Unit-3

Context Free Grammars

Content free grammar (CFG):

A CFG is a grammar (G) is defined as 4-tuple notation i.e $G = (V, T, P, S)$

where V is a set of variables or non-terminals

T is a terminals

P is Production rules or grammar rules of the form $\alpha \rightarrow \beta$

$\alpha \in V$

$\beta \in (VUT)^*$

S -> Start symbol

eg:- Identify the grammar G for the following P

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

$G = (\{E, T, F\}, \{+, *, (,), id\}, E \rightarrow E+T \mid T, T \rightarrow T*F \mid F, F \rightarrow (E) \mid id, \{E\})$

eg:- $E \rightarrow E+E \mid E*E \mid (E) \mid id$

$G = (\{E\}, \{+, *, (,), id\}, E \rightarrow E+E, E \rightarrow E*E, E \rightarrow (E), E \rightarrow id, \{E\})$

Derivation:-

Derivation is a Process of deriving strings from start symbol of the grammar using Production rules or grammar rules.

$$S \xrightarrow{*} w$$

Eg:- Consider the following Production rules

$E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid id$

To derive the string w is id * id + id

$E \to E+T$ [∵ $E \to T$]

$E \to T+T$ [∵ $T \to T*F$]

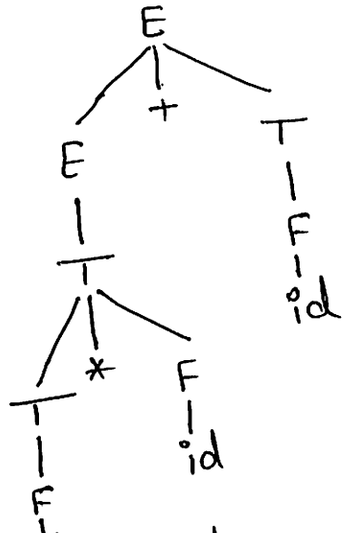
$E \to T*F+T$ [∵ $T \to id$]

$E \to id*F+T$ [∵ $F \to id$]

$E \to id*id+T$ [∵ $T \to F$]

$E \to id*id+F$ [∵ $F \to id$]

$E \to id*id+id$



Derivation can be classified into two types.

1) Left most derivation (LMD) or Left Sentential form
2) Right most derivation (RMD) or Right Sentential form

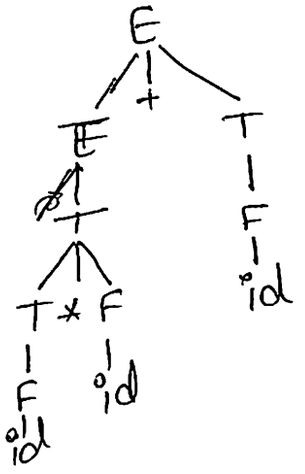
Left most derivation:

A derivation in which left most variable is replaced at every step of the derivation Process

Right most derivation:

A derivation in which right most variable is replaced at every step of the derivation process.

$E \to E+T$
$\to E+F \quad \because T \to F$
$E+id \quad \because F \to id$
$T+id \quad \because E \to T$
$T*F+id \quad \because T \to T*F$
$T*id+id \quad \because F \to id$
$F*id+id \quad \because T \to F$
$id*id+id \quad \because F \to id$



Derivation Tree (or) Parse Tree:

It is a graphical representation of the derivation process.

(or)

Representation of string derivation in the form of tree is called Parse Tree.

It is a simple way to show how the derivation can be done using production rules.

Derivation tree properties:

1) Every node is labelled by either variable or terminal or ϵ.
2) Root node is labelled by start symbol of the grammar i.e S
3) Every internal node or interior node is labelled by non-terminals of the grammar.

4) Every leaf node is labelled by either terminals or $\in$.
5) The leaf nodes are also called as terminal nodes.
6) The derivation can be read from left to right direction.
7) The derivation tree can be classified into 2 types
   → left most derivation tree
   → Right most derivation tree

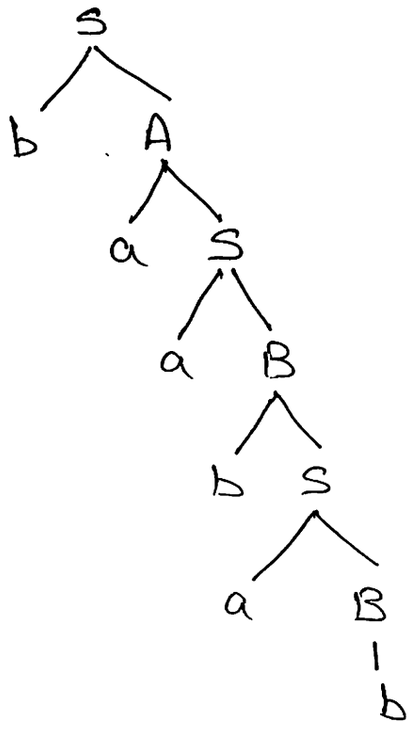
Consider the following grammar

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

1) Derive the string $\omega = baabab$
2) Construct derivation tree for the above string

Sol:

