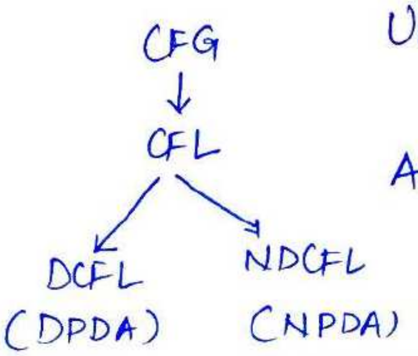
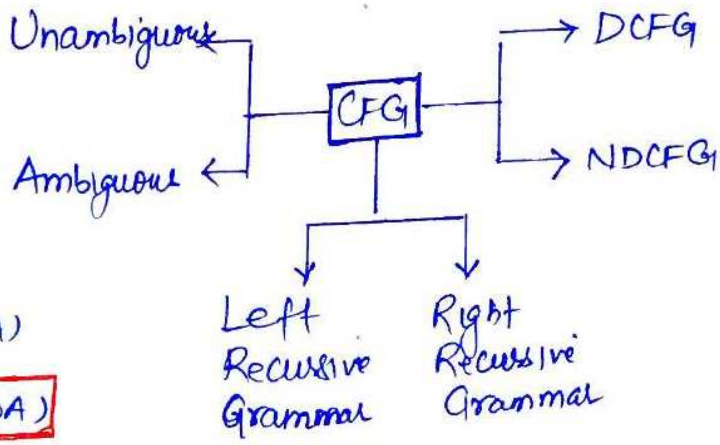
Context free Grammar (CFG):- A Grammar G is a CFG if every Production is of the form $\boxed{A \rightarrow \beta, \text{ where } A \in V, \beta \in (V \cup T)^*}$

Ex:- $S \rightarrow aSa \mid bSb \mid \varepsilon$





#Imp: $\boxed{P(DPDA) \subset P(NPDA)}$

Derivation:

- It is a Process of deriving a string
- A string $w \in L(G)$ iff there is atleast one derivation for it

Derivation Tree OR Parse Tree: Graphical represestation of derivation Process

- How to represent a derivation tree?
  - Root node is always start symbol
  - Internal nodes are always non terminals
  - Leaf node is always terminal node

NOTE:- Derivation is read from Left to Right

Ex:- $G: S \rightarrow aAB \mid a,\ A \rightarrow aBA \mid a,\ B \rightarrow b$ Construct Parse tree for String aabab

Sentencial form

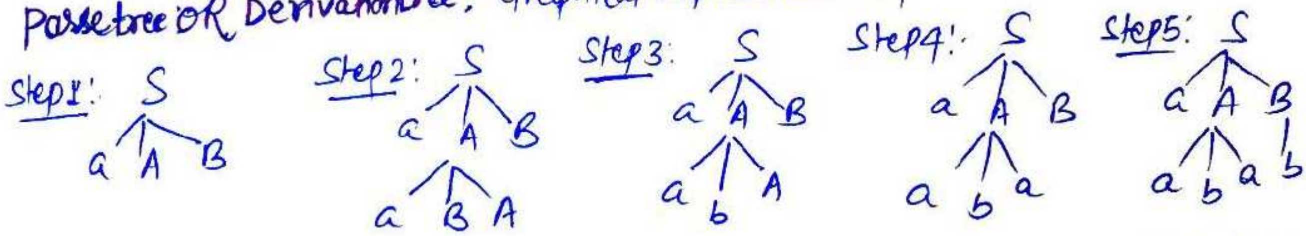
Sentence

Start Symbol

Derivation:- $S \Rightarrow aAB \Rightarrow aaBAB \Rightarrow aabAB \Rightarrow aabaB \Rightarrow aabab$

Derivation Process Sign

Parse tree OR Derivation tree: Graphical representation of derivation Process.



Parse tree. Ans

Types of derivation Process: two types

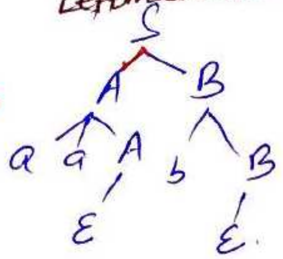
1- Leftmost derivation (LMD):- A derivation $S \overset{*}{\Rightarrow} \omega$ is called a Leftmost derivation if we apply a Production only to the Leftmost variable at every step.

Ex:- Consider a Grammar G_1 for the language $L = \{a^{2n} b^m,\ m, n \geq 0\}$

G_1: $S \to AB$, $A \to aaA \mid \varepsilon$, $B \to bB \mid \varepsilon$, find LMD for string $\omega = aab$

$S \Rightarrow AB$
$\Rightarrow aaAB$; $A \to aaA$
$\Rightarrow aaB$; $A \to \varepsilon$
$\Rightarrow aabB$; $B \to bB$
$\Rightarrow aab$; $B \to \varepsilon$

Leftmost derivation tree:

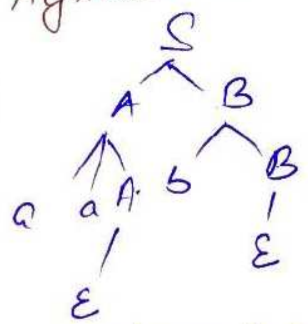


2- Right most derivation (RMD):- A derivation $S \overset{*}{\Rightarrow} \omega$ is a rightmost derivation if we apply a Production to the right most variable at every step

Ex:- RMD for string $\omega = aab$, consider above grammar.

