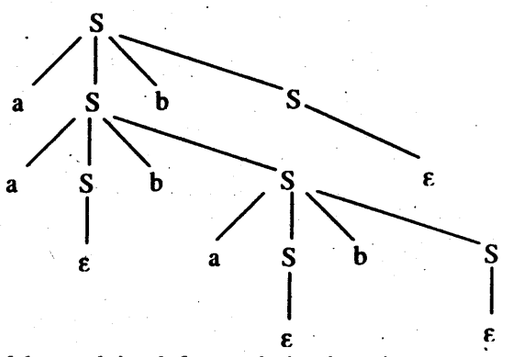
Consider the left most derivation again for the string *aababb* but using different set of productions

$S \Rightarrow aSbS$ by using $S \rightarrow aSbS$
$\Rightarrow aaSbSbS$ by using $S \rightarrow aSbS$
$\Rightarrow aabSbS$ by using $S \rightarrow \varepsilon$
$\Rightarrow AabaSbSbS$ by using $S \rightarrow aSbS$
$\Rightarrow aababSbS$ by using $S \rightarrow \varepsilon$
$\Rightarrow aababbS$ by using $S \rightarrow \varepsilon$
$\Rightarrow aababb$ by using $S \rightarrow \varepsilon$



Since there are two parse trees for the string *aababb* by applying leftmost derivation, the grammar is ambiguous.

Note: Instead of deriving the string **aababb** in the above example, we can derive the string "**abab**" so that two different parse trees are obtained and hence we can show that the grammar is ambiguous.

Example 5.22: Obtain the unambiguous grammar for the grammar shown

$$E \rightarrow E+E \mid E-E$$
$$E \rightarrow E*E \mid E/E$$
$$E \rightarrow (E) \mid I$$
$$I \rightarrow a \mid b \mid c$$

and obtain the derivation for the expression (a+b) * (a-b)

In example 5.14, it has been proved that the grammar is ambiguous. This grammar can be converted into unambiguous grammar based on the precedence. We know that identifiers such as *a*, *b* and *c* have the highest precedence, then the expression within '(' and ')' and then * or / whichever occurs first from left to right and finally + or – whichever occurs first from left to right. So, the final grammar which is unambiguous is shown below by assuming * and / have the highest priority compared to that of + and – and also by assuming all these operators are left associative.

$$I \rightarrow a \mid b \mid c$$
$$F \rightarrow (E) \mid I$$
$$T \rightarrow T*F \mid T/F \mid F$$
$$E \rightarrow E+T \mid E-T \mid T$$

So, the final grammar which is unambiguous is shown below:

$$E \rightarrow E+T \mid E-T \mid T$$
$$T \rightarrow T*F \mid T/F \mid F$$
$$F \rightarrow (E) \mid I$$
$$I \rightarrow a \mid b \mid c$$

The derivation for the string (a+b) * (a-b) is shown below:

$$
\begin{aligned}
E &\Rightarrow T \\
&\Rightarrow T * F \\
&\Rightarrow F * F \\
&\Rightarrow (E) * F \\
&\Rightarrow (E+T) * F \\
&\Rightarrow (T+T) * F \\
&\Rightarrow (F+T) * F \\
&\Rightarrow (I+T) * F \\
&\Rightarrow (a+T) * F \\
&\Rightarrow (a+F) * F \\
&\Rightarrow (a+I) * F \\
&\Rightarrow (a+b) * F \\
&\Rightarrow (a+b) * (E) \\
&\Rightarrow (a+b) * (E-T) \\
&\Rightarrow (a+b) * (T-T) \\
&\Rightarrow (a+b) * (F-T) \\
&\Rightarrow (a+b) * (I-T) \\
&\Rightarrow (a+b) * (a-T) \\
&\Rightarrow (a+b) * (a-F) \\
&\Rightarrow (a+b) * (a-I) \\
&\Rightarrow (a+b) * (a-b)
\end{aligned}
$$

So, the string (a+b) * (a-b) is derived from the given grammar.

Note: If we assume + and – are having highest priority compared to that of * and / and all the operators left associate the grammar will be of the form

$$
\begin{aligned}
E &\rightarrow E*T \mid E/T \mid T \\
T &\rightarrow T+F \mid T-F \mid F \\
F &\rightarrow (E) \mid I \\
I &\rightarrow a \mid b \mid c
\end{aligned}
$$

Note: If we assume + and – are having highest priority compared to that of * and / and all the operators right associate the grammar will be of the form

$$
\begin{aligned}
E &\rightarrow T*E \mid T/E \mid T \\
T &\rightarrow F+T \mid F-T \mid F \\
F &\rightarrow (E) \mid I \\
I &\rightarrow a \mid b \mid c
\end{aligned}
$$

Note: If we assume + and – are having highest priority and left associate when compared to that of * and / and if * and / operators are right associate the grammar will be of the form

$$
\begin{aligned}
E &\rightarrow T*E \mid T/E \mid T \\
T &\rightarrow T+F \mid T-F \mid F \\
F &\rightarrow (E) \mid I \\
I &\rightarrow a \mid b \mid c
\end{aligned}
$$

Note: Whenever a left associative operator is involved use left recursive production and if the operator involved is right-associative, use right recursive production.

Note: A grammar G is *ambiguous* if there exists some string $w \in L(G)$ for which there are two or more distinct derivation trees. If there exists a language L for which there is no unambiguous grammar, then the language is called *inherently ambiguous language* and the grammar from which the language is derived is called ***inherently ambiguous grammar***.

For example, the grammar shown in example 5.15 is ambiguous. But, we have an equivalent unambiguous for the grammar (see example 5.22). So, the language generated from the grammar shown in example 5.15 is unambiguous.

