UNIT-III

Formal Grammar:

* Introduction
* classification of formal Grammar
  1. chomsky Hierarchy.
  2. Types.

* Introduction:—

Mathematically A formal Grammar is a tuple five

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

V = finite and nonempty set of non-terminal symbols (or) variables.

variables are represented by upper case letters

T = finite and non empty set of Terminal symbols represented by lower case letters and some special symbols are there.

P = It is a set of production rules are of the form

$P \rightarrow \alpha \rightarrow \beta$

$\alpha \in V$

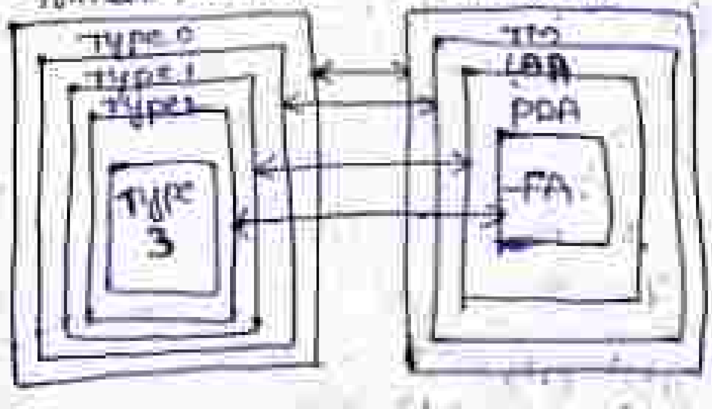
$\beta \in (V \cup T)^*$

S → It is the starting symbol of the Grammar is always, a variable which is $S \in V$.

Note:— Grammar's are used to describe a language

* classification of Grammar:—

- Using chomsky hierarchy.



Type 3 Grammar:—

* It is also called as Regular grammar.
* Type 3 Grammar is defined as $G = (V, T, P, S)$ where,

  $V \rightarrow$ set of variables

  $T \rightarrow$ set of Terminals

  $P \rightarrow$ set of production rules are of the form

  $\boxed{\begin{array}{l} A \rightarrow Ba \\ A \rightarrow a \end{array}}$ — According to left linear grammar

  (or)

  $\boxed{\begin{array}{l} A \rightarrow aB \\ A \rightarrow a \end{array}}$ — According to right linear grammar

Ex:— $A \rightarrow aB$

$A \rightarrow Ba$

$A \rightarrow a$

$A \rightarrow \epsilon$

where

$(A, B) \in V$

$a \in T^*$

* Type 3 Grammar is used to generating Regular language
* Regular languages are recognised (or) accepted by finite automata i.e., NFA (or) DFA

Type 2 Grammar:—

* It is also called as Context-free grammar
* A context-free grammar is defined as $G = (V, T, P, S)$

  where $V \rightarrow$ finite set of variables

  $T \rightarrow$ finite set of terminals

  $P \rightarrow$ finite set of production rules are of the form

  $\alpha \rightarrow \beta$

  where $\alpha \in V$

  $\beta \in (V \cup T)^*$

Ex:— $S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow ab$

$S \rightarrow \epsilon$

* Context-free grammars are used to generate Context-free language.

* Context-free language recognised (or) Accepted by pushdown-Automata.

Types Grammar :—

* It is also called as context - Sensitive Grammar.
* A CSG is defined as $G=(V,T,P,S)$ where

V = finite set of variables

T = finite set of Terminals

P = set of production rules are of the form $\alpha \to \beta$

where $\alpha \in (V \cup T)^+$

$\beta \in (V \cup T)^*$

length of $|\alpha| \le$ length of $|\beta|$

Ex:- $S \to aBb$

$bB \to aa$

$B \to b$

* CSG is used to generating Context - Sensitive language
* CSL recognised (or) Accepted by Linear Bounded Automata

Type 0 Grammar :—

* It is also called also Recursive - Grammar (or) Recursive Enumerable grammar (or) phrase structured Grammar.
* mathematically Recursive grammar is defined as

$G=(V,T,P,S)$ where $V \to$ finite set of variables

$T \to$ finite set of Terminals

$P \to$ set of production rules are of the form.

$\alpha \to \beta$

$\alpha \in (V \cup T)^{+}$

$\beta \in (V \cup T)^*$

$|\alpha| \gtrless |\beta|$

Ex:- $S \to aAbB$

$aAbB \to aB$

$abB \to aa$

$A \to \epsilon$

* Recursive Grammars are used to generating recursive language (or) Recursive enumerable language (or) phrase structured language.
* Recursive languages are recognized are accepted by turing machine.

