Chapter 5

CFGs

CFG:

A context free grammar is a notation for describing language.

Applications:

1. Played central role in compiler technology since 1960's.
2. It is used to describe document formats.
3. Useful for nested structure.
4. It is more powerful than FA and RE.

 # components of CFG:

— 4 (four) components of CFG.

$G_1 = (V, T, P, S)$ → Start variable

- V → Variable / non-terminal
- T → set of terminals
- P → Production / rules

① $E \Rightarrow E+T \mid T$

$T \Rightarrow T * F \mid F$

$F \Rightarrow (E) \mid id$

Here, $V = \{E, T, F\}$

$T = \{+, *, (,), id\}$

$P =$ $E \Rightarrow E+T \mid T$

$T \Rightarrow T * F \mid F$

$F \Rightarrow (E) \mid id$

$S = \{E\}$

Palindrome :

A Palindrome is a word that reads same from forward and backward such as otto, madam etc.

Properties :

1. It must be begin with and end with same symbol.
2. when 1st & last symbol are removed the resulting string must also be Palindrome.

CFG for Palindrome :

1. $P \Rightarrow \epsilon$
2. $P \Rightarrow 0$
3. $P \Rightarrow 1$
4. $P \Rightarrow 0P0$
5. $P \Rightarrow 1P1$

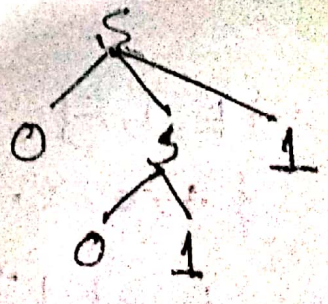
CFG for $(0^n 1^n \mid n \geq 1)$

Production :

$S \Rightarrow 01$

$S \Rightarrow 0S1$

Parsing → 0011



Two types of derivation:

In each step we replace the leftmost variable with one of its production bodies.

Ex:

1. $S \to AB$
2. $A \to aaA$
3. $A \to \lambda$
4. $B \to Bb$
5. $B \to \lambda$

Leftmost derivation order of string: aab

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

Rightmost derivation:

In each step we replace the rightmost variable with one of its Production bodies.

Rightmost derivation order of string: aab

$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{5}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

LMD & RMD

CFG: $E \to I \mid E+E \mid E*E \mid (E)$

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

a*(a+b00):

LMD-

$E \Rightarrow E*E$

$\Rightarrow I*E$

$\Rightarrow a*E$

$\Rightarrow a*(E)$

$\Rightarrow a*(E+E)$

$\Rightarrow a*(I+E)$

$\Rightarrow a*(a+E)$

$\Rightarrow a*(a+I)$

$\Rightarrow a*(a+I0)$

$\Rightarrow a*(a+I00)$

$\Rightarrow a*(a+b00)$

a*(a+b00):

