UNIT-II

GRAMMAR INTRODUCTION

Context-free Grammar: (CFG)

Def: A CFG is a way of describing languages by recursive rules (or) substitution rules called productions

A CFG is denoted $G = (V, T, P, S)$ where V and T are finite set of variables and terminals respectively. We assume that V and T are disjoint. P is a finite productions, each production is of the form $A \rightarrow \alpha$ where A is a variable and α is a string of symbols from $(VUT)^*$. Finally S is a special variable called the start state.

Eg: $CFG = (\{E\}, \{+, *, (,), id\}, P, E)$

P can be defined as

$E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow (E)$
$E \rightarrow id$

(or)

$E \rightarrow E+E \mid E*E \mid (E) \mid id$

Note:-

i) Capital letters — Variables / Non terminals

ii) Small letters / { } — Terminals.

Derivations and languages:

A grammar is used to describe a language by generating each string of that language in the following manner:

i) Write down the start variable
It is the variable on the left hand side of the top rule, unless specified otherwise.

ii) find a variable that is written down on right hand side, that starts with that variable. Replace the written down variable with the right hand side of that rule.

iii) Replace step (ii) until no variables remain.

The sequence of substitutions to obtain a string is called a derivation.

Formal definition of derivation:

If $A \to \beta$ is a production in a grammar G and α, γ are any two strings on $(V \cup \Sigma)$, then we say $\alpha A \gamma$ directly derives $\alpha \beta \gamma$ in G.

i)

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma .$$

Formal definition of language:

If $G(V, T, P, S)$ is a CFG, the language of G, denoted $L(G)$ is the set of terminal strings that have derivations from the start symbol. i.e.

$$L(G) = \{ w \text{ in } T^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

If the language L is the language of some context-free grammar, then L is said to be a context-free language or CFL.

$L(G) = \{ w \mid w \text{ is in } T^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w \}$

What does it mean?

w is a string such that

(i) string consists solely of terminals
(ii) string can be derived from S.

Construct the language $L(G)$ for given CFG.

$G = (V, T, P, S)$ where $V = \{S\}$

$T = \{a, b\}$

$P: \{ S \to aSb, \; S \to ab \}$

Soln:

$S \to aSb$ —①

$S \Rightarrow a\underline{S}b$

$\Rightarrow aa\underline{S}bb \quad (\because ①)$

$\Rightarrow aaa\underline{S}bbb \quad (\because ②)$

$\cdots$

$\Rightarrow a^{n-1}\underline{S}\,b^{n-1}$

$\Rightarrow a^{n-1}ab\,b^{n-1}$

$\Rightarrow a^n b^n$

$\therefore L(G) = \{ a^n b^n \mid n \geq 1 \}$.

3. Let $G = (\{S, C\}, \{a, b\}, P, S)$ where P consists of $S \to aCa$, $C \to aCa \mid b$. Find $L(G)$.

Ans: From this grammar we get the strings such as,

$\{ aba, aabaa, aaabaaa \ldots \}$.

$\therefore L(G) = \{ a^n b a^n \mid n \geq 1 \}$.

4. Find $L(G)$ for G: $S \to aSbbS$
$S \to abb \mid \varepsilon$

G generates the strings like $\{\varepsilon, abb, aabbbb, aaabbbbbb \ldots\}$. So the language,

4. Construct the CFG for the regular expression $(0+1)^*$

P: $S \to 0S / 1S$
$S \to \varepsilon$

$V = \{S\}$
$T = \{0, 1\}$

5. Construct the CFG for the language having any no. of a's over the set $\Sigma = \{a\}$

i) a^*

P: $S \to Sa / \varepsilon$

6. Construct a grammar for the language containing two a's over the set $\Sigma = \{a, b\}$

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

$V = \{S, A\}$

$T = \{a, b\}$

P: $\{ S \to AaAaA$
$A \to aA / bA$
$A \to \varepsilon \}$

7. Construct CFG for the language L which has all the strings which are all palindrome over the set $\Sigma = \{a, b\}$ i) ww^R.

