T, F\}$

Σ (Terminals) $\rightarrow \{'+', '*', '(', ')', id\}$

Rules: $E \Rightarrow E+T \mid T$

$T \Rightarrow T*F \mid F$

$F \Rightarrow (E) \mid id$

START VARIABLE $S = \{E\}$

$S \Rightarrow (L) \mid a$

$L \Rightarrow L, S \mid S$

$V = \{S, L\}$

$\Sigma = \{'(', ')', ','\}$

$S = \{S\}$

STRING ACCEPTANCE

Derivation Format: The process of deriving a string from given grammer is called as derivation.

Parse Tree: The geomatrical represent ation of derivation is known as derivation tree or parsing tree.

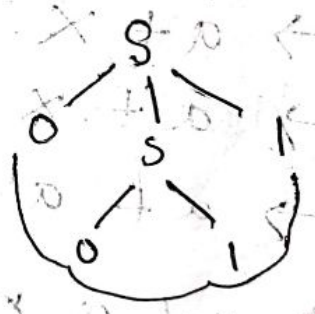
- Check if the following grammer accept the string 0011 is or not.

$S \rightarrow 01 \mid 0S1$

$S \rightarrow 0S1$

$\rightarrow 00\underline{11}$

Derivation Format



Parsing format

Differences	Derivation is a linear sequence of steps, while a parse tree is a hierarchical tree structure.	A parse tree visually represents the structure of the derivation, showing the hierarchical relationship between symbols.

Types of Parse Tree or Derivation

1) **Left most derivation:** A left most derivation is obtained by applying production to the leftmost variable in each step.

Ex: $X \rightarrow X+X \mid X*X \mid a$ over an alphabet $\{a\}$

String = "a+a*a"