$S \Rightarrow AB$
$\Rightarrow AbB$; $B \to bB$
$\Rightarrow Ab$; $B \to \varepsilon$
$\Rightarrow aaAb$; $A \to aaA$
$\Rightarrow aab$; $A \to \varepsilon$

Rightmost derivation tree:



Ex:- $G \{S \to 0B \mid 1A,\ A \to 0 \mid 0S \mid 1AA,\ B \to 1 \mid 1S \mid 0BB\}$ for the string 00110101 find LMD & RMD:

LMD:

$S \Rightarrow 0B$
$\Rightarrow 00BB$; $B \to 0BB$
$\Rightarrow 001B$; $B \to 1$
$\Rightarrow 0011S$; $B \to 1S$
$\Rightarrow 00110B$; $S \to 0B$
$\Rightarrow 001101S$; $B \to 1S$
$\Rightarrow 0011010B$; $S \to 0B$
$\Rightarrow 00110101$; $B \to 1$

RMD:

$S \Rightarrow 0B$
$\Rightarrow 00BB$; $B \to 0BB$
$\Rightarrow 00B1S$; $B \to 1S$
$\Rightarrow 00B10B$; $S \to 0B$
$\Rightarrow 00B101S$; $B \to 1S$
$\Rightarrow 00B1010B$; $S \to 0B$
$\Rightarrow 00B10101$; $B \to 1$
$\Rightarrow 00110101$; $B \to 1$

Ex:- $S \to aAS \mid aSS \mid \varepsilon$, $A \to SbA \mid ba$, find Leftmost and rightmost derivation tree for string $\omega = aabaa$

Sol^n:- Given String $\omega = aabaa$.

LMD:

$S \Rightarrow aSS$

$\Rightarrow aaASS$; $S \Rightarrow aAS$

$\Rightarrow aabaSS$; $A \to ba$

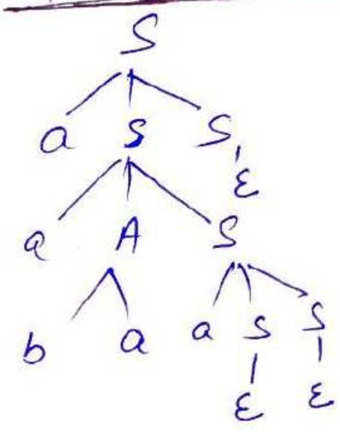
$\Rightarrow aabaaSSS$; $S \to aSS$

$\Rightarrow aabaaSS$; $S \to \varepsilon$

$\Rightarrow aabaaS$; $S \to \varepsilon$

$\Rightarrow aabaa$; $S \to \varepsilon$

Left most derivation tree:



RMD:

$S \Rightarrow aSS$

$\Rightarrow aSaAS$; $S \to aAS$

$\Rightarrow aSaAaSS$; $S \to aSS$

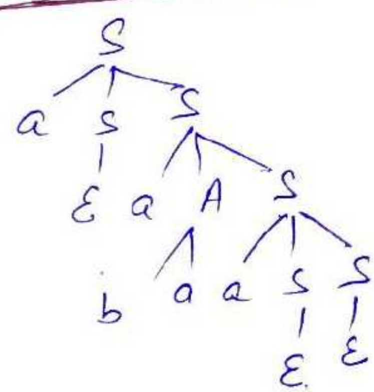
$\Rightarrow aSaAaS$; $S \to \varepsilon$

$\Rightarrow aSaAa$; $S \to \varepsilon$

$\Rightarrow aSabaa$; $A \to ba$

$\Rightarrow aabaa$; $S \to \varepsilon$

Rightmost derivation Tree:



Ex:- Consider the Grammar $S \to S+S \mid S*S \mid a \mid b$ find Left most and Rightmost derivations for string $\omega = a*a+b$

Sol^n: **LMD:**

$S \Rightarrow S*S$

$\Rightarrow a*S$

$\Rightarrow a*S+S$

$\Rightarrow a*a+S$

$\Rightarrow a*a+b$

RMD:

$S \Rightarrow S*S$

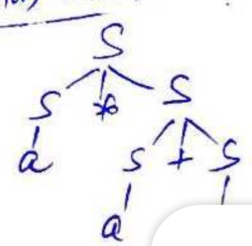
$\Rightarrow S*S+S$

$\Rightarrow S*S+b$

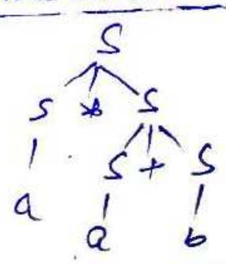
$\Rightarrow S*a+b$

$\Rightarrow a*a+b$

Leftmost derivation Tree:



Rightmost derivation Tree:



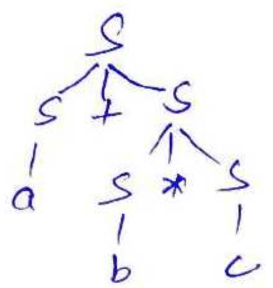
Ambiguous Grammar: A Grammar G is ambiguous grammar if