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

where $V = \{S\}$

P: $\{ S \to aSa$
$S \to bSb$
$S \to a$

9. Construct a grammar generating $L = wcw^T$ where $w \in \{a,b\}^*$.

ii) $L = \{c, aca, bcb, abcba, bacab, \ldots\}$

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

where $V = \{S\}$

$T = \{a, b, c\}$

$P: \{ S \to aSa$
$S \to bSb$
$S \to c \}$

10. Construct CFG which consist of all the strings having at least one occurrence of 000.

$P: \{ S \to SoooS$
$S \to 0/1/\varepsilon \}$

11. Construct CFG for the language containing at least one occurance of double 'a' over the set $\Sigma = \{a,b\}$.

$P: \{ S \to AaaA$
$A \to aA / bA / \varepsilon \}$

Derivation Trees:

It is a graphical representation for the derivation of the given production rules for a grammar CFG. The derivation tree is also called as parse tree. It is useful to display derivations as trees.

Properties of any derivation tree:

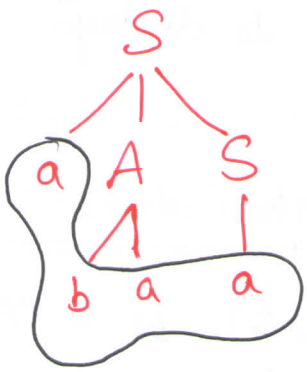
Let $G = (V, T, P, S)$ be a CFG. A tree is a derivation tree for G, if

1) Every node (vertex) has a label, which is a symbol of $V \cup T \cup \{\varepsilon\}$
2) The label of the root is S.
3) If a vertex is interior and has label A then A must be in V.
4) If n has label A and vertices $n_1, \ldots n_k$ are the sons of vertex n, in order from the left with labels $X_1, X_2, \ldots X_k$ respectively. Then $A \rightarrow X_1 X_2 \ldots X_k$ must be a production.
5) If vertex n has label ε, then n is a leaf and is the only one son of its father.

Note :- Root node - Starting Non terminal
Interior nodes - Non terminals
leaf nodes - terminals
Should read from left to right.

1. Consider G: $S \to aAS \mid a$
$A \to ba$

Derivation tree for the string "abaa"



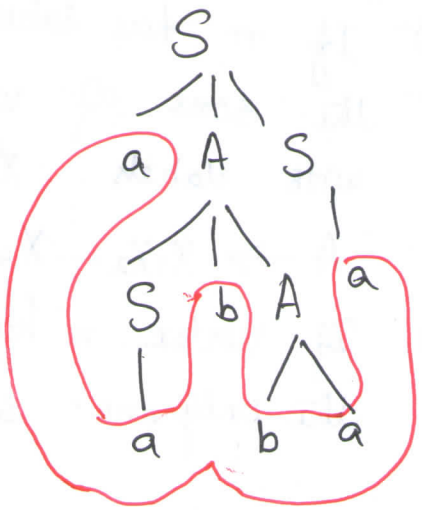
— Root

— Interior nodes

— leaves

2. Consider G whose productions are
$S \to aAS \mid a$, $A \to SbA \mid SS \mid ba$.
Show that $S \overset{*}{\Rightarrow} aabbaa$ and construct a derivation tree whose yeild is "aabbaa"

$S \Rightarrow aAS$
$\Rightarrow aSbAS$
$\Rightarrow aabAS$
$\Rightarrow aabbaS$
$\Rightarrow aabbaa$



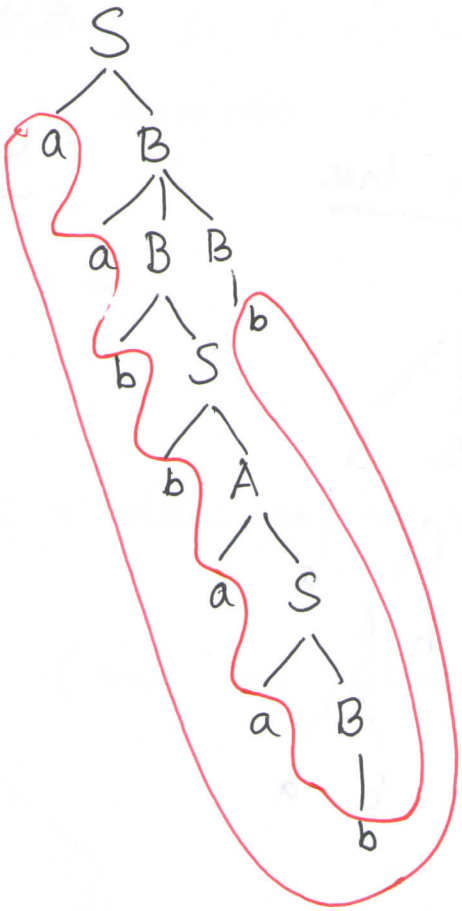
3. Construct the derivation tree for the string aabbaabb for the given grammar