Relationship b/w formal grammar and automata :-

1. Type 3 ⊆ Type 2 ⊆ Type 1 ⊆ Type 0
2. FA ⊆ PDA ⊆ LBA ⊆ TM

Context - Free Grammar:

* Introduction
* Design of CFL
* closure properties of CFL

Introduction :-

Context - free Grammar is a Grammar which is defined by four tuples like $G = (V, T, P, S)$ where

V - It is finite and non-empty set of non-Terminal symbols (or) variables.

T - finite and non-Empty set of Terminal symbols

P - finite and non-Empty set of production rules are of the form $\alpha \to \beta$

$$\alpha \in V$$
$$\beta \in (V \cup T)^*$$

Ex:- $S \to aSa$
$S \to bSb$
$S \to a|b|\epsilon$
$S \to \epsilon$

S → It is starting symbol.

Context free language :-

Let $G = (V, T, P, S)$ be a Context - free grammar. The CFG generating a language 'L' is called Context - free language

*It is denoted by $L(G)$.
*context-free languages are organized by PDA.

Design of CFL :—

1) Construct a CFL for the following set $\{\epsilon, a, aa, aaa, \ldots\}$

Sol:- Given set $\{\epsilon, a, aa, aaa, aaaa, \ldots a^n\}$

minimum string = ϵ
Next minimum string = a
maximum string = a^n

$S \rightarrow a^n$
$\downarrow$
$a \cdot a^{n-1} \Rightarrow S \rightarrow aS$
$\downarrow$
$a \cdot a \cdot a^{n-2}$ $\quad S \rightarrow \epsilon$
$\downarrow$ $\quad S \rightarrow a$
$a \cdot a \cdot a \cdot a^{n-3}$

CFG:- $\quad S$
$S \rightarrow aS$
$S \rightarrow \epsilon$
$S \rightarrow a$

$L = \{a^n \mid n \geq 0\}$

2) Construct a CFL for the following set $\{\epsilon, ab, aabb, \ldots\}$

Sol:- minimum string = ϵ
Next minimum string = ab
maximum string = $a^n b^n$

$S \rightarrow a^n b^n$
$S \rightarrow a\, a^{n-1} \cdot b^{n-1}\, b$
$S \rightarrow aa\, a^{n-2}\, b^{n-2}\, bb$

$\therefore S \rightarrow aSb$
$S \rightarrow \epsilon$
$S \rightarrow ab$

$\therefore$ CFG :- $S \rightarrow aSb$
$S \rightarrow \epsilon$
$S \rightarrow ab$

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

3) Construct a CFL for the following set $\{a, b, ab, aabb, aaabbb, \ldots\}$

Sol:- Minimum string $= a/b$

Maximum string $= a^n b^n$

$S \to a^n b^n$

$\to a a^{n-1} b^{n-1} b \Rightarrow S \to aSb$

$\to a a a^{n-2} b^{n-2} bb \quad S \to a$

$S \to b$

$\therefore$ CFG $S \to aSb$

$S \to a$

$S \to b$

$\therefore L = \{a^n b^n \mid n \geq 1\}$

4) Construct a CFG to generate the language $L = \{a^n b^{2n} \mid n > 0\}$

Sol:- Minimum string $= abb$

Maximum string $= a^n b^{2n}$

$S \to a^n b^{2n}$

$S \to a a^{n-1} b^{2n-2} bb \Rightarrow S \to aSbb$

$S \to abb$

$\therefore$ CFG $= S \to aSbb$

$S \to abb$

5) Construct CFG for the following CFL

$L = \{0^i 1^{i+1} \mid i \geq 0\}$

Sol: $L = \{0^i 1^{i+1} \mid i \geq 0\}$

$A \to 0^i 1^i$

$= 0^i 1^i 1$

$\to 0 0^{i-1} 1^{i-1} 1$

$S \to A1$

$\to 0A1$

CFG: $S \to A1$ $\quad A \to 0A1$

$A \to 0A1$ $\quad A \to \epsilon$

$A \to \epsilon$ $\quad A \to 01$

$A \to 01$

6) Construct a CFL from the following language

$L = \{a^m b^m c^n \mid m, n \geq 0\}$

Sol:

$\underbrace{a^m b^m}_{A} \underbrace{c^n}_{B}$

$A \to a^m b^m$

$\to a\, a^{m-1} b^{m-1} b$

$A \to aAb$

$A \to \epsilon$

$A \to ab$

$B \to c^n$

$B \to c\, c^{n-1}$

$B \to cB$

$B \to \epsilon$

$B \to c$

CFG :-
$S \to AB$
$A \to aAb$
$A \to \epsilon$
$A \to ab$
$B \to cB$
$B \to \epsilon$
$B \to c$

Closure properties of CFL :-

- context free languages are closed under union
- " " " " " " concatenation
- " " " " " " kleene closure
- " " " " " " Reversal

- context free languages are not closed under Complement
- " " " " " " " Intersection
- " " " " " " " difference

1st Derivation :-

* Introduction * Types of Derivation * Derivation tree

Derivation is a process of generating a string from a given grammar

Derivation process can be represented graphically is called Derivation Tree (or)

* left most derivation * Right most derivation

Left most derivation :- with example

In this, we can replace a left most variable to obtain the given input string.

Right most derivation :-

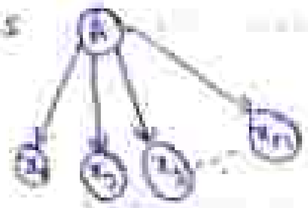
In this, we can replace a Right most variable to obtain the given input string.

Derivation tree :-

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

1. The root node of the tree is labelled with start symbol of G
2. All leaf nodes of tree are labelled by terminals or special symbols of G.
3. The interior nodes are labelled by variables of G
4. If any production rule in G is are the form
$A \rightarrow x_1 x_2 x_3 \ldots x_n$ then the

derivation tree is



- Find the i) left most derivation
ii) Right most derivation
iii) parse tree for the i/p string id+id*id
from the following grammar $E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow id$

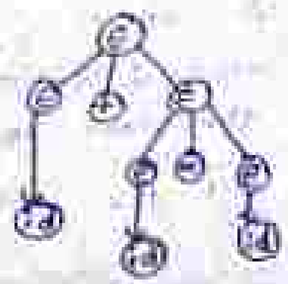
Sol:- the given grammar is $E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow id$

Input string id+id*id.

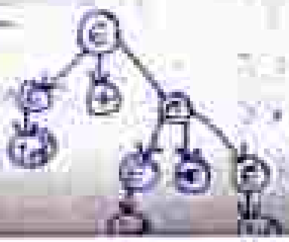
LMD:- $E \rightarrow E+E$
$\rightarrow E+E*E$
$\rightarrow id+E$
$\rightarrow id+E*E$
$\rightarrow id+id*E$
$\rightarrow id+id*id$.

RMD:- $E \rightarrow E+E$
$\rightarrow E+E*E$
$\rightarrow E+E*id$
$\rightarrow E+id*id$
$\rightarrow id+id*id$

Parse tree:-



Parse tree:-



Ambiguous Grammar:-

- A CFG $G=(V,T,P,S)$ which generates two (or) more parse trees for given i/p string is called Ambiguous grammar.
- that means an Ambiguous grammar has two or more left most derivations (or) right most derivation (or) parse tree.

Eg: Prove that $S \to aSbS$ is ambiguous for the i/p
$S \to bSaS$
$S \to \epsilon$

string abab

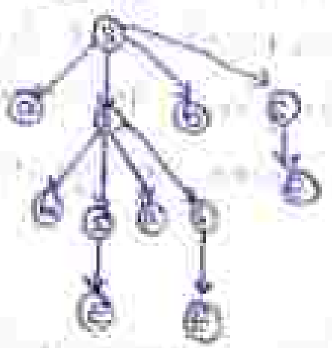
sol:- the given context free grammar is $S \to aSbS$
$S \to bSaS$
$S \to \epsilon$

the input string is $w = abab$

(i) LMD:

$S \to aSbS$
$\to abSaSbS$
$\to ab\epsilon aSbS$
$\to abaSbS$
$\to aba\epsilon bS$
$\to ababS$
$\to abab\epsilon$
$\to abab$

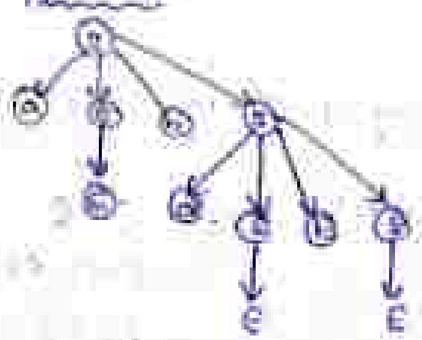
parse tree



(ii) LMD:

$S \to aSbS$
$\to a\epsilon bS$
$\to abS$
$\to abaSbS$
$\to aba\epsilon bS$
$\to ababS$
$\to abab\epsilon$
$\to abab$

parse tree



∴ the above grammar generates two parse trees (or) two left most derivation for the same i/p string $w = abab$. Hence the above grammar is ambiguous grammar.