$\exists w \in L(G)$ Such that w has > 1 P.T, D.T, S.T, LMD, RMD } Either using 2LMD OR 2RMD

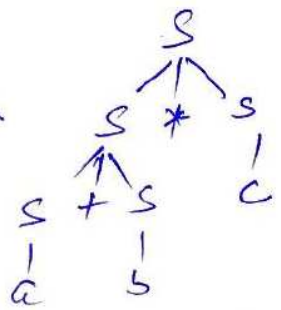
i.e a grammar G is ambiguous if there is more than one Parsetree or LMD/RMD for a string $w \in L(G)$

Ex:- G : $S \rightarrow S+S \mid S*S \mid a \mid b \mid c$, Grammar G is ambiguous because There exist two different LMD for a string $w = a+b*c$

LMD:



OR



NOTE:- A Grammar G is unambiguous if there exist exactly one Parsetree or LMD/RMD for all string $w \in L(G)$

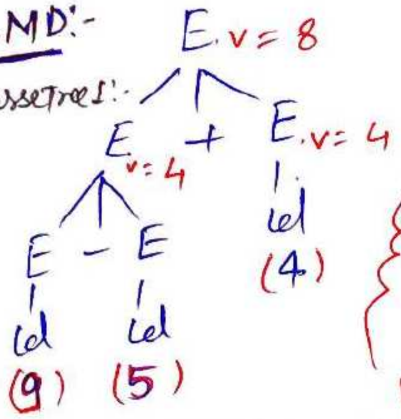
NOTE:- if a Grammar G is ambiguous, it doesn't mean language(L) is ambiguous

Ex:- State whether given grammar is ambiguous or not G: $E \rightarrow E+E \mid E-E \mid id$

Sol^n: We need take one string $w \in L(G)$ if there are more than one LMD/RMD if there are more than one Parsetree Then given grammar is ambiguous. Also if L(G) has more than one Parsetree Then given grammar is ambiguous for w (Let $w = 9-5+4$)

LMD:-

Parsetree 1:-



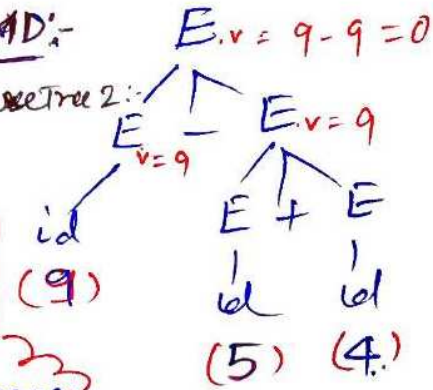
$w = 9-5+4 = 4$

OR

Two different Ans for 9-5+4 b/c given Grammar is Ambiguous Grammar

LMD:-

Parsetree 2:-

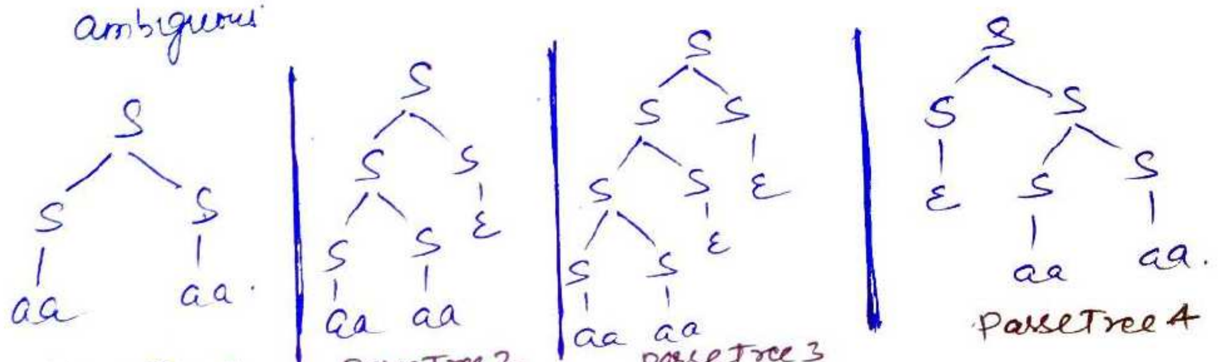


$w = 9-5+4 = 0$

Ex:- state whether given grammar is ambiguous or not

G: $S \to aa \mid bb \mid SS \mid \varepsilon$

Sol^n: We need take one string $w \in L(G)$ if there are more than one LMD/RMD then given grammar is ambiguous. Also if $L(G)$ has more than one Parse tree for W (Let $w = aaaa$) then given grammar is ambiguous.



ParseTree 1 | ParseTree 2 | parsetree 3 | ParseTree 4

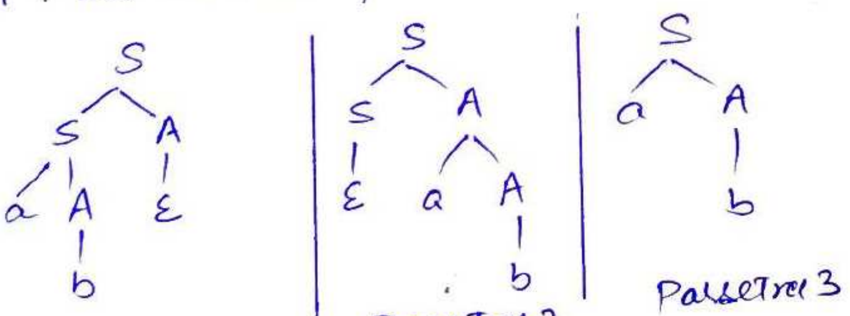
We see that there are more than one LMD or ParseTree for a String $w \in L(G)$

Therefore, given grammar is ambiguous.