$S \to aB \mid bA$
$A \to a \mid aS \mid bAA$

Derivation for the string "aa bbaabb":

$$
\begin{aligned}
S &\Rightarrow a\underline{B} \\
&\Rightarrow aa\underline{B}B \\
&\Rightarrow aab\underline{S}B \\
&\Rightarrow aabb\underline{A}B \\
&\Rightarrow aabba\underline{S}B \\
&\Rightarrow aabbaa\underline{B}B \\
&\Rightarrow aabbaab\underline{B} \\
&\Rightarrow aabbaabb
\end{aligned}
$$

Derivation tree:



Sub tree:

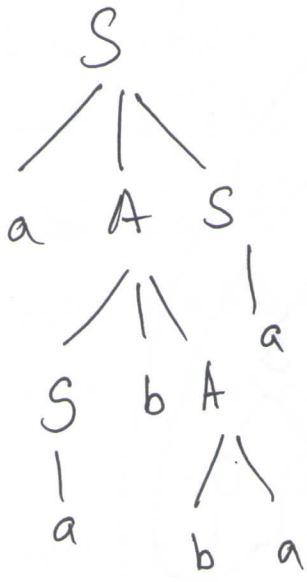
A Sub tree of a derivation tree T is tree satisfying the following constraints:

a) whose root is some vertex V of T
b) whose vertices are the descendants of V to gether with their labels.
c) whose edges are those connecting the descendants of V.

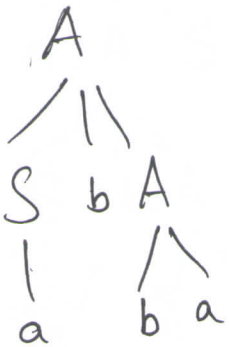
Note:- Sub tree looks like a derivation tree except that label of the root may not be the start symbol of the grammar.

Consider the geammar G: $S \to aAS \mid a$
$A \to SbA \mid SS \mid ba$

Derivation tree



Sub tree



Theorem:

The Relationship between derivation and derivation Tree

Let $G = (V, T, P, S)$ be a context free grammar. Then prove that $S \overset{*}{\Rightarrow} \alpha$ if and only if there is a derivation tree in grammar G with yield α.

Proof:

Here what we have to prove is that for any A in V, $A \overset{*}{\Rightarrow} \alpha$ if and only if there is an A-tree with α as the yield.

Basis:

Assume that there is only one interior node A. The derivation tree yielding $X_1, X_2 \cdots X_n$. In that case $X_1 X_2 \cdots X_n$ must be terminals. i.e., $\alpha_1 \alpha_2 \cdots \alpha_n$.

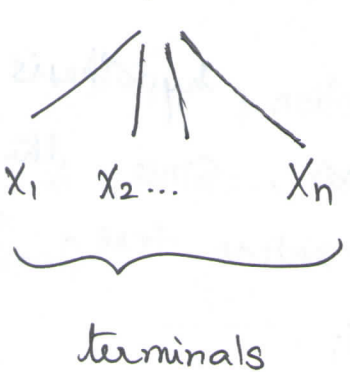
$A \Rightarrow X_1 X_2 \cdots X_n$

Replace X_i by α_i

$A \Rightarrow \alpha_1 \alpha_2 \cdots \alpha_n$

$A \overset{*}{\Rightarrow} \alpha$

Then there exists a derivation tree which derives the string α.



Induction Step:

We assume that the derivation will enable us to draw the tree with $k-1$ interior nodes. Here we have to prove that it is possible to draw the derivation tree for k interior nodes which gives the string α.

Consider the sons of the root A be $X_1, X_2 \ldots X_n$. Then surely $A \rightarrow X_1 X_2 \ldots X_n$ is a production in P where $n \geq 1$.

Case (i)

Let us assume "Vertex i is a leaf".