There are some inherently ambiguous grammars (for these grammars unambiguous grammars do not exist). For example, consider the language

$$L = \{a^n b^n c^m d^m \mid m \geq 1, n \geq 1\} \cup \{a^n b^m c^m d^n \mid m \geq 1, n \geq 1\}$$

The grammar to generate the language is shown below:

$$\begin{aligned} S &\rightarrow AB \mid C \\ A &\rightarrow aAb \mid ab \\ B &\rightarrow cBd \mid cd \\ C &\rightarrow aCd \mid aDd \\ D &\rightarrow bDc \mid bc \end{aligned}$$

The string *abcd* can be derived from the grammar as shown below:

$$\begin{aligned} S &\Rightarrow AB && \text{by using } S \rightarrow AB \\ &\Rightarrow abB && \text{by using } A \rightarrow ab \\ &\Rightarrow abcd && \text{by using } B \rightarrow cd \end{aligned}$$

The same string *abcd* can be derived from the by applying different set of productions as shown below:

$$\begin{aligned} S &\Rightarrow C && \text{by using } S \rightarrow C \\ &\Rightarrow aDd && \text{by using } C \rightarrow aDd \\ &\Rightarrow abcd && \text{by using } D \rightarrow bc \end{aligned}$$

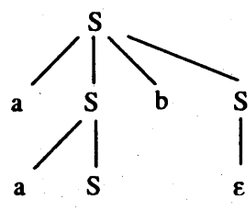
So, if we write the parse trees for both the derivations, the parse trees are different and naturally the grammar is ambiguous. It is not possible to obtain the unambiguous grammar for this and so it is inherently ambiguous grammar.

Example 5.23: Show that the following grammar is ambiguous by taking the string *aab* and also obtain the equivalent unambiguous grammar

$$S \rightarrow aS \mid aSbS \mid \varepsilon$$

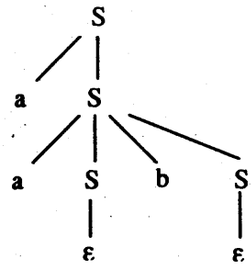
Solution: Consider the leftmost derivation for the string *aab* and the corresponding parse tree

$S \Rightarrow aSbS$ by using $S \rightarrow aSbS$
$\Rightarrow aaSbS$ by using $S \rightarrow aS$
$\Rightarrow aabS$ by using $S \rightarrow \varepsilon$
$\Rightarrow aab$ by using $S \rightarrow \varepsilon$



The string *aab* can also be obtained using the following leftmost derivation (shown along with parse tree)

$S \Rightarrow aS$ by using $S \rightarrow aS$
$\Rightarrow aaSbS$ by using $S \rightarrow aSbS$
$\Rightarrow aabS$ by using $S \rightarrow \varepsilon$
$\Rightarrow aab$ by using $S \rightarrow \varepsilon$



Similarly we can obtain two right most derivations and show that the grammar is ambiguous. The unambiguous grammar can be obtained by introducing a non-terminal A as shown below:

$S \rightarrow aS \mid aAbS \mid \varepsilon$
$A \rightarrow aAbA \mid \varepsilon$

5.7 Parsing

By parsing, we can check whether a string *w* can be derived from the grammar G. If *w* can be derived by applying leftmost derivation from the start symbol, we say that parsing is successful otherwise, parsing is not successful. Parsing is nothing but finding a sequence of productions by which *w* in L(G) is derived. Parsing can be easily done by a special type of context free grammar called *Simple Grammar* which is also called *S-Grammar*. The formal definition of S-Grammar is shown below:

Definition: The Simple Grammar or S-Grammar is a special type of grammar where all productions are of the form

$$A \rightarrow a\alpha$$

where $A \in V$, $a \in T$ and $\alpha \in V^*$ where a pair (A, a) can occur in at most once in P i.e., if A is there on the left hand side of the production, then there can be maximum of one production where *a* is the first symbol on the right hand side of the production. For example, consider the grammar shown below which is an S-Grammar

$S \rightarrow aABB$
$A \rightarrow aA \mid b$
$B \rightarrow bB \mid a$

An S-Grammar

where as the following grammars are not S-Grammars.

$$S \rightarrow aABb$$
$$A \rightarrow aA \mid b$$
$$B \rightarrow bB \mid b$$

Not an S-Grammar because of terminal b in S-production

$$S \rightarrow aABB$$
$$A \rightarrow aA \mid a$$
$$B \rightarrow bB \mid b$$

Not an S-Grammar because of first symbol in both the A-productions is a

Example 5.24: Find a Simple Grammar (S-Grammar) for the regular expression aaa*b + b

For a grammar to be simple grammar, no two production should have the same variable A on the left and same terminal *a* as the first symbol on the right hand side of the production and this terminal should be followed by zero or more variables. So, the resulting grammar can take the form

$$S \rightarrow aA \mid b$$
$$A \rightarrow aB$$
$$B \rightarrow aB \mid b$$

If we apply the production

$$S \rightarrow b$$

then from S we get the string b and the derivation for this is

$$S \Rightarrow b$$

So, the string *b* can be obtained successfully from S. The first part of the regular expression aaa*b can be derived using the productions

$$S \rightarrow aA$$
$$A \rightarrow aB$$
$$B \rightarrow aB \mid b$$

Note that by applying the first two productions, we get the partial derivation

$$S \Rightarrow aA \Rightarrow aaB$$

using which we have obtained two *a*'s followed by a variable B. It is clear from the B-production that B can generate one or more a's or a *b*. Thus, the required language is generated by the grammar.

Note: The same regular expression can be represented using the following grammar also.