Ex:- State whether given grammar is ambiguous or not

G: $S \to SA \mid aA \mid \varepsilon$; $A \to aA \mid b \mid \varepsilon$

Sol^n: Let $w = ab$. if there are more than 1 parse tree Then G is ambiguous



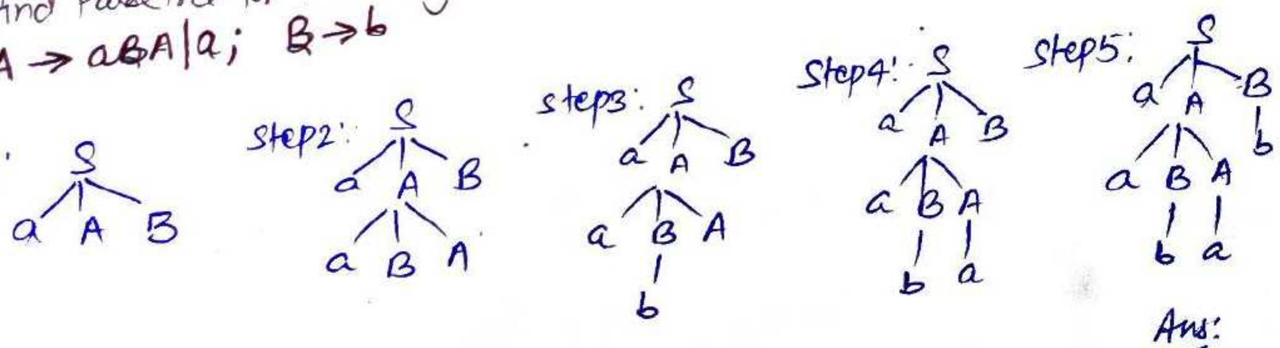
ParseTree 1 | ParseTree 2 | ParseTree 3

We see that there are more than one ParseTree for a String $w \in L(G)$. Therefore given grammar is ambiguous.

Ex:- find Parsetree for the String $w = aabab$ such that $w \in L(G)$, G: $S \to aAB \mid a$; $A \to aBA \mid a$; $B \to b$

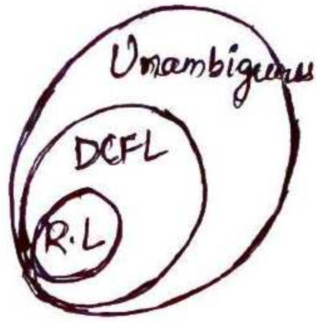
Sol^n:

Step1: | Step2: | step3: | Step4: | Step5:



Ans:

NOTE:-
- Regular grammar → Ambiguous (NFA) / Unambiguous (DFA)
- Regular language always Unambiguous
- DCFL is always Unambiguous language
- Ambiguity start from CFL



Inherent ambiguous language :- if all grammar ambiguous for a language(L) Then the language (L) is Inherently ambiguous language

L → G_1, G_2, G_3, G_4 - - - - G_n

- if all grammar ($G_1, G_2, G_3, G_4 - - - G_n$) are ambiguous Then The language L is Inherently ambiguous language
- if any one grammar ($G_1, G_2, G_3, G_4 \cdots G_n$) is Unambiguous Then The language L is Unambiguous language.
- Inherent word used for language not for Grammar.

i.e Grammar → Ambiguous / Unambiguous   language → Inherently ambiguous / Unambiguous

Ex:- Check wether the given language $L = \{a^n \mid n \geq 0\}$ is a Inherently ambiguous or not

Sol^n: $L = \{a^n \mid n \geq 0\}$ ← Unambiguous Language

$G_1 = \{S \to aS \mid a \mid \varepsilon\}$ Ambiguous Grammar
$G_2 = \{S \to Sa \mid a \mid \varepsilon\}$ Ambiguous Grammar
$G_3 = \{S \to aS \mid \varepsilon\}$ UnAmbiguous Grammar
- - $G_n = \{\ \}$

G_3 is UnAmbiguous Grammar for L Therefore L is not Inherently Ambiguous

Ex:- $L = \{a^m b^n c^k \mid m, n, k \geq 1$, either $m = n$ OR $n = k$

Sol^n: Case 1: if $m = n$ Then $L = \{a^m b^m c^k \mid m, k \geq 1\} = L_1$
Case 2: if $n = k$ Then $L = \{a^m b^n c^n \mid m, n \geq 1\} = L_2$
Now: $L_1 \cap L_2 = \{a^m b^m c^m \mid m \geq 1\}$, Therefore no Unambiguous Grammar for L, all are Ambiguous Grammar, Hence L is Inherently ambiguous language.

Remove Ambiguity:

- Causes such as Left recursion, Common Prefixes etc. make the grammar ambiguous.
- The removal of these causes may convert the grammar into Unambiguous Grammar.
- However, it is not always compulsory.

NOTE:- It is not always Possible to Convert an ambiguous grammar into an Unambiguous grammar b/c Ambiguity finding & Removal both are Undecidable.