RMD-

$E \Rightarrow E*E$

$\Rightarrow E*(E)$

$\Rightarrow E*(E+E)$

$\Rightarrow E*(E+I)$

$\Rightarrow E*(E+I0)$

$\Rightarrow E*(E+I00)$

$\Rightarrow E*(E+b00)$

$\Rightarrow E*(Ia+b00)$

$\Rightarrow E*(a+b00)$

$\Rightarrow I*(a+b00)$

$\Rightarrow a*(a+b00)$

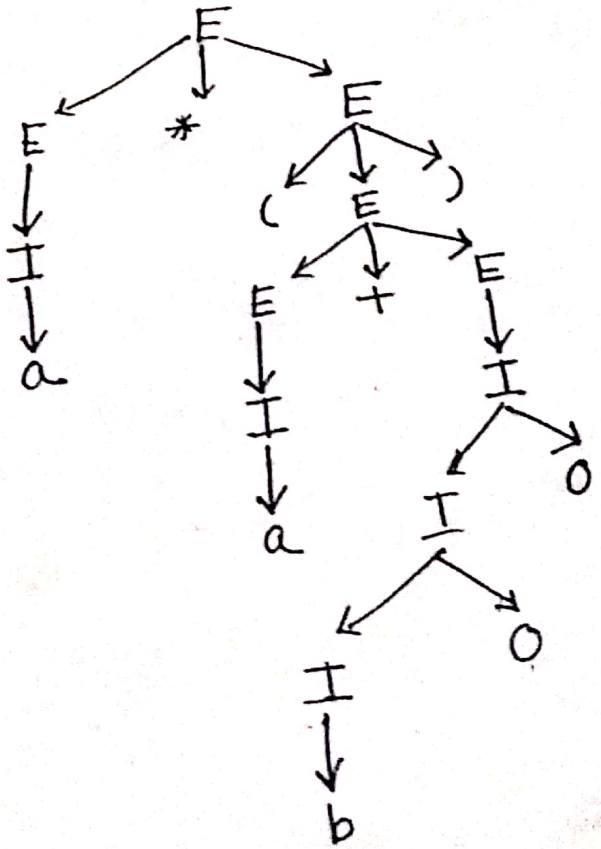
Parse tree:

Graphical representation for any derivation is called parse tree.

condition:

- Each internal node is labled by a variable V.
- Each leaf node is labled by variable terminals or ∈.
- If an interior node is labeled A, & its children are labeled $X_1, X_2 \cdots X_n$ respectively from the left then $A \to X_1 X_2 \cdots X_n$ is a production.

Parse tree for showing a* (a+b00) → LMD:



Question — 2018

A(G): $S \to A1B$
$A \to 0A \mid \epsilon$
$B \to 0B \mid 1B \mid \epsilon$

LMD — 00101 :

$S \Rightarrow A1B$
$\Rightarrow 0A1B$
$\Rightarrow 00A1B$
$\Rightarrow 001B$
$\Rightarrow 0010B$
$\Rightarrow 00101B$
$\Rightarrow 00101$

RMD — 00101 :

$S \Rightarrow A1B$
$\Rightarrow A10B$
$\Rightarrow A101B$
$\Rightarrow A101$
$\Rightarrow 0A101$
$\Rightarrow 00A101$
$\Rightarrow 00101$

Q-2018 :

4(b): $E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid a$

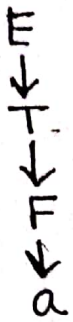
Give Parse tree & derivation for —

i) a :

derivation :

$E \Rightarrow T \Rightarrow F \Rightarrow a$

Parse tree



ii) a+a :

derivation :

