Ambiguity

## ⊛ Ambiguous Grammar:-

↳ An Ambiguous grammar is a grammar with either more than one leftmost derivation tree or more than one right-most derivation trees.

## ⊛ Eliminating Ambiguity of a Context-Free Grammar:-

Let us take an example:-

Grammar:

$E \to E+E$

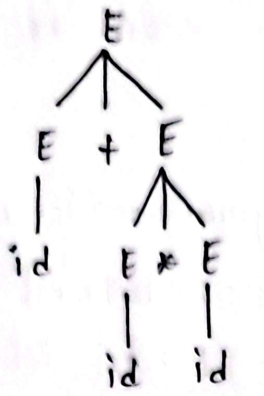
$E \to E*E$

$E \to id$.

Input string: $id+id*id$

The leftmost derivation can be done in two ways:

$$
\begin{aligned}
E &\to E+E \\
&\to id+E \\
&\to id+E*E \\
&\to id+id*id
\end{aligned}
$$

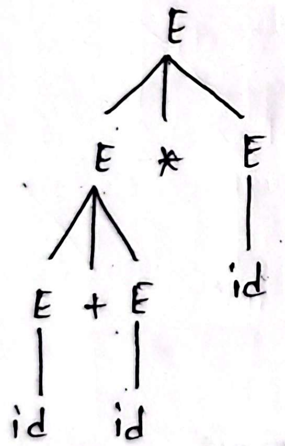


Also, $E \to E*E$

$\to E+E*E$

$\to id+E*E$

$\to id+id*id$



For the given input string, we get two leftmost derivation trees.
We need to eliminate the ambiguity in the grammar.

↳ Ambiguity from all grammar cannot be eliminated.

↳ We can use Precedence and Associativity to remove the ambiguity from some grammar.

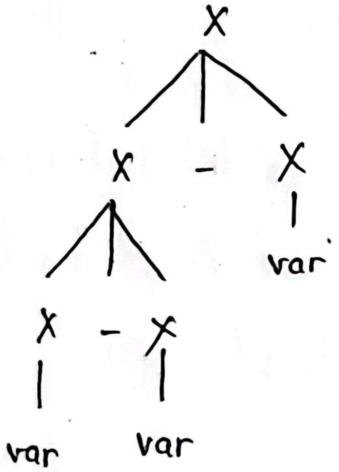
Let us take an example:-

Grammar:

$X \rightarrow X - X$

$X \rightarrow var \mid const$

A string a-b-c has two leftmost derivations:



$X \rightarrow X - X$

$\rightarrow var - X$

$\rightarrow var - X - X$

$\rightarrow var - var - X$

$\rightarrow var - var - var$

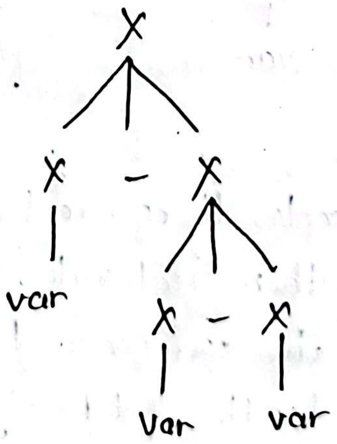
$\rightarrow a - b - c$

$X \rightarrow X - X$

$\rightarrow X - X - X$

$\rightarrow var - var - var$

$\rightarrow a - b - c$



For example, if we take the values $a=2$, $b=3$ and $c=4$

$a-b-c = 2-3-4 = -5$

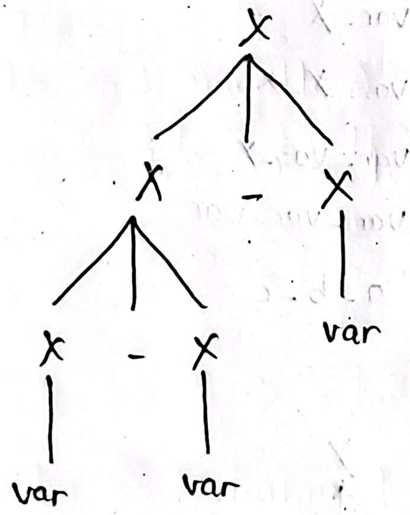
In the first derivation tree, expression will be evaluated as

$(a-b)-c = (2-3)-4 = -5$

In the second derivation tree, expression will be evaluated as

$a-(b-c) = 2-(3-4) = 3$:

Observe that both parse trees aren't giving the same value. They have differnt meaning. Therefore, the first derivation tree is the correct parse tree for grammar.



In the above example, there are two same operators in the expression. According, to the mathomatical rules, the expression must be evaluated based on the associativity of the operator used. The operator (-) which gives left to right associativity. So, for the left to right associative operators, the parse tree has to be left associative.

Let us convert the grammar into unambiguous grammar:

X → X-X

X → var|const

we need to make the grammar left-recursive we need to place a random non-terminal in place of the right non-terminal.

X → X-P|P

P → var|const

Now for the string. a-b-c

X → X-P

→ X-X-P

→ P-P-var

→ var-var-var

<u>For the Grammar</u>:

E → E+E

E → E*E

E → id

↳ This grammar will give two left derivation trees for the string. id+id*id

↳ we can't use associativity here as there are two different operators + and *. Hence we need to use "Precedence".

↳ In the string id+id*id, the order of evaluation must be id+(id*id) as '*' has more precedance than '+'. The operator at with the highest priority must be evaluated first. Hence, the operators with high priority are to be arranged in the lower levels of the parse tree.

$E \rightarrow E+P \mid P$

$P \rightarrow P*Q \mid Q$

$Q \rightarrow id$