If vertex i is a leaf, Let $X_i = \alpha_i$. It is easy to see that if $j < i$, vertex j and all its descendants are to the left of vertex i and all of its descendants. Thus $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$.

Case (ii)

Let us assume "vertex i is not a leaf".

If the i^{th} son (vertex) is not a leaf, it is the root of a subtree and X_i must be a non-terminal. The subtree must be an X_i-tree and has some yield α_i.

By the Induction hypothesis, for each vertex that is not a leaf $X_i \overset{*}{\Rightarrow} \alpha_i$, Since the subtree with root X_i is not the entire tree. If $X_i = \alpha_i$, then surely $X_i \overset{*}{\Rightarrow} \alpha_i$.

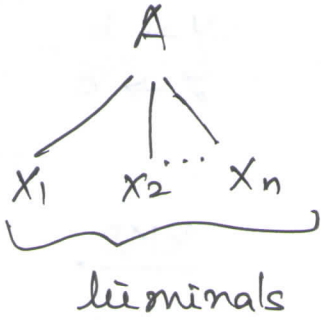
we can put all these partial derivations together

$$A \Rightarrow X_1 X_2 \cdots X_n$$
$$\overset{*}{\Rightarrow} \alpha_1 X_2 \cdots X_n$$
$$\overset{*}{\Rightarrow} \alpha_1 \alpha_2 \cdots X_n$$
$$\overset{*}{\Rightarrow} \alpha_1 \alpha_2 \cdots \alpha_n$$
$$\overset{*}{\Rightarrow} \alpha$$

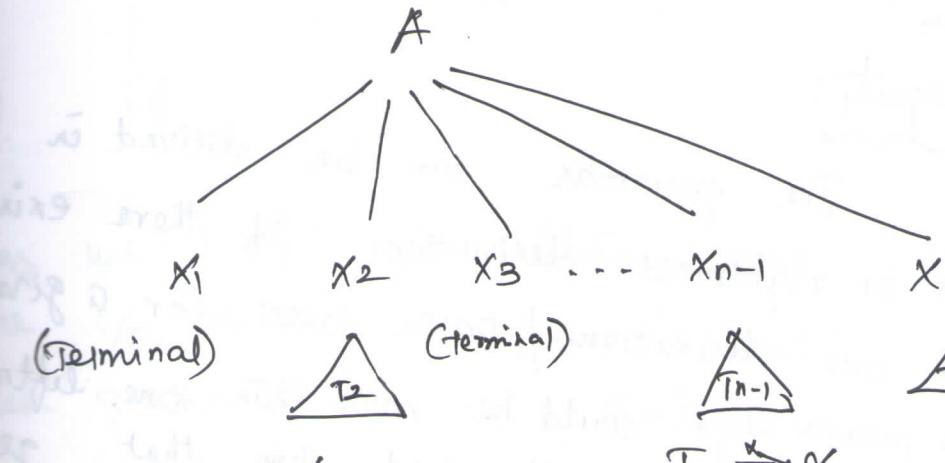
Thus $A \overset{*}{\Rightarrow} \alpha$.

we could produce as many as possible derivations from the given tree.

We begin by constructing an A-tree with n leaves labeled $X_1 X_2 \ldots X_n$ and no other vertices. This is shown in below



Each vertex with label X_i where X_i is not terminal, is replaced by the tree T_i which yields α_i. If X_i is a terminal, no replacement is made. An example appears in below figure. The yield of this is α.



AMBIGUITY

Leftmost derivation: (LMD)

The leftmost derivation is a derivation in which leftmost non-terminal is replaced first from the sentential form.

Rightmost Derivation (RMD)

The rightmost derivation is a derivation in which rightmost non-terminal is replaced first from the sentential form.

Consider the grammar G: $S \to XYZ$
$X \to a$
$Y \to b$
$Z \to c$

LMD

$S \underset{lm}{\Rightarrow} \underline{X}YZ$
$\underset{lm}{\Rightarrow} a\underline{Y}Z$
$\underset{lm}{\Rightarrow} ab\underline{Z}$
$\underset{lm}{\Rightarrow} abc$

RMD

$S \underset{rm}{\Rightarrow} XY\underline{Z}$
$\underset{rm}{\Rightarrow} X\underline{Y}c$
$\underset{rm}{\Rightarrow} \underline{X}bc$
$\underset{rm}{\Rightarrow} abc$

Ambiguity:

The grammar can be derived in either left most or rightmost derivation. If there exists more than one derivation / parse trees for a given grammar, that means there could be more than one leftmost or right

Check whether the given grammar is an ambiguous or not.

