# Ambiguity in Grammars & Language:

## Ambiguous Grammar:

A grammar of a language is called ambiguous if there are two or more then two different Parse tree generated for atleast one string of Language.

or

There are more then one LMD or more then one RMD for atleast one string. then the grammar is called Ambiguous.

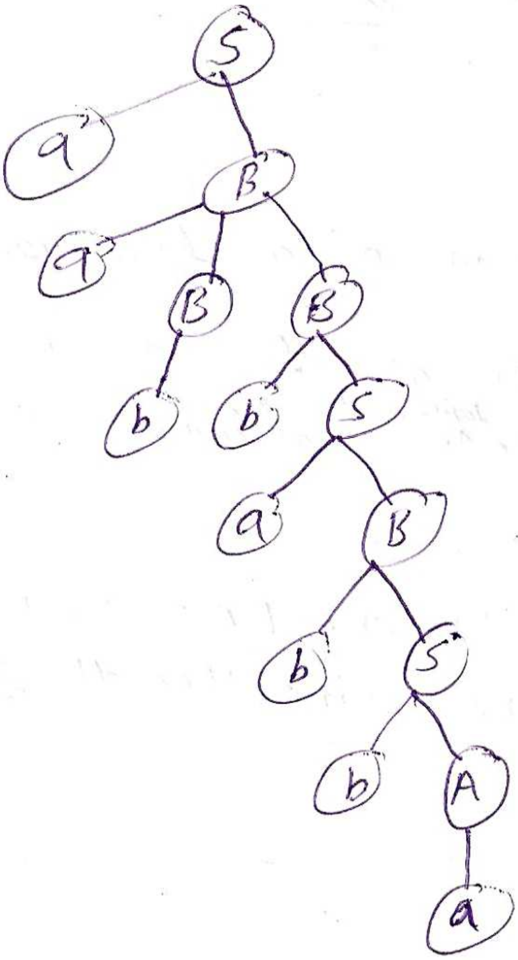
ex.

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

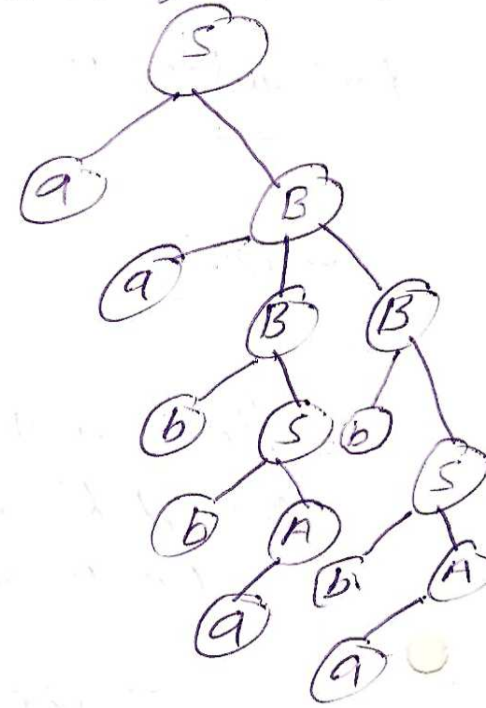
and generate a string "aabbabba"

| LMD(1): | LMD(2) |
|---|---|
| $S \rightarrow aB$ | $S \rightarrow aB$ |
| $\rightarrow aaBB$ | $\rightarrow aaBB$ |
| $\rightarrow aabB$ | $\rightarrow aabSB$ |
| $\rightarrow aabbS$ | $\rightarrow aabbAB$ |
| $\rightarrow aabbaB$ | $\rightarrow aabbaB$ |
| $\rightarrow aabbabS$ | $\rightarrow aabbabS$ |
| $\rightarrow aabbabbA$ | $\rightarrow aabbabbA$ |
| $\rightarrow aabbabba$ | $\rightarrow aabbabba$ |

Parse Tree for LMD(1)



Parse tree for LMD(2):



There are two different Parse tree for a string "aabbabba" so the given grammar is ambiguous grammar.