$S \rightarrow bA$ ($\because S \rightarrow bA$)
$S \rightarrow baS$ ($A \rightarrow aS$)
$S \rightarrow baaB$ ($\because S \rightarrow aB$)
$S \rightarrow baabS$ ($\because B \rightarrow bS$)
$S \rightarrow baabaB$ ($\because S \rightarrow aB$)
$S \rightarrow baabab$ ($\because B \rightarrow b$)



— find the LMD & RMD and LMDT & RMDT for

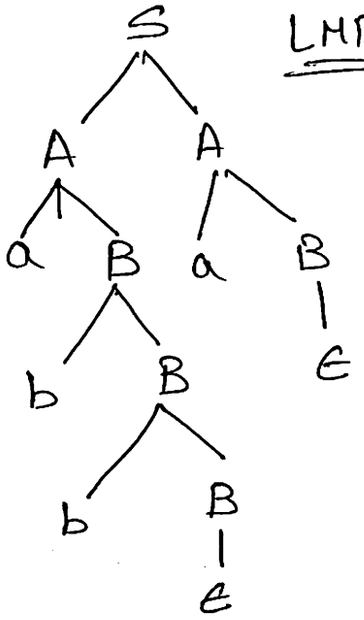
1) $S \to AA$
$A \to aB$ ω: abba
$B \to bB \mid \epsilon$

2) $S \to 0B \mid 1A$
$A \to 0 \mid 0S \mid 1AA$
$B \to 1 \mid 1S \mid 0BB$
ω: 00110101

Sol LMD for abba

$S \to AA$ ($A \to aB$)
$\to aBA$ ($B \to bB$)
$\to abBA$ ($B \to bB$)
$\to abbBA$ ($B \to \epsilon$)
$\to abb\epsilon A$ ($A \to aB$)
$\to abbaB$ ($B \to \epsilon$)
$\to abba\epsilon$
$\therefore$ abba //

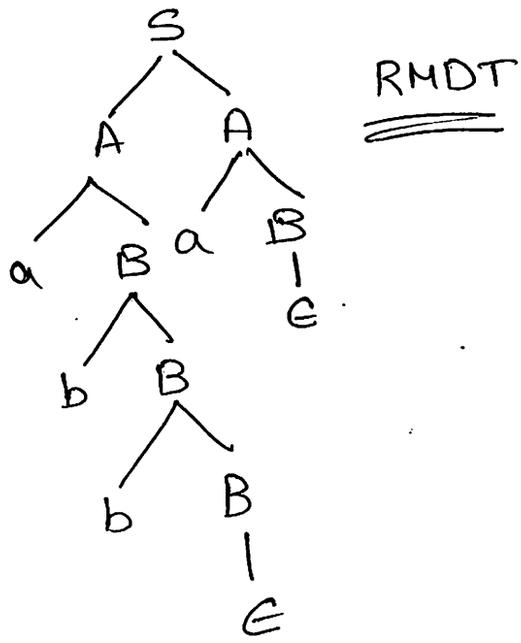
LMDT



RMD:-

$S \to AA$ ($\because A \to aB$)
$S \to AaB$ ($\because B \to \epsilon$)
$S \to Aa\epsilon$ ($A \to aB$)
$S \to aBa$ ($B \to bB$)
$S \to abBa$ ($\because B \to bB$)
$S \to abbBa$ [$\because B \to \epsilon$)
$\to abba$ //

RMDT



2Sol $S \to 0B|1A$

$A \to 0|0S|1AA$ $\quad w: 00110101$

$B \to 1|1S|0BB$

LMD

$S \to 0\underline{B}$ $(\because B \to 0BB)$

$\to 00\underline{BB}$ $(\because B \to 1S)$

$\to 001\underline{SB}$ $(\because S \to 1A)$

$\to 0011\underline{AB}$ $(\because A \to 0S)$

$\to 00110\underline{SB}$ $(\because S \to 1A)$

$\to 001101\underline{AB}$ $(A \to 0)$

$\to 0011010\underline{B}$ $(\because B \to 1)$

$\to 00110101$ //

RMD:

$S \to 0\underline{B}$ $(B \to 0BB)$

$S \to 00B\underline{B}$ $(B \to 1)$

$S \to 00\underline{B}1$ $(\because B \to 1S)$

$\to 001\underline{S}1$ $(\because S \to 1A)$

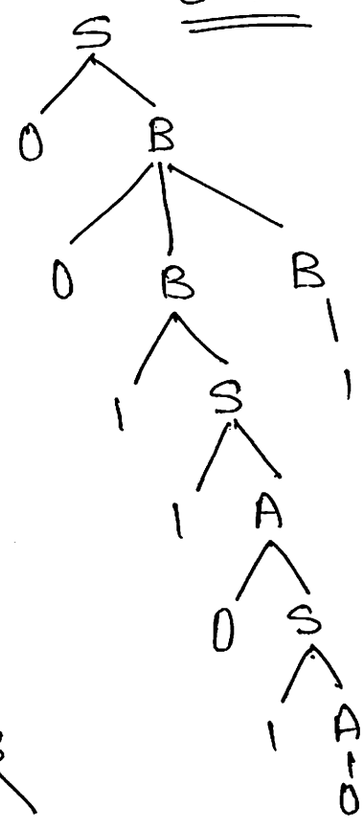
$\to 0011\underline{A}1$ $(\because A \to 0S)$