$X \rightarrow X+X$

$X \rightarrow a+X$

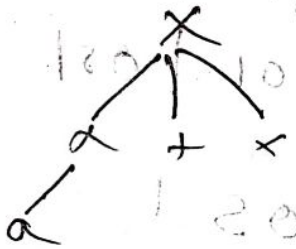
$X \rightarrow a+X*X$

$X \rightarrow a+a*X$

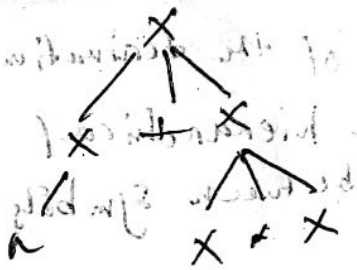
$X \rightarrow a+a*a$

Step-1

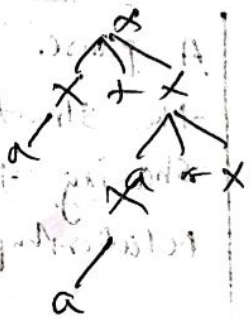
Step 2



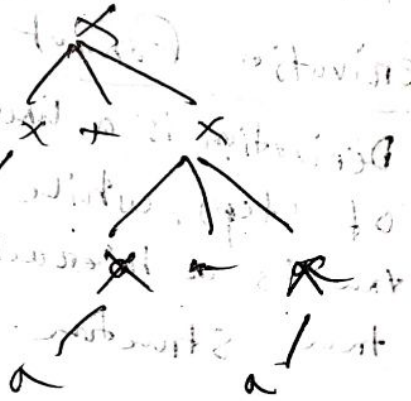
Step-3



Step-4



Step 5



Right most Derivation

$X \to X+X \mid X*X \mid X \mid a$ over alphabet $\{a\}$

String $a+a*a$

$X \to X*X$

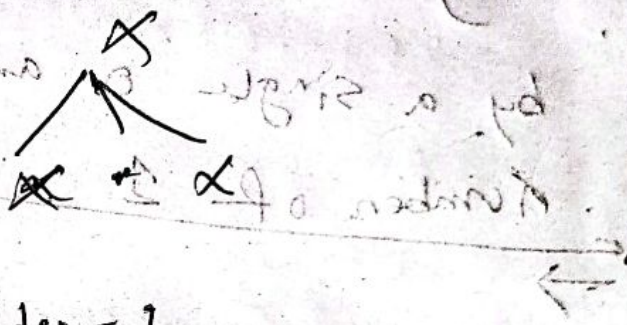
$X \to X*a$

$X \to X+X*a$

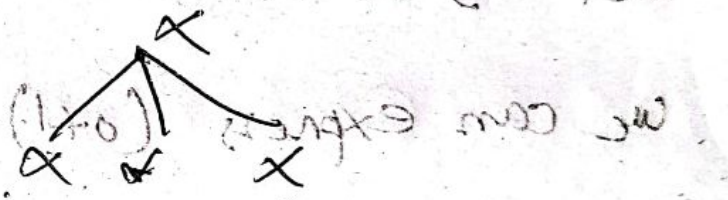
$X \to X+a*a$

$X \to a+a*a$

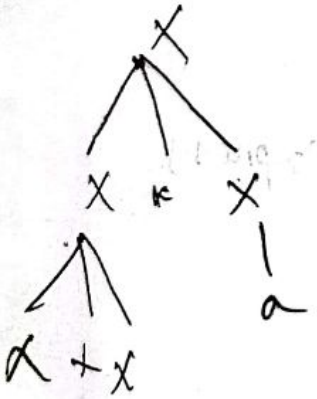
Step-1



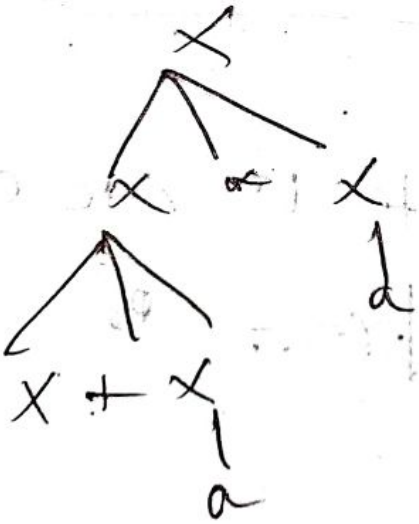
Step-2



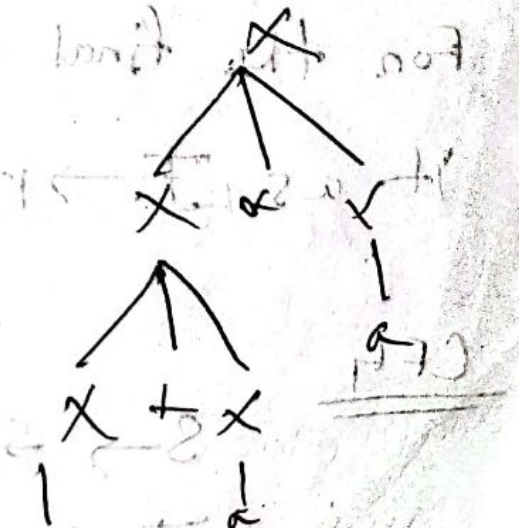
Step-3



Step-4



Step-5



Regular expression to CFG

AU-22

Construct CFG for $(0+1)^*01^*$

any combination of 0 and 1 followed by a single 0 and ending with any number of 1

→

Let S be ~~the~~ Start symbol

We can express $(0+1)^*$ as $S \rightarrow \underbrace{S0 \mid S1 \mid \epsilon}_{A}$.

~~then we need a single 0 so we add rule~~

~~$S \rightarrow T0$~~

For the final part 1^* we can express it as ~~$T \rightarrow 1T \mid \epsilon$~~. B

CFG

$S \rightarrow$ ~~$S0 \mid S1 \mid T0$~~ $A\,0\,B$

~~$T \rightarrow 1T \mid \epsilon$~~

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

$B \rightarrow 1B \mid \epsilon$

0* 1 (0+1)* any number of 0 followed by a single 1 ending with any combination of 0 and 1

$L = \{0, 01, 1010101, 001, 100001, \ldots\}$

$S \rightarrow A|B$

$A \rightarrow 0A \mid \epsilon$

$B \rightarrow 0B \mid 1B \mid \epsilon$

AU-22 1-b

$S \rightarrow aAS/a$

$A \rightarrow SbA/SS/ba$

Generate $S \rightarrow aabbaa$ by rightmost derivation.