Q. show that the given grammar is ambiguous.

$S \to aSb \mid SS \mid \varepsilon$

check for a string "aabb"

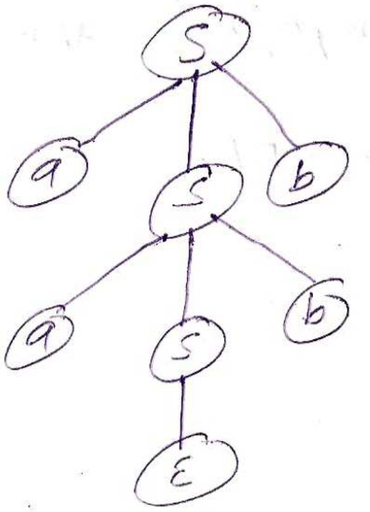
LMD(1):

$S \to aSb$
$\to aaSbb$
$\to aa\varepsilon bb$
$\to aabb$

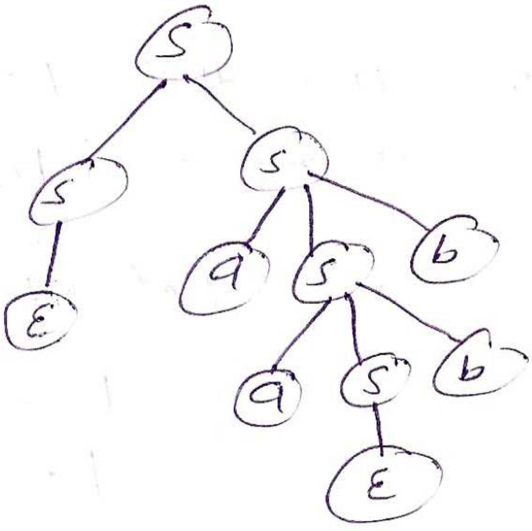
LMD(2):

$S \to SS$
$\to \varepsilon S$
$\to aSb$
$\to aaSbb$
$\to aa\varepsilon bb$
$\to aabb$

Parse tree for LMD(1)




Parse tree for LMD(2)




There are two different parse tree for a string "aabb". so the given grammar is ambiguous grammar.

Note: A Grammar is called unambiguous only if there exist a unique parse tree for every string of its language.

ex. $S \rightarrow aSb \mid \varepsilon$

<u>Inherently ambiguous:</u>

if L is a context free language for which there exists an unambiguous grammar, then language L is said to be unambiguous language.

if every grammar that generates L is ambiguous, then the language is called inherently ambiguous.

ex. show that $L = (a^n b^n c^m) \cup (a^n b^m c^m)$ $\forall n, m \geq 0$ is ~~an~~ an inherently ambigous CFL.

Let $L = L_1 \cup L_2$

$L_1 = \{a^n b^n c^m\}$

$L_2 = a^n b^m c^m\}$

write CFG for $L_1$

$$S_1 \to S_1 c / A$$
$$A \to aAb / \varepsilon$$

write CFG for $L_2$

$$S_2 \to aS_2 / B$$
$$B \to bBc / \varepsilon$$

Now with the help of CFG of $L_1$ & $L_2$ write CFG for L

$$S \to S_1 / S_2$$
$$S_1 \to S_1 c / A$$
$$A \to aAb / \varepsilon$$
$$S_2 \to aS_2 / B$$
$$B \to bBc / \varepsilon$$

where S is the starting symbol of CFG of L.

The grammar is ambiguous since the string $a^n b^n c^n$ (n=m) has two distinct LMD. or RMD. one starting with $S \Rightarrow S_1$, and another with $S \Rightarrow S_2$. it does of course not follows that L is inherently ambiguous as there might exist some other nonambiguous grammar for it. But in some way $L_1$ and $L_2$ have some conflicting requirement.