Removing Ambiguity by Precedence & Associativity rules:

An ambiguous Grammar may be converted into an Unambiguous grammar by implementing -
- Precedence Constraints
- Associativity Constraints

These Constraints are implemented using the following rules:

Rule-01:- The Precedence Constraint is implemented using the following rules.
- The level at which the Production is Present defines the Priority of the operator contained in it.
- The higher the level of the Production, the lower the Priority of operator.
- The lower the Level of the Production, the higher the Priority of operator.

Rule-02:- The Associativity Constraint is implemented using the following rules.
- If the operator is Left associative, induce Left recursion in its Production.
- If the operator is Right associative, induce right recursion in its Production.

Ex:- $G: E \to E+E \mid E*E \mid id$. Ambiguous Grammar Convert into Unambiguous Grammar.

Sol^n: Ambiguous Grammar $G: E \to E+E$, $E \to E*E$, $E \to id$ $\xrightarrow[\text{Apply Rules 1 \& 2}]{\text{UnAmbiguous G.}}$ UnAmbig Grammar

$$G: E \to E+T \mid T$$
$$T \to T*F \mid F$$
$$F \to id$$

Ans.

Now precedence: $id > * > +$

$+ > +$, $* > *$ } b/c Left Associative

Ex: Convert the following Ambiguous Grammar into Unambiguous Grammar.

$G: R \to R+R \mid R.R \mid R* \mid a \mid b$

Sol^n: $G: R \to R+R$
$R \to R.R$
$R \to R^*$
$R \to a/b$

using the Precedence and Associativity rules, we write the Corresponding Unambiguous grammar as $\Rightarrow$

$E \to E+T \mid T$
$T \to T.F \mid F$
$F \to F* \mid G$
$G \to a/b$

OR

$E \to E+T \mid T$
$T \to T.F \mid F$
$F \to F* \mid a \mid b$

Ans.

Ex:- Convert the following ambiguous grammar into Unambiguous grammar.

bexp → bexp OR bexp | bexp and bexp | not bexp | T | F, where bexp represents Boolean expression, T represents TRUE and F represents FALSE.

Solⁿ: To convert the given grammar into its corresponding Unambiguous grammar, we implement the Precedence and associativity Constraints.

The Priority order is (T, F) > not > and > or

where
- and operator is Left associative
- or operator is left associative

$$\begin{aligned} bexp &\rightarrow bexp \text{ OR } bexp \\ &\mid bexp \text{ and } bexp \\ &\mid \text{not } bexp \\ &\mid TRUE \\ &\mid FALSE \end{aligned} \xRightarrow[\text{Apply Rule 1 \& 2}]{\text{Unambiguous}} \begin{aligned} E &\rightarrow E \text{ or } F \mid F \\ F &\rightarrow F \text{ and } G \mid G \\ G &\rightarrow \text{Not } G \mid T \mid F \end{aligned}$$

Ans UnAmbiguous Grammar

Ex:- Ambiguous Grammar.

+ Left Associative

* Left Associative

$$G \left\{ \begin{aligned} E &\rightarrow E+E \\ E &\rightarrow E*E \\ E &\rightarrow E\uparrow E \\ E &\rightarrow id \end{aligned} \right. \xRightarrow[\text{Apply Rules 1 \& 2.}]{\text{Remove Ambiguity}} \begin{aligned} E &\rightarrow E+T \mid T \\ T &\rightarrow T*F \mid F \\ F &\rightarrow G\uparrow F \mid G \\ G &\rightarrow id \end{aligned}$$

↑ Right Associative

The priority order: id > ↑ > * > +

Associativity: + and * Left Associative and ↑ operator Right Associative

Ex:- Find precedence & Associativity

(i) G_1: $\left\{ \begin{aligned} A &\rightarrow A \$ B \mid B \\ B &\rightarrow B \# C \mid C \\ C &\rightarrow D @ C \mid D \\ D &\rightarrow d \end{aligned} \right.$

(ii) G_2: $\left\{ \begin{aligned} E &\rightarrow E*F \mid F+E \mid F \\ F &\rightarrow F-F \mid id \end{aligned} \right.$

Solⁿ. (i) According to Given Grammar.

@ > # > $

$: Left Associative

: Left Associative

@ : Right Associative

(ii) According to Given Grammar.

(i.e) [* ≐ +] > −

* : Left Associative

+ : Right Associative

− : Both Left & Right Associative

Simplification of CFG

- In CFG, it may happen that all the production rules and symbols are not needed for the derivation of strings.
  Elimination of these productions & symbols is called **simplification of CFG**
- It consists of three steps
  1. Removal of Useless Production
  2. Removal of Unit Production
  3. Removal of NULL Production.

Removal of Useless Production

- Variables/Non-Terminals which don't derive any string are called as useless symbols.
- The production rule generating useless symbol becomes useless production.
- eg 1: $\{ S \to AB \mid a \;;\; A \to BC \mid b \;;\; B \to eD \mid e \}$

  In above grammar useless productions are

  $A \to BC$

  $B \to eD$

  Because Variable 'C' & 'D' are not deriving any string. Hence after removal of useless product^n We have grammar as

  $\{ S \to AB \mid a \;;\; A \to b \;;\; B \to e \}$

eg 2: $L(G)$; $\{ S \to AB \;;\; A \to aA \mid bC \;;\; B \to bB \mid a \;;\; D \to e \}$

In above grammar useless productions are

$A \to bC$

$D \to e$.