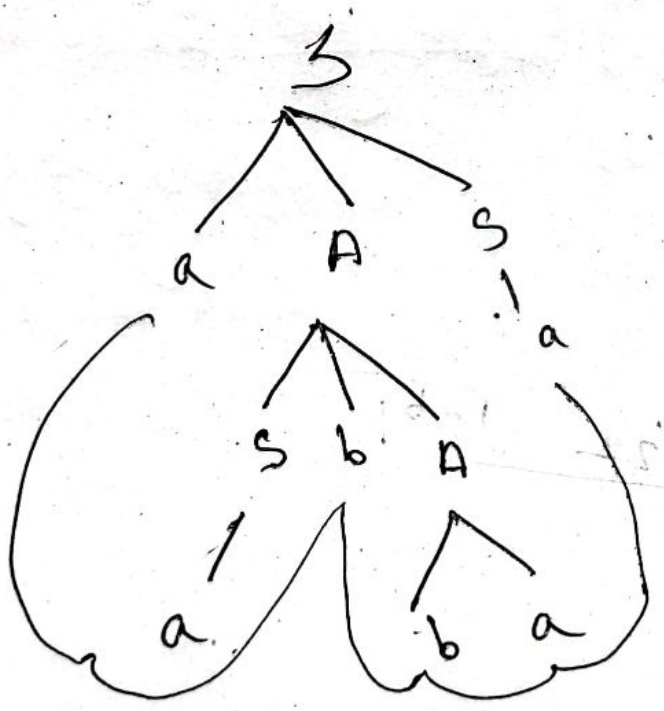
$S \rightarrow aAS$ $S \rightarrow aAS$

$S \rightarrow aAa$ $A \rightarrow SbA$ a

$S \rightarrow aSbAa$ $A \rightarrow ba$

$S \rightarrow aSbbaa$ $S \rightarrow a$

$S \rightarrow aabbaa$



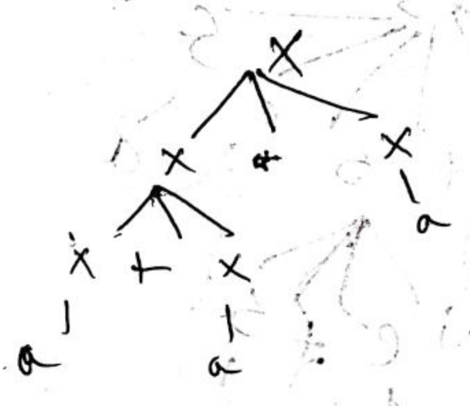
5 marks

Ambiguity: If a grammer generates the same string in several ways, we say that the string is derived ambigously in that grammer. If a grammer generates some string ambiguosly, we say that the grammer is ambiguos.

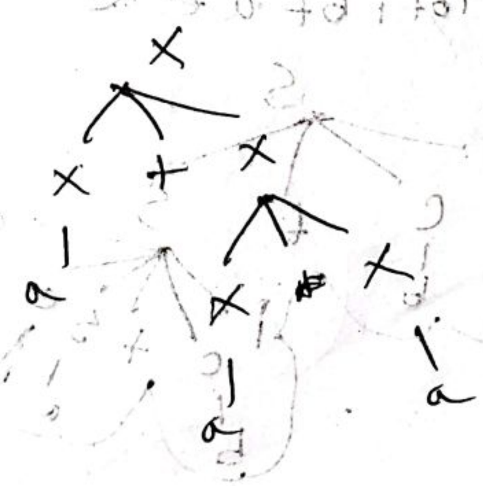
CFG তে যদি একাধিক RMD ও LMD এর অথবা parsing tree হয় same string এর জন্য তাহলে amgious বলে। যেমন একাধিক RMD ও LMD থাকতে হবে।