$E \Rightarrow E+T$

$\Rightarrow T+T$

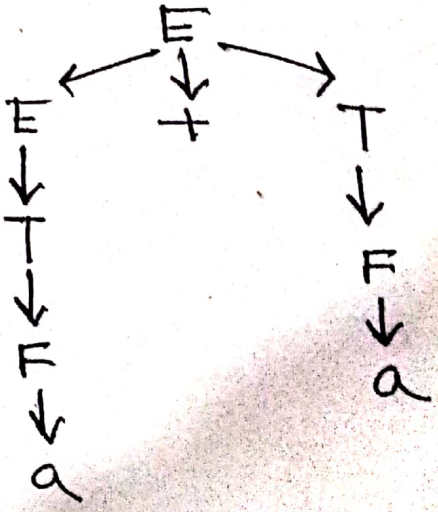
$\Rightarrow F+T$

$\Rightarrow a+T$

$\Rightarrow a+F$

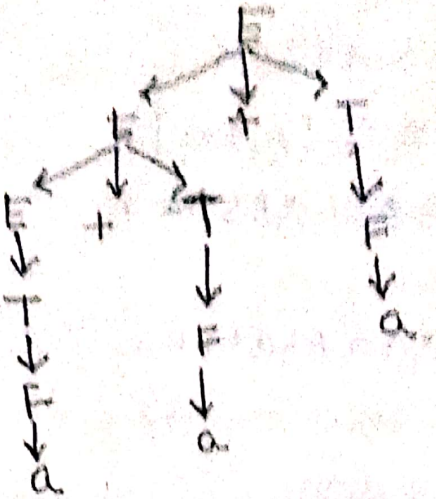
$\Rightarrow a+a$

Parse tree

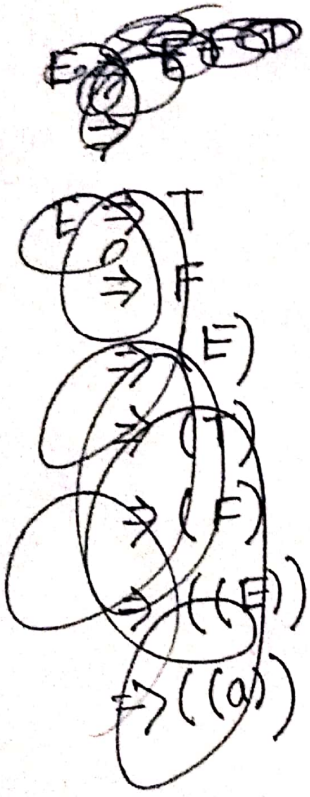


iii) a+a+a :

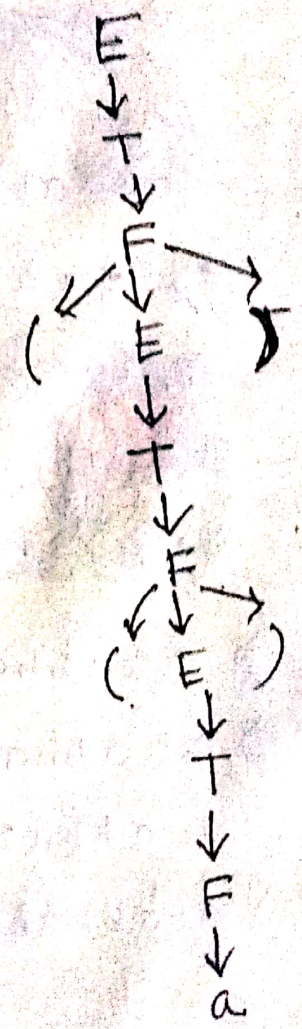
$E \Rightarrow E+T$
$\Rightarrow E+T+T$
$\Rightarrow T+T+T$
$\Rightarrow F+T+T$
$\Rightarrow a+T+T$
$\Rightarrow a+F+T$
$\Rightarrow a+a+T$
$\Rightarrow a+a+F$
$\Rightarrow a+a+a$



iv) ((a))



$E \Rightarrow T$
$\Rightarrow F$
$\Rightarrow (E)$
$\Rightarrow (T)$
$\Rightarrow (F)$
$\Rightarrow ((E))$
$\Rightarrow ((T))$
$\Rightarrow ((F))$
$\Rightarrow ((a))$



Q-2016 :

1(c)

$S \rightarrow aB \mid bA$

$A \rightarrow aS \mid bAA \mid a$

$B \rightarrow bS \mid aBB \mid b$

String — aaabbabbba

① LMD :