Variable C is not deriving any string.
Variable D is not generated in any of the product^n
We have grammar after removing useless product^n as

$\{ S \to AB$; A [illegible] $\to bB \mid a \}$

Removal of Unit Production

- Any rule in the form <Variable> $\rightarrow$ <variable> is called unit product
  $X \rightarrow Y$
- Substitute all Y product if y is generating any terminal symbol.
- e.g 1. $\{S \rightarrow A \; ; \; A \rightarrow a|b\}$

  substitute variable A in the production $S \rightarrow A$ with the terminal 'a' & 'b' because $A \rightarrow a$; $A \rightarrow b$

  After removing unit production we have

  $\{S \rightarrow a|b\}$

e.g 2. $\{S \rightarrow AB \; ; \; A \rightarrow a|b \; ; \; B \rightarrow A\}$

Rule $B \rightarrow A$ is the unit production.

We can substitution A with 'a' & 'b'

∴ After removing unit production. We have,

$\{S \rightarrow AB \; ; \; A \rightarrow a|b \; ; \; B \rightarrow a|b\}$

e.g. 3 $\{S \rightarrow A|bb \; ; \; A \rightarrow B|b \; ; \; B \rightarrow S|a\}$

Unit productions in above grammar are

$S \rightarrow A$	; substitute	A	with	b & B	$S \rightarrow b \mid B$
$A \rightarrow B$	; "	B	"	a & S	$A \rightarrow a \mid S$
$B \rightarrow S$	; "	S	"	bb & A	$B \rightarrow bb \mid A$

After subtituting values we again get unit production

$S \rightarrow B$	; substitute	B	with	a & S	$S \rightarrow a \mid S$
$A \rightarrow S$	; "	S	"	bb & A	$A \rightarrow bb \mid A$
$B \rightarrow A$	; "	A	"	b & B	$B \rightarrow b \mid B$

Again we get unit production

$S \rightarrow S$	; substitute	S	with	bb & A	$S \rightarrow bb \mid A$
$A \rightarrow A$	; "	A	"	b & B	$A \rightarrow b \mid B$
$B \rightarrow B$	; "	B	"	a & S	$B \rightarrow a \mid S$

Grammar we get

$\{S \rightarrow a|b|bb$ ... $b|bb$; $B \rightarrow a|b|bb\}$

Removal of Null Production

- Any production of the form $X \rightarrow \varepsilon$ should be removed
  (variable)
- Removal Procedure
  - ↳ find all variables which derive ε. eg $X \rightarrow \varepsilon$
  - ↳ If variable X is present in any production then remove X ~~and~~ from the production.
  - ↳ combine original productions with above step & remove ε- productions.

eg 1. $L(G) : \{ S \rightarrow aA \mid bBA \mid AB \;;\; A \rightarrow aA \mid \varepsilon \;;\; B \rightarrow b \}$

There is one NULL production in above grammar i.e $A \rightarrow \varepsilon$

Variable A is present in 4 production rules i.e ($S \rightarrow aA \mid bBA \mid AB \;;\; A \rightarrow aA$). Remove A from the product^n

$S \rightarrow a \mid bB \mid B \;;\; A \rightarrow a$

Now combine original productions with above step & remove ε- productions. We have

$L(G') : \{ S \rightarrow aA \mid bBA \mid AB \mid a \mid bB \mid B \;;\; A \rightarrow aA \mid a \;;\; B \rightarrow b \}$

Ans

eg 2. $L(G) : \{ S \rightarrow ASA \mid aB \mid b \;;\; A \rightarrow B \mid \varepsilon \;;\; B \rightarrow b \mid \varepsilon \}$

There are two NULL productions in above grammar $A \rightarrow \varepsilon$ & $B \rightarrow \varepsilon$

Variable A is present in one production rule i.e $S \rightarrow ASA$
Variable B is present in two production rules i.e $S \rightarrow aB \;;\; A \rightarrow B$
After removing A & B from product^n respectively we get

$S \rightarrow AS \mid SA \mid S \;;\; S \rightarrow a \;;\; A \rightarrow b$

Now combine original productions with above step & remove ε- production, we have

$L(G') : \{ S \rightarrow ASA \mid aB \mid b \mid AS \mid SA \mid S \mid a \;;\; A \rightarrow B \mid b \;;\; B \rightarrow b \}$

Ans

Chomsky Normal Form (CNF)

- A CFG is in CNF if the productions are in following forms

<Variable> $\rightarrow$ < Terminal >

<variable> $\rightarrow$ <variable><variable>

$S \rightarrow \epsilon$ (start symbol can generate ϵ)

ex1: $A \rightarrow a$
$A \rightarrow AB$

ex2: $S \rightarrow AB \mid \epsilon$
$A \rightarrow a$
$B \rightarrow b$

ex3: $A \rightarrow BC \mid a$
$B \rightarrow CD \mid b$
$C \rightarrow e$
$D \rightarrow d$

ex4: $S \rightarrow BA \mid a$
$B \rightarrow a$
$A \rightarrow b$

ex5: $S \rightarrow AB \mid BD$
$A \rightarrow CD \mid a \mid AC$
$C \rightarrow DE \mid e$
$B \rightarrow b$
$D \rightarrow d$
$E \rightarrow f$