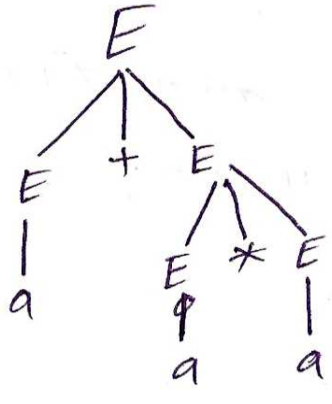
## Removing Ambiguity from grammars:

if a context free grammar is ambiguous, it is often possible and usually desirable to find an equivalent unambiguous CFG. Although some CFL are inherently ambiguous language in the sense that they can not be produced except by Ambiguous grammars. Ambiguity is usually the property of the grammar rather than the language.

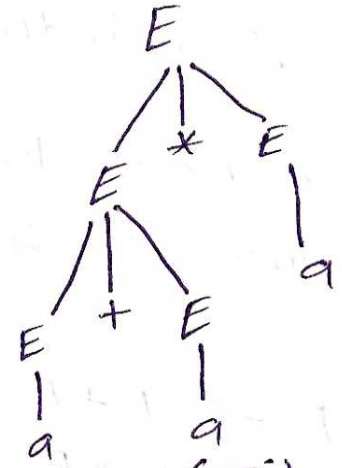
Let a grammar

$$E \rightarrow E+E \,/\, E*E \,/\, a.$$

then there are two parse tree ∴ possible for a algebric expression $a+a*a$



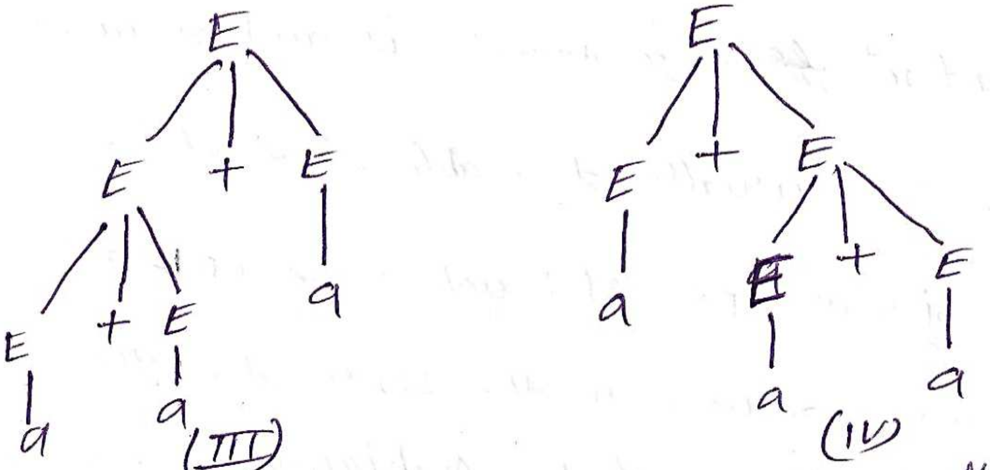
(I)



(II)

To remove ambiguity from grammar we need to force only (II) Parse tree to be legal in which the $*$ operator has higher precedence then $+$ because $a+a*a$ is equivalent to $a+(a*a)$

Again the Parse tree for a+a+a are




in this case we assume the + operator is left assosiative so the string a+a+a is equivalent to (a+a)+a. Thus, we need to force only III Parse tree to be legal.

Note:

for removing the ambiguity we will apply two rules

(i) if the grammar has left assosiative operator (+, -, *) then induce the left recursion.

(ii) if the grammar has right assosiative operator then induce the right recursion.

ex : $E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow a$

this grammar is ambigus for string a+a*a

Let +, * is left assosiative

then add a Production $E \to T$ and change $E \to E+E$ by $E \to E+T$ and replace all other E by T.