P: $E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow id$.

Left Most derivation

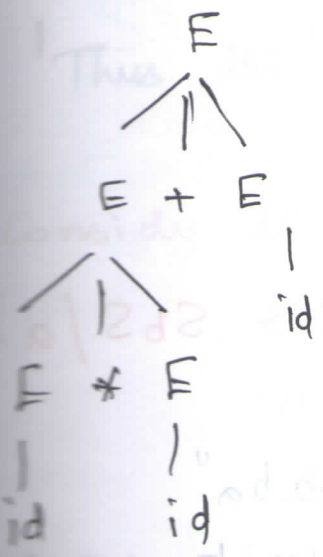
$E \Rightarrow \underline{E} + E$
$\Rightarrow \underline{E} * E + E$
$\Rightarrow id * \underline{E} + E$
$\Rightarrow id * id + \underline{E}$
$\Rightarrow id * id + id$

Rightmost derivation.

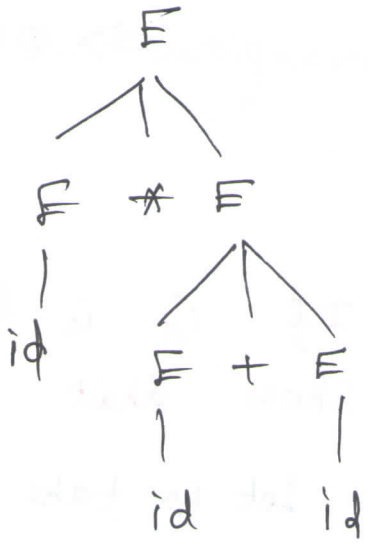
$E \Rightarrow E * \underline{E}$
$\Rightarrow E * E + \underline{E}$
$\Rightarrow E * \underline{E} + id$
$\Rightarrow E * id + id$
$\Rightarrow id * id + id$

Derivation Trees:

Using LMD



Using RMD

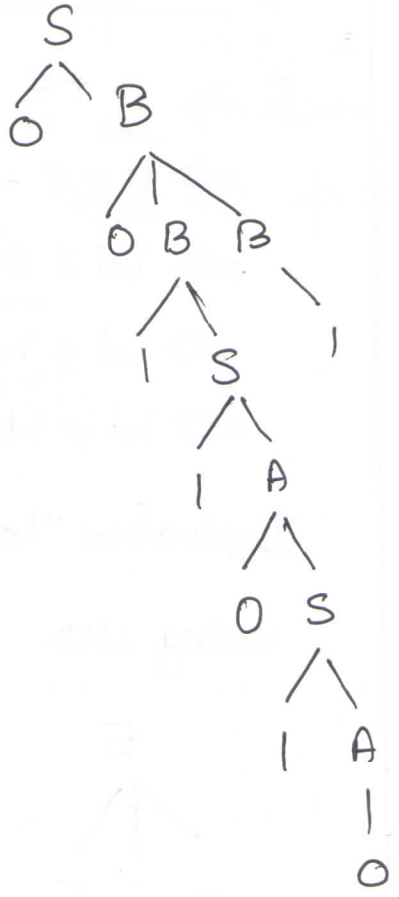


Thus we have more than one tree for same i/p string "id * id + id". Hence the given grammar is an ambiguous.

2. Let G be the grammar $S \to 0B / 1A$
$A \to 0 / 0S / 1AA$
$B \to 1 / 1S / 0BB$

for the string 00110101.
Find its leftmost derivation & derivation tree.

$S \Rightarrow 0\underline{B}$
$\Rightarrow 00\underline{BB}$
$\Rightarrow 001\underline{S}B$
$\Rightarrow 0011\underline{AB}$
$\Rightarrow 00110\underline{S}B$
$\Rightarrow 00110\underline{1AB}$
$\Rightarrow 001101\underline{0}B$
$\Rightarrow 00110101$



3. If G is the grammar $S \to SbS / a$.
Show that G is ambiguous

Let us take the string "ababa"
It can be derived by 2 different ways.