Algo to convert CFG into CNF

1. If start symbol occurs on R.H.S of production rule, create a new start symbol S' & a new production $S' \rightarrow S$
2. Remove NULL production
3. Remove UNIT production
4. If R.H.S of production contains more than two variable then two consecutive variables can be replaced by a new variable.

ex: $A \rightarrow XYZ \Rightarrow A \rightarrow XP$, $P \rightarrow YZ$ or $A \rightarrow PZ$, $P \rightarrow XY$

Here, we can replace either YZ or XY with new variable

5. If R.H.S of production contains combination of Terminal & Variable then each terminal symbol is replaced by new variable.

ex. i) $A \rightarrow aB \Rightarrow A \rightarrow XB$ (CNF form)
$X \rightarrow a$ "

ii) $A \rightarrow BCDE \Rightarrow A \rightarrow BX$ (CNF form)
$X \rightarrow CY$ "
$Y \rightarrow DE$ "

{Here also we can take any consecutive two variables & replace with with new var.

Q1. Construct equivalent CNF for given CFG

a) $S \rightarrow aSa \mid bSb \mid a \mid b$

Sol^n: Since S appears in R.H.S, we add new product^n, we have

$S' \rightarrow S$ (Not in CNF)
$S \rightarrow aSa$ "
$S \rightarrow bSb$ "
$S \rightarrow a$ (CNF)
$S \rightarrow b$ (CNF)

NO NULL/UNIT production.
We find a production $S \rightarrow aSa$ & $S \rightarrow bSb$ where R.H.S contains combination of variable & terminal. Apply step (5)

We have $S \rightarrow AX$ & $S \rightarrow BY$
$X \rightarrow SA$ $\quad$ $Y \rightarrow SB$
$A \rightarrow a$ $\quad$ $B \rightarrow b$

final production set we obtain is as follows

$\{ S' \rightarrow AX \mid BY$; $S \rightarrow AX \mid BY \mid a \mid b$
$X \rightarrow SA$
$Y \rightarrow SB$
$A \rightarrow a$
$B \rightarrow b$
$\}$

b.) $S \rightarrow ASA \mid aB$; $A \rightarrow B \mid S$; $B \rightarrow b \mid \varepsilon$ $S' \rightarrow S$ because S appears in R.H.S

Sol^n: We add new production
$S' \rightarrow S$
$S \rightarrow ASA$ (Not in CNF)
$S \rightarrow aB$ "
$A \rightarrow B$ (UNIT product^n, Not in CNF)
$A \rightarrow S$ ("
$B \rightarrow b$ (CNF)
$B \rightarrow \varepsilon$ (NULL product^n, Not in CNF)

After Removing NULL production $B \rightarrow \varepsilon$, the product^n set becomes

$S' \rightarrow S$
$S \rightarrow ASA \mid AS \mid SA \mid S$
$S \rightarrow aB \mid a$
$A \rightarrow B \mid S$
$B \rightarrow b$

{ After removing $B \rightarrow \varepsilon$, we have
$S' \rightarrow S$; $S \rightarrow ASA \mid aB \mid a$; $A \rightarrow B \mid S \mid \varepsilon$; $B \rightarrow b$
Now, Removing $A \rightarrow \varepsilon$, we have set of production as }

Now remove unit productions

After removing $S \to S$, the product^n set becomes

$S' \to S$; $S \to ASA | AS | SA | aB | a$; $A \to B | S$; $B \to b$

After removing $A \to B$, the product^n set becomes

$S' \to S$; $S \to ASA | AS | SA | aB | a$; $A \to S | b$; $B \to b$

After removing $S' \to S$ & $A \to S$, the product^n set becomes

$S' \to ASA | aB | a | AS | SA$

$S \to ASA | aB | a | AS | SA$

$A \to ASA | aB | a | AS | SA | b$

$B \to b$

Now find more than two variables in the product^n violates CNF form.
Here $S' \to ASA$, $S \to ASA$ & $A \to ASA$
using step (4) we have production set

$S' \to AX | aB | a | AS | SA$

$S \to AX | aB | a | AS | SA$

$A \to AX | aB | a | AS | SA | b$

$B \to b$

$X \to SA$

using set (5) fix the problem in R.H.S for the product^n containing combination of variable & terminal.
final production set becomes

$$\boxed{\begin{aligned}
\{\; & S' \to AX | YB | a | AS | SA \\
& S \to AX | YB | a | AS | SA \\
& A \to AX | YB | a | AS | SA | b \\
& B \to b \\
& X \to SA \\
& Y \to a \\
\}\; &
\end{aligned}}$$

Ans.

c) $S \to bA \mid aB$; $A \to bAA \mid aS \mid a$; $B \to aBB \mid bS \mid b$

Sol^n Create New start symbol S' & new product^n $S' \to S$. We have

$S' \to S$ (UNIT product^n)

$S \to bA$ (Not in CNF)

$S \to aB$ "

$A \to bAA$ "

$A \to aS$ "

$A \to a$ (CNF)

$B \to aBB$ (Not in CNF)

$B \to bS$ "

$B \to b$ (CNF)

There is no NULL/UNIT production except $S' \to S$ & more than two variables After removing combination var & Terminals, We have final set of production rules

$S' \to XA \mid YB$

$S \to XA$

$S \to YB$

$A \to ZA$

$A \to YS$

$A \to a$

$B \to WB$

$B \to XS$

$B \to b$

$X \to b$

$Y \to a$

$Z \to XA$

$W \to YB$

No: of productions (max^m)

$= (K-1) + |P| + |T|$

$= (5-1) + 8 + 2$

$= 4 + 8 + 2$

$= 14$

Gate exam.