$\to 00110\underline{S}1$ $(\because S \to 1A)$

$\to 001101\underline{A}1$ $(\because A \to 0)$

$\to 00110101$ //

LMDT



Language of a Context free grammar (CFG):

Find out the language for the following CFG:

1) $S \to aSa \mid \in$

$S \to \in$ | $S \to aSa$ $[S \to \in]$ $\to a\in a$ $\to aa$ | $S \to aSa$ $[S \to aSa]$ $\to aaSaa$ $[S \to \in]$ $\to aa\in aa$ $\to aaaa$

$\therefore L = \{\in, aa, aaaa, \ldots\}$

i.e $L = (aa)^*$

2) $S \to aSb \mid ab$

$S \to ab$ | $S \to aSb$ $[S \to ab]$ $\to aabb$ | $S \to aSb$ $[S \to aSb]$ $\to aaSbb$ $[S \to ab]$ $\to aaabbb$

$\therefore L = \{ab, aabb, aaabbb, \ldots\}$

i.e $L = \{a^n b^n \mid n > 0\}$

3) ~~$S \to aCa$~~

$S \to aCa$

$C \to aCa \mid b$

$S \to aCa$ $[C \to b]$ aba | $S \to aCa$ $[C \to aCa]$ $\to aacaa$ $[C \to b]$ $\to aabaa$

$\therefore L = \{aba, aabaa, \ldots\}$

$L = \{a^n b a^n \mid n > 0\}$

4) $S \to aAB$
$A \to bBb$
$B \to A/\epsilon$

$L = \{abb, abbbb, abbbbbb, \cdots\}$
$= a(bb)^+$

5) $S \to aB/bA$
$A \to a/aS/bAA$
$B \to b/bS/aBB$

$L = \{ab, ba, abab, baba, \cdots\}$
$= \{n_a(w) = n_b(w) \mid w \in (a,b)^+\}$

6) $S \to 0A$
$A \to 0AB/0A/1$
$B \to 1$

$S \to 0A[A \to 1]$ | $S \to 0A[A \to 0A]$ | $S \to 0A[A \to 0AB]$
01 | $00A(A \to 1]$ | $00AB[A \to 1]$
| 001 | $001B[B \to 1]$
| | 0011

$\therefore L = \{01, 001, 0011, 00011$有, $000111 \cdots\}$

$L = \{0^{m,n} \mid m \geq n, m, n > 0\}$

Applications of context free grammars:

→ In early application, grammars are used to describe the structure of Programming languages.

→ In newer application, they are used in an essential part of the Extensible markup language (XML) called the 'Document type Definition'.

→ In most Programming languages opening & closing of braces, curly brackets is taken care.

Ambiguity in CFG:

Ambigous grammar:-

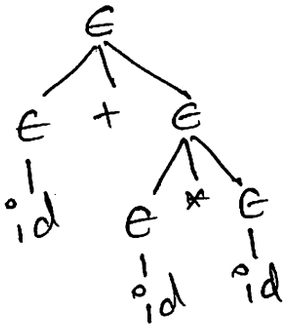
A grammar is said to be ambigous if there exist more than one Parse tree (or) derivation tree for some string generated by the language.

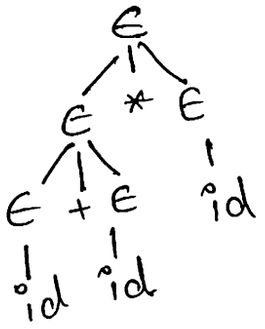
•) Check whether the following grammar is ambigous grammar or not.

$E \to E+E \mid E*E \mid (E) \mid id$ w: id+id*id & id*id*id

$E \to E+E$ [E→id]
$id+E$ [E→E*E]
$id+E*E$ [E→id]
$id+id*E$ [E→id]
$id+id*id$

$E \to E*E$ [E→E+E]
$\to E+E*E$ [E→id]
$\to id+E*E$ [E→id]
$\to id+id*E$ [E→id]
$\to id+id*id$





As the grammar is having more than one Parse tree it is ambigous grammar.

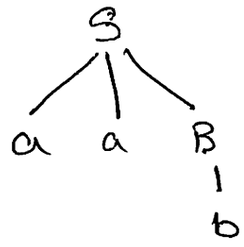
→ find whether the following grammar is ambigous or not

$S \to AB \mid aaB$
$A \to a \mid Aa$
$B \to b$

Sol Consider the production for string aab

$S \to aaB$ ∴ $B \to b$ $S \to aaB$

$S \to aab$ ∴ $B \to b$

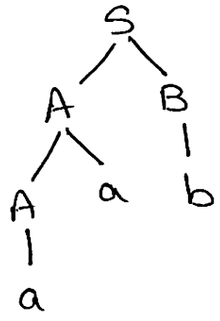


$S \to AB$

$S \to AaB$ ∴ $A \to Aa$

$S \to aaB$ ∴ $A \to a$

$S \to aab$ ∴ $B \to b$



∴ It is a ambigous grammar

→ Construct CFG which generates on $(0+1)^*$

Sol $S \to 0S|1S$

$S \to \epsilon$

→ Construct CFG which generates set of palindrome strings over {a,b}

$S \to asa$

$S \to bSb$

$S \to a|b|\epsilon$

i/p aabbaa

$S \to aSa$

$S \to aaSaa$ ∴ $S \to aSa$

$S \to aabSbaa$ ∴ $S \to bSb$

$S \to aab\epsilon baa$ ∴ $S \to \epsilon$

$\therefore$ aabbaa//

a
b
aa
bb
aba
bab

Write a CFG which generates equal no. of a's & b's.