(i) Leftmost Derivation: $S \Rightarrow \underline{S}bS$
$\Rightarrow \underline{S}bSbS$
$\Rightarrow abSbS$

(ii) rightmost derivation: $S \Rightarrow SbS$

$\Rightarrow SbSbS$

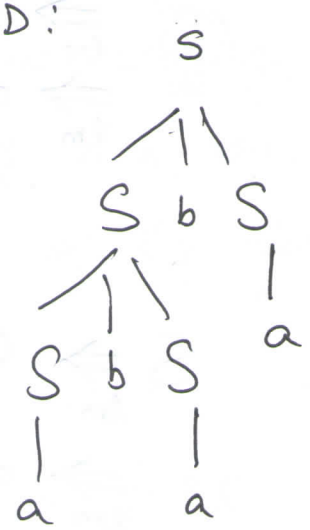
$\Rightarrow SbSba$

$\Rightarrow Sbaba$

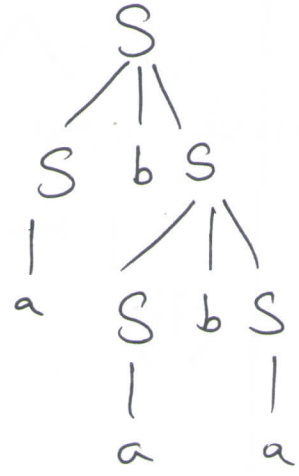
$\Rightarrow ababa.$

Derivation trees:

LMD:



RMD:



Thus the grammar is an ambiguous.

4. Consider the Grammar G:

$S \rightarrow AB \mid C$

$A \rightarrow aAb \mid ab$

$B \rightarrow cBd \mid cd$

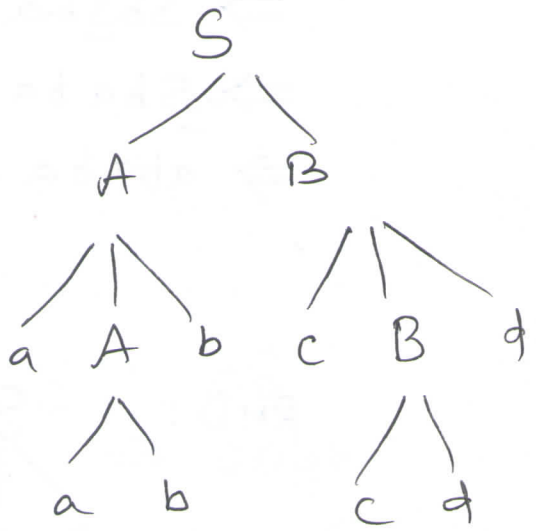
$C \rightarrow aCd \mid aDd$

$D \rightarrow bDc \mid bc$

Show that this grammar G is ambiguous for the string

LMD's :

(i)



$S \underset{lm}{\Rightarrow} \underline{A}B$

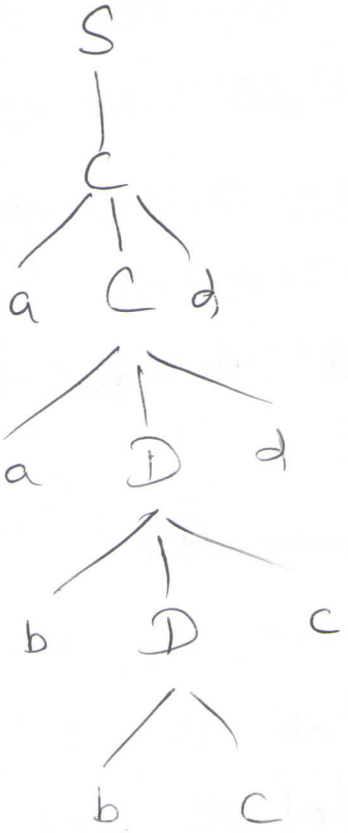
$\underset{lm}{\Rightarrow} aAbB$

$\underset{lm}{\Rightarrow} aabb\underline{B}$

$\underset{lm}{\Rightarrow} aabbc\underline{B}d$

$\underset{lm}{\Rightarrow} aabbccdd$

(ii)



$S \underset{lm}{\Rightarrow} \underline{C}$

$\underset{lm}{\Rightarrow} a\underline{C}d$

$\underset{lm}{\Rightarrow} aa\underline{D}dd$

$\underset{lm}{\Rightarrow} aabDcdd$

$\underset{lm}{\Rightarrow} aabbccdd$

Thus G is an ambiguous grammar.