Note: Let G be CFG without NULL production & UNIT product^n

'K' be the max^m no: of symbols on R.H.S of any product^n, then Equivalent CNF contains max^m no: of product^n

$= (K-1) + |P| + |T|$

no: of symbols — production — Terminal

GREIBACH NORMAL FORM (GNF)

- A CFG is in GNF if the productions are in the following forms

<variable> $\rightarrow$ <Terminal>

<Variable> $\rightarrow$ <single Terminal> <Variables>

$S \rightarrow \varepsilon$

Start symbol

ex 1:

$A \rightarrow a$

$B \rightarrow aBB \mid a$

ex 2:

$A \rightarrow aABBA \mid b$

$B \rightarrow a$

ex 3:

$S \rightarrow aAABBB$

$A \rightarrow a$

$B \rightarrow b$

ex 4:

$S \rightarrow bAAAABBB$

$A \rightarrow a$

$B \rightarrow b$

- ALGO TO CONVERT CFG INTO GNF

1. If start symbol S occurs in any of R.H.S rule, create a new start symbol S' & add new production $S' \rightarrow S$.
2. Remove Null productions
3. Remove Unit productions
4. Remove all direct and indirect **left-recursion**
5. Do proper substitutions of productions
   a.) Convert CFG to CNF
   b.) Rename all variables as $A_1, A_2, A_3 \ldots\ldots A_n$
   c.) For every production A_i do following
      (i) If $A_i \rightarrow A_j X$; $i < j$ then apply substitution of A_j
      (ii) If $A_K \rightarrow A_K X$; then Remove left recursion] include this if step (4) not included in algo

Q1. Convert given CFG into equivalent GNF

a.) $S \rightarrow XY \mid Xn \mid p$

$X \rightarrow mX \mid m$

$Y \rightarrow Xn \mid o$

Sol^n: Here S doesn't appear in any of R.H.S product^n.

There are no NULL/UNIT production.

Using step ⑤ a) convert CFG to CNF

We have set of production

$S \rightarrow XY \mid XN \mid p$

$X \rightarrow mX \mid m$

$Y \rightarrow XN \mid o$

$N \rightarrow n$

Apply step ⑤ c) Apply substitution after renaming all var.

$$\left.\begin{aligned} A_1 &\rightarrow A_2A_3 \mid A_2A_4 \mid p \\ A_2 &\rightarrow mA_2 \mid m \\ A_3 &\rightarrow A_2A_4 \mid o \\ A_4 &\rightarrow n \end{aligned}\right\} \text{Rename variables}$$

In $A_1 \rightarrow A_2A_3 \mid A_2A_4$ we can substitute value of A_2. Also in $A_3 \rightarrow A_2A_4$ we can substitute $A_2 \rightarrow mA_2 \mid m$.

final set of production is as follows.

$$\boxed{\begin{aligned} A_1 &\rightarrow mA_2A_3 \mid mA_3 \mid mA_2A_4 \mid mA_4 \mid p \\ A_2 &\rightarrow mA_2 \mid m \\ A_3 &\rightarrow mA_2A_4 \mid mA_4 \mid o \\ A_4 &\rightarrow n \end{aligned}}$$

Ans

b.) $B \to AB|AC|DA$; $A \to a$; $C \to DA$; $D \to d$

Sol

Given CFG has start symbol 'B' which is present in the R.H.S of the production also. Therefore create new start symbol B' & new production $B' \to B$. The set of production we have as

$B' \to B$; $B \to AB|AC|DA$; $A \to a$; $C \to DA$; $D \to d$

There are no NULL/UNIT production.
There is no left recursion in the given grammar.
Convert CFG into CNF. We have set of production as

$B' \to AB|AC|DA$; $B \to AB|AC|DA$; $A \to a$; $C \to DA$; $D \to d$

Now substitute all variables with new name

$A_1 \to A_2A_3|A_2A_4|A_5A_2$; $A_3 \to A_2A_3|A_2A_4|A_5A_2$; $A_2 \to a$; $A_4 \to A_5A_2$
$A_5 \to d$

Apply substitution using step ⑤ c.)(i)

$A_1 \to aA_3|aA_4|dA_2$

$A_3 \to aA_3|aA_4|dA_2$

$A_2 \to a$

$A_4 \to dA_2$

$A_5 \to d$

Ans.

Elimination of Left Recursion

If production is of form $A \to AX|t$ that means left recursion is present.

Variable

Terminal

same symbol

Recursion is present due to Variable A because variable in L.H.S is equal to first variable in R.H.S

To remove left recursion create new symbol variable A' & new production like below

A->tA' | t (Replace A with t, X with new variable A' | t is a terminal)

A'->XA'|X (New production for new variable A')

NOTE: variable X must derive a terminal, otherwise left recursion will be there in new production too.