$S \to SaSbS$

$S \to SbSaS$

$S \to \epsilon$

→ check whether the following grammar is ambigous or not.

1) $S \to S(S) \mid \epsilon$ — (), ()()

2) $S \to AA$
   $A \to AAA \mid a \mid bA \mid Ab$ i/p - aba & baa

3) $S \to iCts \mid iCtsES \mid a$ - Ambigious
   $C \to b$

4) $S \to aSb \mid aAb \mid aBb$
   $A \to aA \mid a$
   $B \to Bb \mid b$

Minimization of CFG:-

In Context free grammar it may not be necessary to use all the symbols or all Productions in 'P' for deriving sentences

A grammar may have Extra symbols or unnecessary symbols or Productions which increases the length of the grammar, hence we can eliminate such symbols

In order to minimize the CFG we can follow the below steps:

Step1: Elimination of useless symbols

Step2: Elimination of ϵ- Productions $[A \to \epsilon]$

Step3: Elimination of Unit Productions $[A \to B]$

Elimination of useless symbols:-

A variable is said to be useless if it cannot derive a terminal string or if it cannot be derived from starting symbol.

eg:- $S \to aS \mid A \mid C$
$A \to a$
$B \to aa$
$C \to acb$

→ As $B \to aa$, B is not derived from starting symbol it is useless. so we can remove it.

→ $C \to acb \Rightarrow aaacbb \Rightarrow aaacbbb$

Therefore C is derived from starting symbol, but it cannot derive a terminal string. Hence we can remove $C \to acb$ Production.

∴ $S \to aS \mid A$
$A \to a$

→ Eliminate the useless symbols from the following grammar.

$S \to AB \mid CA$
$B \to BC \mid AB$
$A \to a$
$C \to aB \mid b$

$S \to AB \mid CA$ shows A, B, C are derived from starting symbol

$A \to a$: A gives terminal string, hence A is useful symbol.

$B \to BC$: $B \to Bb$ ∵ $C \to b$
$B \to ABb$ ∵ $B \to AB \Rightarrow B \to aBb$ ∵ $A \to a$

Hence we cannot derive terminal string B is useless symbol. Therefore we remove all B's production from the grammar.

$C \to b$: C derives terminal string

$\therefore S \to CA$
$A \to a$
$C \to b$

→ Eliminate the useless symbols from the following grammar.

$S \to ABa \mid BC$
$A \to aC \mid BCC$
$C \to a$
$B \to bCC$
$D \to E$
$E \to d$
$F \to e$

Sol: In the above grammar A, B, C are derived from starting symbol 'S' where as D, E, F are not derived.

$\therefore S \to ABa \mid BC$
$A \to aC \mid BCC$
$C \to a$
$B \to bCC$

Elimination of ϵ-productions:

① In a given CFG a non-terminal or variable 'X' is said to be nullable if there exists a Production of the form $X \to \epsilon$.

② If there is a derivation starts with X

$X \Rightarrow X_1 \Rightarrow X_2 \Rightarrow \cdots \epsilon$ i.e $X \overset{*}{\Rightarrow} \epsilon$

Elimination of ϵ :-

- Construct V_n with set of nullable variables
- → For each Production at the right hand side replace nullable variable by ϵ & add all possible Combinations.
- → Do not include ϵ-Productions (or) $X \to \epsilon$

Eliminate ϵ-Productions from the following grammar.

$S \to aA$
$A \to BB$
$B \to aBb \mid \epsilon$

Sol: Consider ϵ Productions

$B \to \epsilon$
$A \to BB \Rightarrow \epsilon$
$\therefore V_n = \{A, B\}$

$S \to aA \mid a$
$A \to BB$
$B \to ab \mid aBb$ //

② Eliminate ϵ-Productions from the following grammar

$S \to ABaC$
$A \to BC$
$B \to b|\epsilon$
$C \to D|\epsilon$
$D \to d$

Sol: $B \to \epsilon$
$C \to \epsilon$
$A \to BC \Rightarrow \epsilon$

$\therefore V_n = \{A, B, C\}$

$S \to ABaC \mid BaC \mid AaC \mid ABa \mid ac \mid Aa \mid Ba \mid a$
$A \to BC \mid B \mid C$
$B \to b$
$C \to D$
$D \to d$ //

③ $S \to AB$
$A \to aA|\epsilon$
$B \to b$

$A \to \epsilon$, $V_n = \{A\}$

$S \to AB \mid B$
$A \to aA \mid a$
$B \to b$ //

④ $S \to ABAC$
$B \to bB|\epsilon$
$A \to aA|\epsilon$
$C \to c$

$\Rightarrow$ $B \to \epsilon$
$A \to \epsilon$
$V_n = \{A, B\}$

$S \to ABAC \mid BAC \mid AAC \mid AB$
$\Rightarrow \mid AC \mid c \mid BC$
$B \to bB \mid b$
$A \to aA \mid a$
$C \to c$ //

Elimination of Unit Productions :-

A Unit Production is a Production of the form $A \to B$. Assume that there are no ϵ-Productions

'A' after finite no. of steps gives $B \to A \overset{*}{\Rightarrow} B$ is a unit production and $B \to w_1 w_2 \dots w_n$ then while removing unit Production A we include $A \to w_1 w_2 \dots w_n$ where $w_1, w_2, \dots w_n$ are termina

① Eliminate Unit Productions from the following grammar.