SIMPLIFICATION OF CFG

Elimination of Useless Symbols:

The following approaches are used to identify the symbol has to be able to do to be useful.

1. We Say X is generating, if $X \overset{*}{\Rightarrow} w$ for some terminal string w.
2. We Say X is reachable, if there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta$ for some α and β.

Example:

$S \rightarrow aB \mid bX$
$A \rightarrow Bad \mid bSX \mid a$
$B \rightarrow aSB \mid bBX$
$X \rightarrow SBD \mid aBX \mid ad$

Solution:

Step 1. To find generating symbols.

1) Directly, A is generating since $A \rightarrow a$
2) Directly, X is generating since $X \rightarrow ad$
3) Since X is a useful symbol, S is generating because $S \rightarrow bX$
4) But B does not produce any string w

So, B is a non-generating symbol.
So eliminate B, we get

$S \rightarrow bX$
$A \rightarrow bSX$
$X \rightarrow ad$

Step 2. To find reachable Symbols.

Consider the derivation $S \Rightarrow bX$
$\Rightarrow bad$

∴ using X we can able to derive "bad" from S. So X is reachable symbol.

But A is a non-reachable symbol.
So we remove it and the final grammar after elimination of the useless symbols is

$S \rightarrow bX$
$X \rightarrow ad$

Elimination of ϵ - Productions:

(To find nullable Symbols.)

Example:
Consider the grammar $S \rightarrow ABC$
$A \rightarrow BC / a$
$B \rightarrow bAC / \epsilon$

1) first we identify the nullable Symbols.
   B, C are directly nullable.
2) Next, A is nullable because of the rule
   $A \rightarrow BC$.
3) finally, we find that S is nullable because of the rule $S \rightarrow ABC$.

After eliminating ε productions, G becomes

$S \rightarrow ABC \mid AB \mid BC \mid AC \mid A \mid B \mid C$
$A \rightarrow BC \mid B \mid C \mid a$
$B \rightarrow bAC \mid bA \mid bC \mid b$
$C \rightarrow cAB \mid cA \mid cB \mid c$

Eliminating Unit Productions:

A unit production is a production of the form $A \rightarrow B$, where both A and B are variables.

Example:-

$E \rightarrow T \mid E+T$
$T \rightarrow F \mid T*F$
$F \rightarrow I \mid (E)$
$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Unit productions are $E \rightarrow T$
$T \rightarrow F$

After eliminate the unit productions from the grammar, we get resulting grammar:

$E \rightarrow E+T \mid T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid Io \mid I1$
$T \rightarrow T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid Io \mid I1$
$F \rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid Io \mid I1$
$I \rightarrow a \mid b \mid Ia \mid Ib \mid Io \mid I1$

has no unit productions.

(X) Some care must be taken in the order of application of the constructions. A safe order is:

1. Eliminate ϵ-productions
2. Eliminate unit productions.
3. Eliminate useless symbols.

1. Begin with the grammar:

$S \rightarrow ABC \mid BaB$
$A \rightarrow aA \mid BaC \mid aaa$
$B \rightarrow bBb \mid a \mid D$
$C \rightarrow CA \mid AC$
$D \rightarrow \epsilon$

a) Eliminate ϵ-productions.
b) Eliminate any unit productions in the resulting grammar.

Solution:

(a) To Eliminate ε-productions:

- Directly D is nullable.
- B is also nullable since we have a rule B → D
- S is also nullable since we have a rule S → BaB

So we can eliminate nullables from the grammar. Then the resulting grammar is:

$$S \to ABC \mid BaB \mid AC \mid aB \mid Ba \mid a$$
$$A \to aA \mid BaC \mid aaa \mid aC$$
$$B \to bBb \mid a \mid D \mid bb$$
$$C \to CA \mid AC$$

(b) Eliminate unit productions:

from the resulting grammar, B → D is unit production.

$$S \to ABC \mid BaB \mid AC \mid aB \mid Ba \mid a$$
$$A \to aA \mid BaC \mid aaa \mid aC$$
$$B \to bBb \mid a \mid bb$$
$$C \to CA \mid AC$$

(c) Eliminate useless symbols:

step ① To find generating symbols.