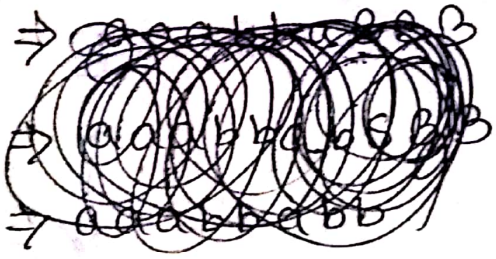
$S \Rightarrow aB$

$\Rightarrow aaBB$

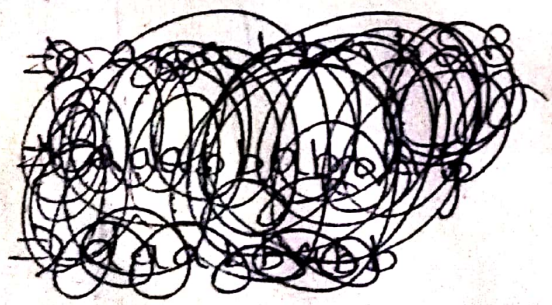
$\Rightarrow aaaBBB$

$\Rightarrow aaabSBB$

$\Rightarrow aaabbABB$

$\Rightarrow$ 

$\Rightarrow aaabbaBB$



$\Rightarrow aaabbabB$

$\Rightarrow aaabbabbS$

$\Rightarrow aaabbabbbA$

$\Rightarrow aaabbabbba$

i) RHD :

$S \Rightarrow aB$

$\Rightarrow aaBB$

$\Rightarrow aaBaBB$

$\Rightarrow aaBaBbS$

$\Rightarrow aaBaBbbA$

$\Rightarrow aaBaBbba$

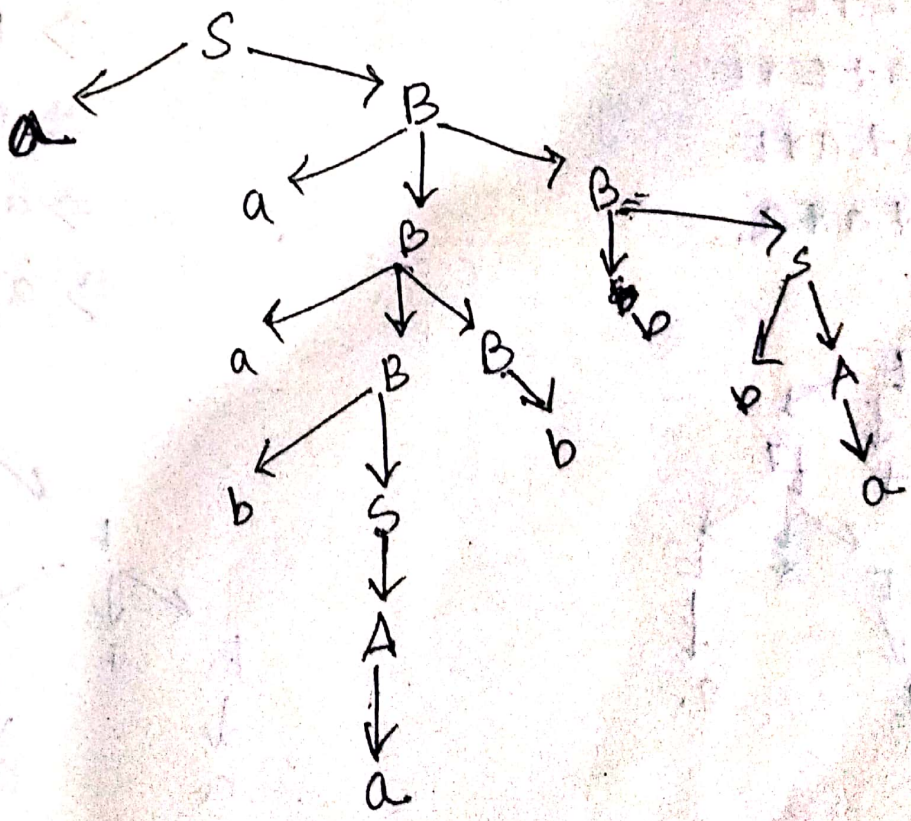
$\Rightarrow aaBabbba$

$\Rightarrow aaaBBabbba$

$\Rightarrow aaaBbabbba$

$\Rightarrow aaabbabbba$

ii) tree :



Ambiguity :

When a grammar fails to provide unique structure for each string in its language. this is called ambiguous grammer.

It happens due to —

(i) ~~Breech~~ Precedence of operator is neglected.

(ii) A sequence of identical operator can group either from left or from right.