$S \to A \mid bb$
$A \to B \mid b$
$B \to S \mid a$

Sol :- $S \to A \Rightarrow S \to b \quad \because A \to b$
$A \to B \Rightarrow A \to a \quad \because B \to a$
$B \to S \Rightarrow B \to bb \quad \because S \to bb$

$\therefore$ $S \to a \mid b \mid bb$
$A \to a \mid b \mid bb$
$B \to b \mid a \mid bb$

② Eliminate Unit Productions from the following grammar

$S \to Aa \mid B$
$B \to A \mid bb$
$A \to a \mid bc \mid B$

Sol:

$S \to B$

$S \to bb \Rightarrow S \to Aa \mid bb$

$B \to A$

$B \to a \mid bc \Rightarrow B \to a \mid bc$

$A \to B$

$A \to a \mid bc \Rightarrow A \to a \mid bc \mid a \mid bc$

$\Rightarrow A \to a \mid bc$

$\therefore S \to Aa \mid bb \mid a \mid bc$

$B \to a \mid bc \mid bb$

$A \to a \mid bc$ //

(3) CFG: $S \to 0A \mid 1B \mid C$

$A \to 0S \mid 00$

$B \to 1 \mid A$

$C \to 01$

$S \to C \quad \therefore C \to 01$

$S \to 01$

$\therefore S \to 0A \mid 1B \mid 01$

$B \to A \quad \therefore A \to 0S \mid 00$

$B \to 0S \mid 00$

$\therefore B \to 1 \mid 0S \mid 00$

Therefore CFG is

$S \to 0A \mid 1B \mid 01$

$A \to 0S \mid 00$

$B \to 1 \mid 0S \mid 00$

$C \to 01$ //

(4) → Eliminate Unit Productions

$S \to AB$
$A \to a$
$B \to C \mid b$
$C \to D$
$D \to E \mid bC$
$E \to d \mid Ab$

Sol: $B \to C \quad \therefore C \to D$

$C \to D \quad \therefore D \to E \mid bC$

$C \to E \mid bC$

$C \to E \quad \therefore E \to d \mid Ab$

$\therefore C \to d \mid Ab \mid bC$

$\therefore B \to d \mid Ab \mid bC \mid b$

$D \to E \quad \therefore E \to d \mid Ab$

$\therefore D \to d \mid Ab \mid bC$

After removing unit Productions

$S \to AB$
$A \to a$
$B \to d \mid Ab \mid bC \mid b$
$C \to d \mid Ab \mid bC$
$D \to d \mid Ab \mid bC$
$E \to d \mid Ab$ //

⑥ Simplify the grammar

$G = \{\{S, A, B, C, E\}, \{a, b, c\}, P, S\}$ where P is

$S \to AB$
$A \to a$
$B \to b$
$B \to C$
$E \to c \mid \wedge$

Sol: For simplifying the grammar

1) we must eliminate null Productions
2) we must remove unit Productions
3) we must remove useless symbols

1) There is no null Productions

2) $E \to c \mid A$ is not derived from starting symbol 'S' hence E can be eliminated

3) $B \to C$ is a unit production. There is no rule for deriving C hence we can eliminate it

∴ Simplified grammar is

$S \to AB$
$A \to a$
$B \to b$ //

Normal Forms:

At the right hand side of production there are any no. of terminals & non-terminals. If we want grammar in some specific format i.e there should be some fixed no. of variables & terminals in context free grammar.

Grammar with these conditions are called Normal forms. There are two types of Normal forms:

1. Chomsky Normal form (CNF)
2. Greibach Normal form (GNF)

Chomsky Normal form (CNF):-

A Context free grammar without $\in$ Productions and also of the form $NT \to NTNT$ or $NT \to T$ is called chomsky normal form.

Eg:- $A \to BC$ (Non Terminal → Exact 2 non terminals)

or

$A \to a$ (Non-Terminal → terminal)

Conversion of CFG to CNF:-

step1: Eliminate null, unit and useless Productions

step2: Include Production of the form

$A \to BC|a$

Step 3: Eliminate strings of terminals on the RHS of the Production if they are of the form $S \to a_1 a_2 a_3 \cdots a_n$ where $a_1, a_2 \ldots$ are terminals then add Productions as

$Ca_1 \to a_1$
$Ca_2 \to a_2$
$\vdots$
$Ca_n \to a_n$

Step 4: To restrict the no. of non-terminals on the right hand side we introduce new variables | NT & seperate them as follows

Consider the Production $A \to A_1 A_2 \cdots A_n$. Can be replaced by

$A \to A_1 B$
$B \to A_2 C$
$C \to A_3 D$
$\vdots$

1) Convert the following grammar into CNF:

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

$G = (\{S, A, B\}, \{a, b\}, P, S)$

Step1: NO ϵ-Productions, unit productions & useless Productions.