$x \to x+x \mid x*x \mid x \mid a$ $(a+a*a)$

Right most derivation



Left most derivation



(c) 5 marks

$S \rightarrow iCtS / iCtSeS / a$

$C \rightarrow b$

Show that string **ibtibtaea** has two parse trees, RMD, LMD

i) 2 → Parse trees

$S \rightarrow iCtS$

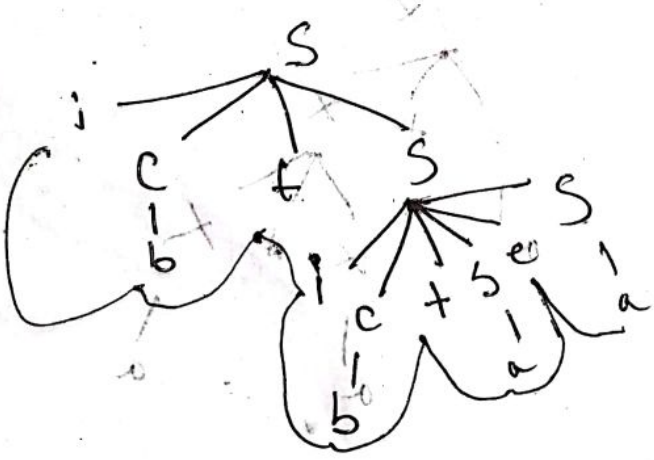
$S \rightarrow ibtS$

$S \rightarrow ibtiCtSeS$

$S \rightarrow ibtibtSeS$

$S \rightarrow ibtibtaeS$

$S \rightarrow ibtibtaea$



$S \rightarrow iCtSeS$

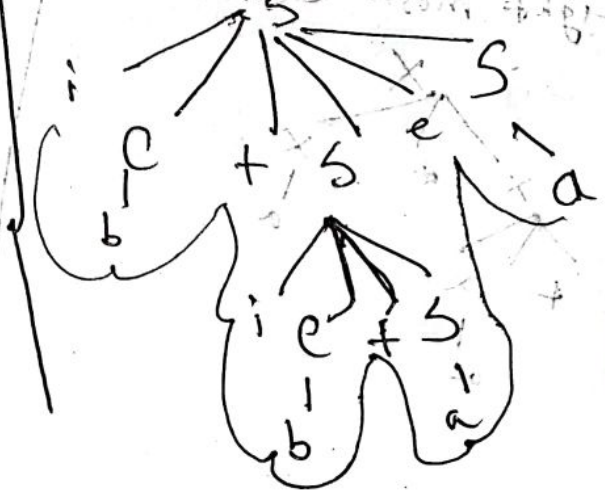
$S \rightarrow ibtSeS$

$S \rightarrow ibtiCtSeS$

$S \rightarrow ibtibtSeS$

$S \rightarrow ibtibtaeS$

$S \rightarrow ibtibtaea$



ii) two Rightmost derivations ibtibtaeaa.