Given, $E \to E+E \mid E*E \mid (E) \mid a$

a+a*a

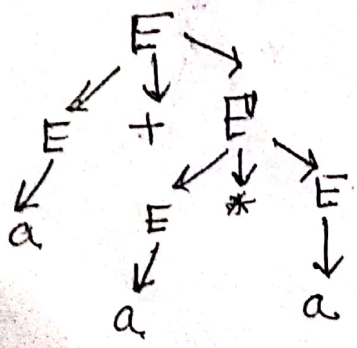
Now,

$E \Rightarrow E+E$

$\Rightarrow a+E*E$

$\Rightarrow a+a*E$

$\Rightarrow a+a*a$



a+a*a

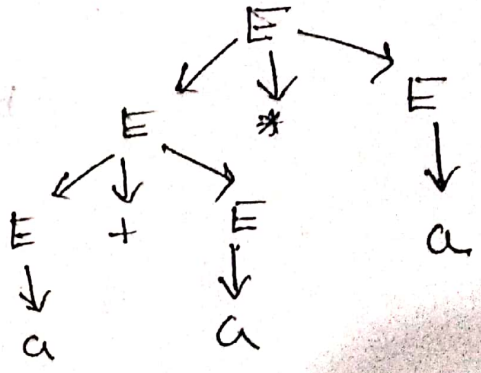
$E \Rightarrow E*E$

$\Rightarrow E+E*E$

$\Rightarrow a+E*E$

$\Rightarrow a+a*E$

$\Rightarrow a+a*a$



We see that this grammer fails to provide unique structure for language & we get two different derivation tree. so, this grammar is ambiguous.

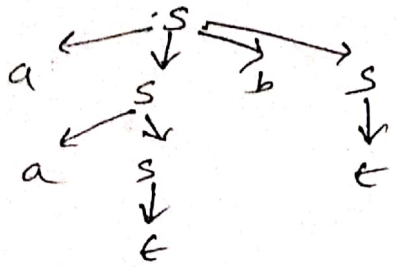
6 - 2018 :

$S \rightarrow aS \mid asbS \mid \epsilon$

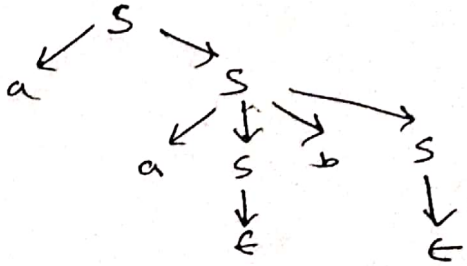
aab (string) :

LMD :

i) $S \Rightarrow asbS$
$\Rightarrow aasbS$
~~$\Rightarrow aasbasbS$~~ $\Rightarrow aaabS$
~~$\Rightarrow aaabasbS$~~
$\Rightarrow aaab$

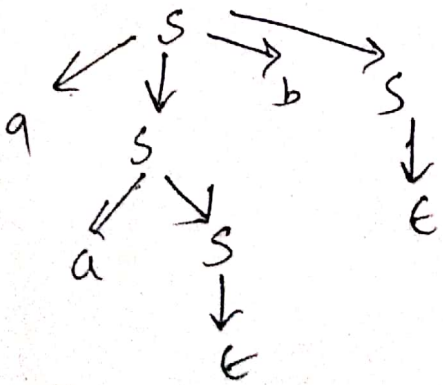


ii) $S \Rightarrow aS$
$\Rightarrow aaSbS$
$\Rightarrow aaabS$
~~$\Rightarrow aaabasbS$~~
$\Rightarrow aaab$



RMD :

$S \Rightarrow asbS$
~~$\Rightarrow asbasbS$~~
~~$\Rightarrow asbaasbS$~~
$\Rightarrow asb$
$\Rightarrow aasb$
$\Rightarrow aaab$



$S \Rightarrow aS$
$\Rightarrow aasbS$
$\Rightarrow aabS$
$\Rightarrow aaab$