S is generating symbol (∵ S → a)
B is generating symbol (∵ B → a)
A is generating " (∵ A → aaa)
C is non generating

Then after removal of C we get,

$$S \rightarrow BaB \mid aB \mid Ba \mid a$$
$$A \rightarrow aA \mid aaa$$
$$B \rightarrow bBb \mid a \mid bb$$

Step (ii) To find reachable symbol from the above grammar,

$$S \Rightarrow \underline{B}aB$$
$$\Rightarrow aa\underline{B}$$
$$\Rightarrow aaa$$
$$S \overset{*}{\Rightarrow} aaa$$

$\therefore$ B is reachable symbol.

A is non-reachable from S. So we can eliminate A from the grammar. Then

Ans:

$$S \rightarrow BaB \mid aB \mid Ba \mid a$$
$$B \rightarrow bBb \mid a \mid bb$$

2. Simplify the grammar:

$$S \rightarrow 0A0 \mid 1B1 \mid BB$$
$$A \rightarrow C$$
$$B \rightarrow S \mid A$$
$$C \rightarrow S \mid \epsilon$$

1. Elimination of ε- productions:

Directly C is nullable.

Indirectly A is nullable since $A \to C$
B is nullable since $B \to A$
S is also " since $S \to BB$

After eliminating nullable symbols, we get

$S \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11 \mid B$
$A \to C$
$B \to S \mid A$
$C \to S$

2. Elimination of unit productions: [$S \to B$, $A \to C$, $B \to S \mid A$, $C \to S$]

$S \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$
$A \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$
$B \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$
$C \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$

3. Elimination of useless symbols:

All are generating symbols. So we cannot eliminate.

But C is non-reachable from S. So we can eliminate C from resulting grammar.

Ans:

$S \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$
$A \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$
$B \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$

3. Simplify the grammar:

$S \rightarrow AAA \mid B$
$A \rightarrow aA \mid B$
$B \rightarrow \epsilon$

i. After elimination of ϵ-production:

$S \rightarrow AAA \mid B \mid AA \mid A$
$A \rightarrow aA \mid B \mid a$

2. After Elimination of unit production: [$S \rightarrow B$, $S \rightarrow A$, $A \rightarrow B$]

$S \rightarrow AAA \mid AA \mid aA \mid a$
$A \rightarrow aA \mid a$

3. There is no useless symbol.

Ans:

$S \rightarrow AAA \mid AA \mid aA \mid a$
$A \rightarrow aA \mid a$

4. Simplify the grammar:

$S \rightarrow aAa \mid bBb \mid \epsilon$
$A \rightarrow C \mid a$
$B \rightarrow C \mid b$
$C \rightarrow CDE \mid \epsilon$
$D \rightarrow A \mid B \mid ab$

1. Elimination of ε- production:

Directly S, C are nullables. [∵ S → ε, C → ε]

Indirectly A, B, D are nullable symbols.

Because, A → C
B → C
D → A / B

∴ After eliminating all the nullable symbols, we get

S → aAa / bBb / aa / bb
A → C / a
B → C / b
C → CDE / DE / CE / E
D → A / B / ab

2. Elimination of unit production:

A → C
B → C
C → E
D → A / B } are unit productions.

After eliminating those unit productions, we get

S → aAa / bBb / aa / bb
A → CDE / DE / CE / a
B → CDE / DE / CE / b
C → CDE / DE / CE
D → CDE / DE / CE / a / b / ab

3. Elimination of Useless Symbols:

Step 1: Generating Symbols are, S, A, B and D
C is non-generating symbol. It does not produce any terminals. So we can eliminate it.

After eliminating non-generating symbol from the resulting grammar,

$S \rightarrow aAa / bBb / aa / bb$
$A \rightarrow a$
$B \rightarrow b$
$D \rightarrow a / b / ab$

Step 2: D is not reachable from S to derive some strings. So we can eliminate D from the resulting grammar,

Ans:

$S \rightarrow aAa / bBb / aa / bb$
$A \rightarrow a$
$B \rightarrow b$