$S \to ietSeS$

$S \to ietSea$

$S \to ictictSea$

$S \to ictictaeaa$

$S \to ictibtaeaa$

$S \to ibtibtaeaa$

$S \to ietS$

~~$S \to ietSeS$~~

~~$S \to ietictSeSeS$~~

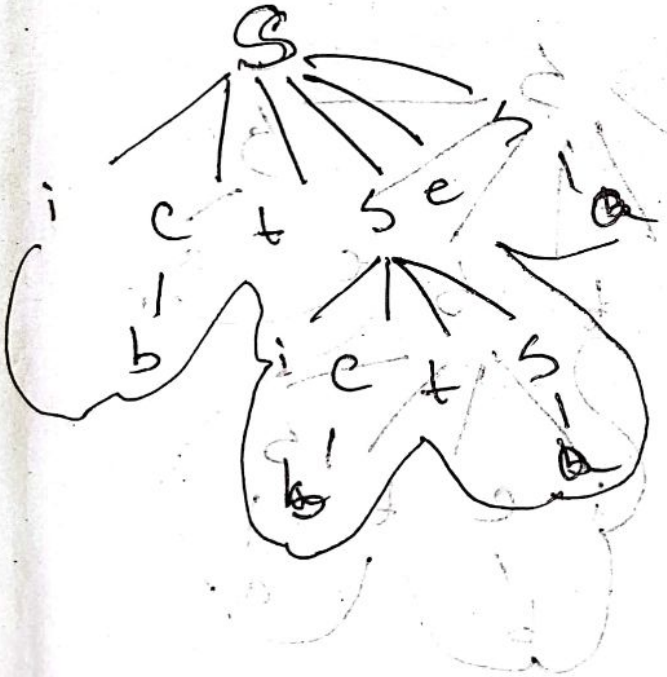
$S \to ictictSeS$

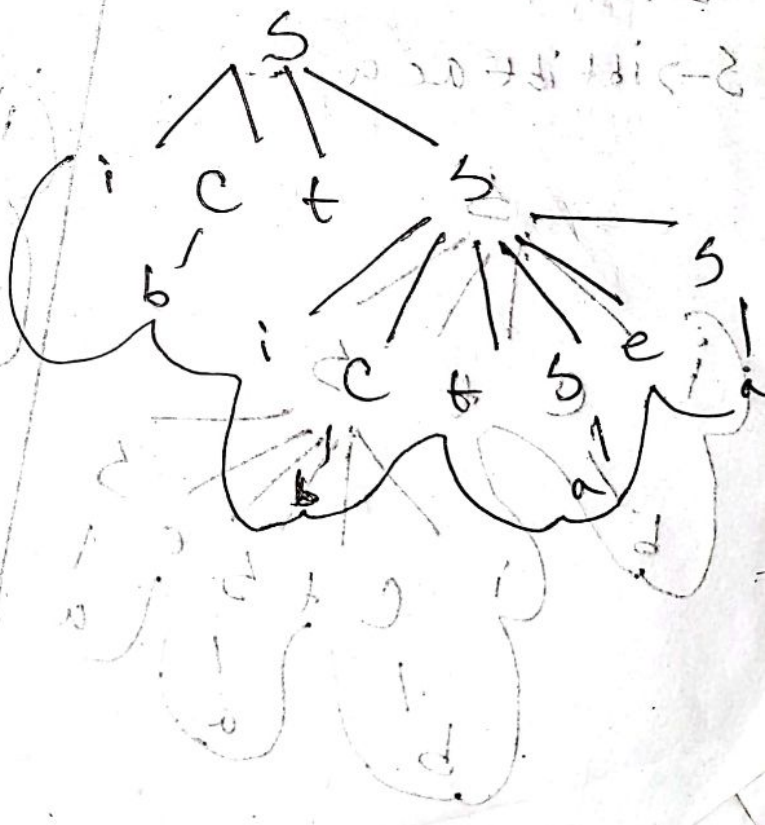
$S \to ictictSea$

$S \to ictictaeaa$

$S \to ictibtaeaa$

$S \to ibtibtaeaa$





two Leftmost Derivation

$S \rightarrow iCtS$

~~$S \rightarrow iCtSeS$~~