Step2: (Eliminate) Replace the terminals on the RHS

$S \to bA \mid aB$ Can be replaced by

$S \rightarrow C_bA \mid C_aB$

$C_b \rightarrow b$

$C_a \rightarrow a$

Consider the Production $A \rightarrow bAA \mid as \mid a$ can be replaced by

$A \rightarrow C_bAA \mid C_aS \mid a$

$C_b \rightarrow b$

$C_a \rightarrow a$

$B \rightarrow aBB \mid bS \mid b$ can be replaced

$B \rightarrow C_aBB \mid C_bS \mid b$

$C_a \rightarrow a$

$C_b \rightarrow b$

Step 4: Restricting the no of variables at the RHS

Consider $A \rightarrow C_bAA$

$A \rightarrow C_bA'$

$A' \rightarrow AA$

Consider $B \rightarrow CaBB$

$B \rightarrow CaB'$

$B' \rightarrow BB$

Therefore resultant grammar is

$S \rightarrow C_bA \mid C_aB$

$A \rightarrow C_bA' \mid C_aS \mid a$

$A' \rightarrow AA$

$B \rightarrow C_aB \mid C_bS \mid b$

$B' \rightarrow BB$

$C_a \rightarrow a$

$C_b \rightarrow b$

//

② Convert given CFG to CNF

$S \to ASA \mid aB$
$A \to B \mid S$
$B \to b \mid \epsilon$

Step 1: Eliminate ϵ- Productions

$B \to \epsilon$
$A \to B \Rightarrow A \to \epsilon$ $\quad \therefore V_n = \{A, B\}$

Identify all possibilities

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

Eliminate Unit Productions

$S \to S$, $A \to B$, $A \to S$

$\therefore S \to ASA \mid AS \mid SA \mid aB \mid a$
$A \to b \mid ASA \mid AS \mid SA \mid aB \mid a$
$B \to b$

Step 2: replace the Productions in the form of

$A \to BC$
$A \to ASA \Rightarrow$

$S \to AX \mid AS \mid SA \mid aB \mid a$
$A \to AX \mid SA \mid AS \mid aB \mid a \mid b$
$B \to b$
$X \to SA$

$S \to aB$, $A \to aB$

$S \to YB \Rightarrow S \to Ax \mid SA \mid AS \mid YB \mid a$

$A \to b \mid Ax \mid SA \mid AS \mid YB \mid a$

$B \to b$

$X \to SA$

$Y \to a$ //

(3) $S \to ASB \mid \epsilon$

$A \to aAS \mid a$

$B \to SbS \mid A \mid bb$

(4) Convert CFG to CNF

$\{S \to \sim S \mid [S \supset S] \mid P \mid q\}$ $\sim, [,], \supset, P, q$ are terminals

Greibach normal form (GNF):

A Context free grammar without $\in$ Productions is said to be in GNF if all the Productions are of the form $A \to a\alpha$

where $A \in V$ (Variables)
$a \in T$ (Terminal)
$\alpha \in$ String of 0 or more Variables

i.e

$A \to b$
$A \to bC_1C_2 \cdots C_n$ where C is NT

Lemma 1: (Substitution)

Let $G = (V, T, P, S)$ be the given Content free gramm

Consider Productions

$A \to B\alpha$
$B \to \beta 1 | \beta 2 | \cdots \beta n$ then the equivalent grammar can be obtained by Substitution

i.e $A_1 \to B_1\alpha | B_2\alpha | \cdots | \beta n\alpha$

Lemma 2: (Elimination of left recursion)

Let $G = (V, T, P, S)$ be given Content free grammar

Consider the Productions

$A \to A\alpha_1 | A\alpha_2 | \cdots | A\alpha n$

$A \to B_1 | B_2 | \cdots | B_n$

let Z be a new variable then equivalent Productions can be written as

$A \to B_1 | B_2 | \ldots | B_n$
$A \to B_1 Z | B_2 Z | \ldots | B_n Z$ } A's Productions

$Z \to \alpha_1 | \alpha_2 | \ldots \alpha_n$
$Z \to \alpha_1 Z | \alpha_2 Z | \ldots | \alpha_n Z$ } Z's Productions

$A \to A\alpha | \beta$
$A \to \beta A'$
$A' \to \alpha A' | \epsilon$
Sub ϵ in A'

$A \to \beta A' | \beta$
$A' \to \alpha A' | \alpha$

Conversion from CFG to GNF:

step 1: Eliminate null & unit Productions

step 2: check whether the CFG is already in CNF or not if not Convert it to CNF.

step 3: change the names of Non-Terminals in to some A, in ascending order of i

step 4: Alter the rules so that Non-Terminals are in ascending order. such that, if a Production is of the form $A_i \to A_j \alpha$ then $i < j$ & should never be $i \geq j$

if a) $i < j$ leave the Production as it is

b) $i = j$ Apply lemma 2

c) $i > j$, apply lemma 1

step 5 :- Remove left recursion, by introducing a new variable.