Then Productions are

$$E \to E+T$$
$$E \to T$$
$$T \to T*T$$
$$T \to a$$

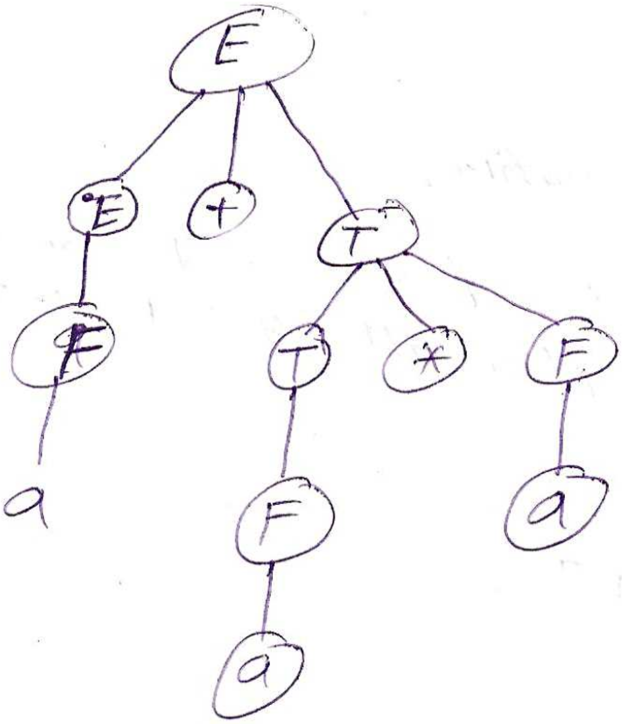
now in $T \to T*T$, right sided T in RHS Replace by a new variable F and introduce a new Production $T \to F$, replace all other T by F.

Then Productions are

$$E \to E+T$$
$$E \to T$$
$$T \to T*F$$
$$T \to F$$
$$F \to a$$

Now the grammar is unambiguous

Now Parse tree for string a+a*a is unique.


E
E
+
T
F
T
*
F
a
F
a
a

## Properties of CFL:

(I) CFL are closed under union or union of two CFL is always CFL

Prove: given $L_1, L_2$ is CFL

$L = L_1 \cup L_2$

CFG for $L_1$ $G(V_1, T_1, P_1, S_1)$
CFG for $L_2$ $G(V_2, T_2, P_2, S_2)$ $\Big\}$ $V_1 \cap V_2 = \phi$

then CFG for $L$ $G(V, T, P, S)$

where $V = V_1 \cup V_2 \cup S$
$P = P_1 \cup P_2 \cup \{S \to S_1 / S_2\}$
$T = T_1 \cup T_2$
$S = \{S\}$

(II) CFL are closed under concatenation or concatenation of two CFG is also CFG

given $L_1, L_2$ is CFL

then $L = L_1 . L_2$

CFG for $L \Rightarrow G = (V, T, P, S)$

where $V = V_1 \cup V_2 \cup S$
$T = T_1 \cup T_2$
$P = P_1 \cup P_2 \cup \{S \to S_1 . S_2\}$
$S = \{S\}$

(3) CFL are closed under closer operation
or closer of any CFL also CFL

Prove

L is CFG then CFG for L is

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

then $L^*$ is also CFL & CFG for this is

$G' = \{V', T', P', S'\}$

where

$V' = V \cup S'$

$T' = T$

$P' = P \cup \{S' \rightarrow SS' / \varepsilon\}$

$S' = S'$

and $L^+$ is also CFL & CFG for this is

$G'' = (V'', T'', P'', S'')$

where $V' = V \cup S''$

$T'' = T$

$P'' = P \cup \{S'' \rightarrow SS' / S\}$

$S'' = S''$

(4) CFL are not closed under intersection. {it is not Guarantee that intersection of two CFL is CFL}

~~Let~~ $L_1 = a^n b^n c^m : n, m \geq 1$ is CFL

$L_2 = a^n b^m c^m : n, m \geq 1$ is CFL

$L_1 \cap L_2 = a^n b^n c^n : n \geq 1$ it is not CFL.

(V) CFL are not closed under complementation (may or may not be)

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$