~~$S \rightarrow ibt$~~

$S \rightarrow ibtS$

$S \rightarrow ibtiCtSeS$

$S \rightarrow ibtibtSeS$

$S \rightarrow ibtibtaeS$

$S \rightarrow ibtibtaea$

$S \rightarrow iCtSeS$

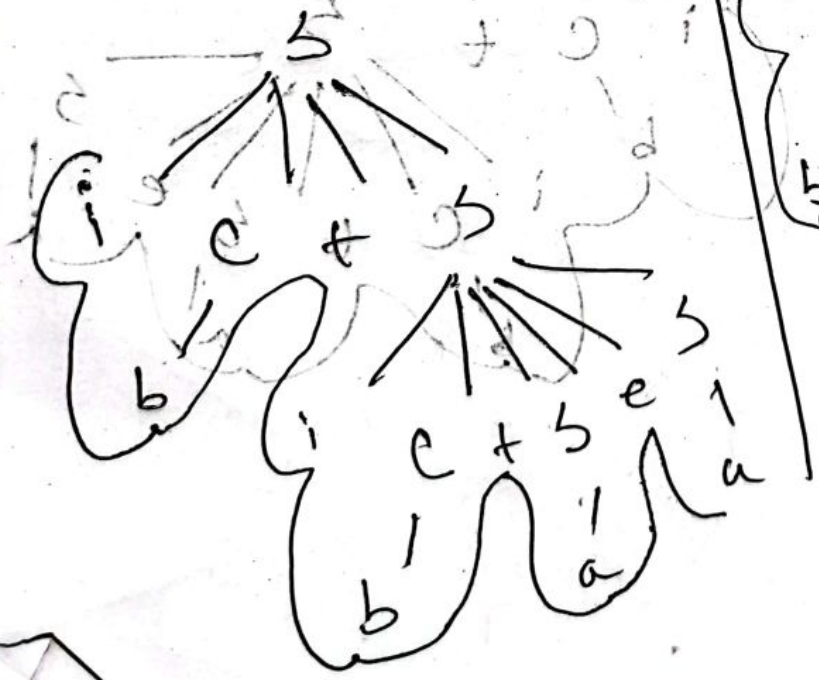
$S \rightarrow ibtSeS$

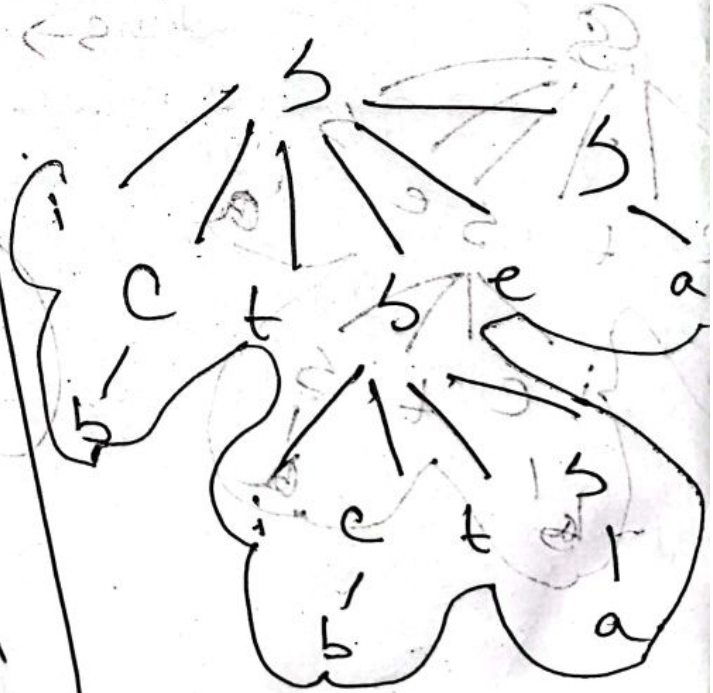
$S \rightarrow ibtiCtSeS$

$S \rightarrow ibtibtSeS$

$S \rightarrow ibtibtaeS$

$S \rightarrow ibtibtaea$





(c)