Given $S \to AA \mid a$
$A \to SS \mid b$

step 1:- check the given grammar is in CNF or not
given grammar is in CNF

step 2:- change the names of NT

$S \to A_1$
$A \to A_2$
$\Rightarrow$ $A_1 \to A_2 A_2 \mid a$
$A_2 \to A_1 A_1 \mid b$

step 3:- Consider $i < j$ condition

$A_1 \to A_2 A_2 \mid a$ $\quad \because i < j = 1 < 2 \checkmark$
$i \quad j$

leave as it is

$A_2 \to A_1 A_1 \mid b$ $\quad i < j - 2 < 1$ apply lemma 1

$A_2 \to A_2 A_2 A_1 \mid a A_1 \mid b$ $\quad i < j \; 2 = 2$, apply lemma 2

Eliminate left recursion

$A \to A\alpha \mid \beta$

$A_2 \to A_2 A_2 A_1 \mid a A_1 \mid b$
$A \quad \alpha \quad \beta_1 \quad \beta_2$

$A_2 \to a A_1 A_2' \mid b A_2' \mid a A_1 \mid b$ — CNF

$A_2' \to A_2 A_1 A_2' \mid A_2 A_1$ - not in CNF

Sub A_2 in A_2'

$A \to A\alpha \mid \beta$
$A \to \beta A'$
$A' \to \alpha A' \mid \epsilon$
$\therefore A \to \beta A' \mid \beta$
$A' \to \alpha A' \mid \alpha$

$A_2' \rightarrow aA_1A_2'A_1A_2' \mid bA_2'A_1A_2' \mid aA_1A_1A_2' \mid bA_1A_2' \mid aA_1A_2'A_1A_2'$
$\mid bA_2'A_1 \mid aA_1A_1 \mid bA_1$ — GNF

$A_1 \rightarrow A_2A_2 \mid a$ → not in GNF

Sub A_2 in A_1

$A_1 \rightarrow aA_1A_2'A_2 \mid bA_2'A_2 \mid aA_1A_2 \mid bA_2 \mid a$ - GNF

Therefore GNF form of given grammar is

$A_1 \rightarrow aA_1A_2'A_2 \mid bA_2'A_2 \mid aA_1A_2 \mid bA_2 \mid a$

$A_2 \rightarrow aA_1A_2' \mid bA_2' \mid aA_1 \mid b$

$A_2' \rightarrow aA_1A_2'A_1A_2' \mid bA_2'A_1A_2' \mid aA_1A_1A_2' \mid bA_1A_2' \mid aA_1A_2'A_1$
$\mid bA_2'A_1 \mid aA_1A_1 \mid bA_1$

① Given

$S \to CA \mid BB$

$B \to b \mid SB$

$C \to b$

$A \to a$

step1: NO $\in$ or unit productions & also the given grammar is in CNF

step3: change the names of NT in ascending order

$\longrightarrow$ $S \to CA \mid BB$ replace $S \to A_1$, $C \to A_2$, $A \to A_3$, $B \to A_4$

$\therefore A_1 \to A_2A_3 \mid A_4A_4$

$A_4 \to b \mid A_1A_4$

$A_2 \to b$

$A_3 \to a$

step4: check the condition $i < j$ else replace the NT

Consider $A_1 \to A_2A_3$ $i=1$ & $j=2$, $i<j = 1<2$ ✓

consider $A_1 \to A_4A_4$ $i=1$ & $j=4$, $1<4$ ✓

$A_4 \to b$ ✓

$A_4 \to A_1A_4$ $4<1$ ×

replace A_1 with A_1 Productions

$A_4 \to b \mid A_2A_3 \mid A_4A_4A_4$

$A_4 \to A_2A_3A_4$ $4<2$ ×

replace A_2 with A_2 Productions

$A_4 \rightarrow b \mid bA_3A_4 \mid A_4A_4A_4$

$A_4 \rightarrow A_4A_4A_4$ $\quad 4 < 4$ left recursion
hence eliminate it

Steps: Remove left Recursion (apply lemma 2)
Introduce new variable to remove left recursion

$A_4 \rightarrow b \mid bA_3A_4 \mid A_4A_4A_4$

$A_4 \rightarrow \underbrace{A_4}_{A}\underbrace{A_4}_{\alpha}\underbrace{A_4}_{\alpha} \mid \underbrace{bA_3A_4}_{\beta_1} \mid \underbrace{b}_{\beta_2}$

$A \rightarrow A\alpha \mid \beta$

$A \rightarrow \beta A'$
$A' \rightarrow \alpha A' \mid \epsilon$

$\therefore$ $A_4 \rightarrow bA_3A_4A_4' \mid bA_4'$
$A_4' \rightarrow A_4A_4A_4' \mid \epsilon$ $\}$ rule

$A \rightarrow \beta A' \mid \beta$
$A' \rightarrow \alpha A' \mid \alpha$ $\}$ rule

$A_4 \rightarrow bA_3A_4A_4' \mid bA_4' \mid bA_3A_4 \mid b$ - GNF

$A_4' \rightarrow A_4A_4A_4' \mid A_4A_4$ - not in GNF

To bring A_4' in GNF sub A_4 in A_4'