$S \rightarrow SSx \mid SSy \mid SSz \mid a \mid b \mid C$ Derive cabxyz

L.m.D

$S \rightarrow SSz$

$S \rightarrow CSz$

$S \rightarrow$

The given grammer can not generate the following string

b

Check ambiguity

$S \rightarrow AB$

$A \rightarrow aA \mid abA \mid \epsilon$

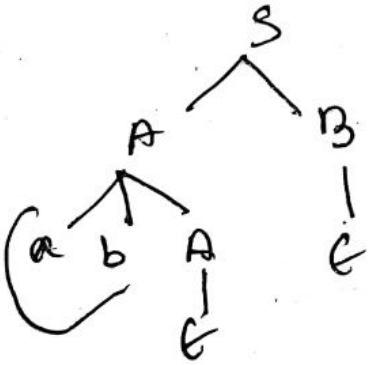
$B \rightarrow bB \mid abB \mid \epsilon$

Lets derive string ab

$S \rightarrow AB$

$S \rightarrow abAB$

$S \rightarrow ab$

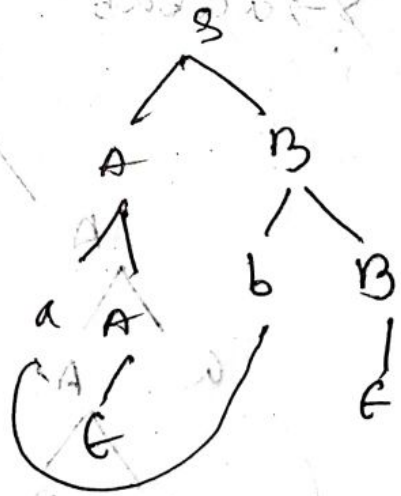


$S \rightarrow AB$

$S \rightarrow aAB$

$S \rightarrow aAbB$

$S \rightarrow ab$



Yes it is ambigious -

Two different parsing trees.

AU-21

1) b)

S → AB

A → aA | abA | ε

B → bB | abB | ε

c) derive " aababab " using L.M.D

S → AB

S → aAB

S → aaAB

S → aaB

S → aabB

S → aabababB

S → aababab

S → AB

S → aAB

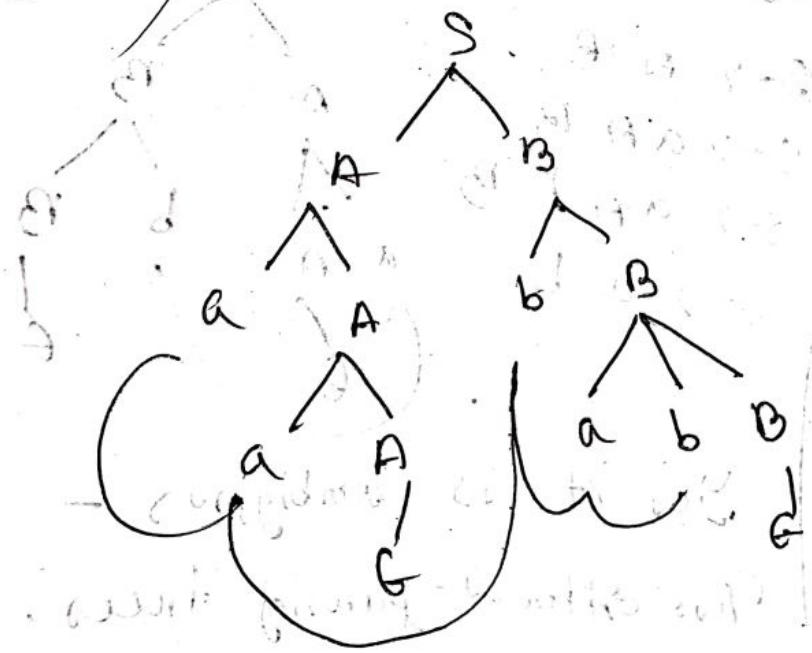
S → aaAB

S → aaB

S → aabB

S → aababB

S → aababab



derive "abaabb" using R.M.D

$S \rightarrow AB$

$S \rightarrow AbB$

$S \rightarrow AbbB$

$S \rightarrow Abb$

$S \rightarrow abAbb$

$S \rightarrow abaAbb$

$S \rightarrow abaaAbb$

$S \rightarrow abaabb$