$A_4' \rightarrow bA_3A_4A_4'A_4A_4' \mid bA_4A_4 \mid bA_3A_4A_4 \mid bA_4A_4' \mid$
$bA_3A_4A_4'A_4 \mid bA_4'A_4 \mid bA_3A_4A_4 \mid bA_4$ - GNF

As A_1 is not in GNF

$A_1 \rightarrow A_2A_3 \mid A_4A_4$ sub A_2 in A_1

$A_1 \rightarrow bA_3 \mid A_4A_4$ sub A_4 in A_1

$A_1 \rightarrow bA_3 \mid bA_3A_4A_4'A_4 \mid bA_4'A_4 \mid bA_3A_4A_4 \mid bA_4$

Therefore GNF form for given grammar is

$A_1 \to bA_3 \mid bA_3A_4A_4'A_4 \mid bA_4'A_4 \mid bA_3A_4A_4 \mid bA_4$

$A_4 \to bA_3A_4A_4' \mid bA_4' \mid bA_3A_4 \mid b$

$A_4' \to bA_3A_4A_4'A_4A_4' \mid bA_4'A_4A_4' \mid bA_3A_4A_4A_4' \mid bA_4A_4' \mid$
$bA_3A_4A_4'A_4 \mid bA_4'A_4 \mid bA_3A_4A_4 \mid bA_4$

$A_2 \to b$

$A_3 \to a$

→ Given ① $A_1 \to A_2A_3$

$A_2 \to A_3A_1 \mid b$

$A_3 \to A_1A_2 \mid a$

② $S \to ABA$

$A \to aA \mid \in$

$B \to bB \mid \in$

Pumping lemma for Context free languages:-

A Context free language (CFL) is a language generated by Context free grammar.

Pumping lemma is used to prove that certain languages are not context free. It is a negative test.

let 'L' be any CFL then choose a string 'z' such that length of 'z' i.e $|z| \geq n$

Using Pumping lemma we can divide 'z' into five parts as $z = uvwxy$ in such that it should satisfy following conditions

i) $|vwx| \leq n$

ii) $|vx| \geq 1$

iii) for all $i \geq 0$ $z = uv^i wx^i y$ is not belongs to 'L'

1) P.T the language $L = \{a^n b^n c^n / n \geq 0\}$ is not a CFL.

let us assume that $L = a^n b^n c^n$ is a CFL

strings that can be derived are

$L = \{\in, abc, aabbcc, aaabbbccc, \ldots\}$

let $z = abc$ $n = 1$

$|z| \geq n$ $3 \geq 1$ ✓

Divide the z into 5 parts

$z = \underset{u}{a} \underset{v}{\in} \underset{w}{b} \underset{x}{\in} \underset{y}{c}$

i) $|vwx| \leq n$ (consider powers of i/p)

$|0 + 1 + 0| \leq 1 \quad = 1 \leq 1$ ✓

ii) $|vx| \geq 1$

$|0+0| \geq 1$

$0 \geq 1$ ✗

iii) $z = uv^i wx^i y$

Consider $i = 0, 1, 2$

Case 1: let $i=0$ then

✗ $n=2$

$z = \underset{u}{aa}\underset{v}{b}\underset{w}{b}\underset{x}{cc}\underset{y}{}$

$i=0$ then $z = aab^0bc^0c$ (∵ b, c are null)

$= aabc \notin L$

let $i=2$ then $z = aab^2bc^2c$

$= aabbbccc$

$= a^2b^3c^3 \notin L$

∴ It proves that given lang is not content free

(2) PT $L = a^n$ is a Prime number is not content free.

$L = a^n$, n is a prime no - {2, 3, 5, 7, 11, 13 - - -}

$L = \{aa, aaa, aaaaa, - - - -\}$

let $z = \underset{u}{aa}\underset{v}{aa}\underset{x}{a}\underset{y}{}$

i) $|vwx| \leq n$

$|1+1+1| \leq 5$

$3 \leq 5$ ✓

ii) $|vx| \geq 1$

$|1+1| \geq 1$

$2 \geq 1$ ✓

iii) $z = uv^i wx^i y$

if $i=1$ $aa^1aa^1a = aaaaa \in L$

if $i=2$ $aa^2aa^2a = a^7 \in L$

if $i=3$ $aa^3aa^3a = a^9 \notin L$

∴ The language is not Content free.

(3) PT $L = \{www \mid w \in (a,b)^*\}$ is not CFL

let $w = a^n b^n$ then

$L = \underbrace{a^n b^n}_{u} \underbrace{a^n}_{v} \underbrace{b^n}_{w} \underbrace{a^n}_{x} \underbrace{b^n}_{y}$

i) $|vwx| \leq n$

$|n+n+n| \leq n$

$3n \leq n$ ✓

ii $|vx| \geq 1$

$|n+n| \geq 1$

$2n \geq 1$ ✓

iii) $z = uv^i wx^i y$

if $i=0$ $a^n b^n a^0 b^n a^0 b^n$

$L = a^n b^n b^n b^n \notin L$

∴ it is not a CFL

Applications of CFG:-

→ Language designing
→ Syntax analysis designing
→ Natural language processing
→ Syntax verification

Enumeration of Properties of Content free languages:-

Content free languages are closed under

→ Union
→ Concatenation
→ Kleene closure

CFL are not closed under

→ Intersection
→ Intersection with Regular language
→ Complement