2) P.T the grammar $E \to E+E$
$E \to E*E$ is ambiguous for i/p
$E \to id$

string $id+id*id$

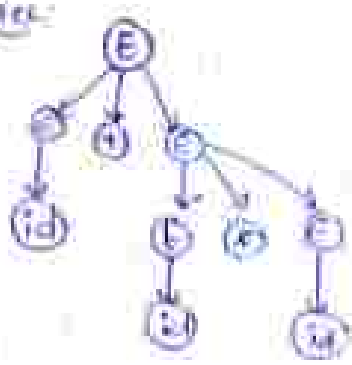
Ex:- the given context free grammar is $E \to E+E$
$E \to E*E$
$E \to id$

the input string is $w = id+id*id$

① LMD:

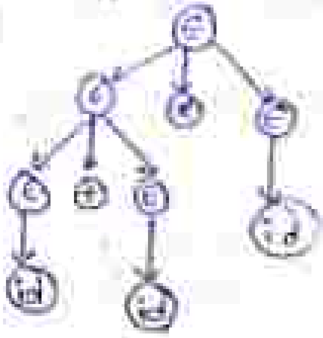
$E \to E+E$
$\to id+E$
$\to id+E*E$
$\to id+id*E$
$\to id+id*id$.

parse tree:



② RMD:

$E \to E*E$
$\to E+E*E$
$\to id+E*E$
$\to id+id*E$
$\to id+id*id$.



Simplification of CFG:

* Introduction
* methods
    1. Elimination of useless symbols.
    2. Elimination of ε-productions
    3. Elimination of unit productions.

Introduction:-

Its means minimizing the no. of productions in the given CFG, that is reducing size of CFG. size of CFG is equal to no. of productions.

Methods:- $S \to AB$
$A \to a$
$A \to aA$
$B \to SB$

Elimination of useless symbols:-

useful symbol:- A variable is said to be useful if and only if

* It generates a terminal string
* It is used in derivation of a string at least one time

Useless symbol:-

* A variable is said to be useless if and only if.
  * It doesn't generate a terminal string
  * It doesn't used in derivation of a string at least one time.

Procedure:-

step 1:- Determine useless symbols in the grammar.

step 2:- Remove the productions which contains useless symbols in the grammar.

Ex:- eliminate useless symbols from the following grammar

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

sol:- The given CFG is

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

In the given grammar 'B' doesn't generating a terminal string.

$S \rightarrow \underline{A}B$
$S \rightarrow a\underline{B}$
$\rightarrow a\underline{B}C$
$\rightarrow a\underline{A}BC$
$\rightarrow aaB\underline{C}$
$\rightarrow aaBb$
$\rightarrow aa\underline{A}Bb$
$\rightarrow aaaBb$
⋮

∴ 'B' is useless symbol.

so we can eliminate the productions which contains 'B'

∴ The reduced CFG is

$S \rightarrow CA$ | $C \rightarrow b$.
$A \rightarrow a$

2) Elimination of ε-production:-

ε-production:- A production is of the form $A \to \epsilon$ is called ε-production (or) NULL production.

procedure:-

step 1:- If the grammar contains $A \to \epsilon$ then replace 'A' with ε in the remaining productions.

step 2:- Remove $A \to \epsilon$ from the grammar.

Ex:- Remove ε-productions from the following grammar

$A \to 0B1 \mid 1B1$

$B \to 0B \mid 1B \mid \epsilon$

sol:- the given CFG is $A \to 0B1$

$A \to 1B1$

$B \to 0B$

$B \to 1B$

$B \to \epsilon$

$A \to 0B1$
$\to 0\epsilon 1$
$\to 01$

$\therefore A \to 0B1$
$A \to 01$

$B \to 0B$
$\to 0\epsilon$
$\to 0$

$\therefore B \to 0B$
$B \to 0$

$A \to 1B1$
$\to 1\epsilon 1$
$\to 11$

$\therefore A \to 1B1$
$A \to 11$

$B \to 1B$
$\to 1\epsilon$
$\to 1$

$\therefore B \to 1B$
$B \to 1$

After eliminating $B \to \epsilon$ the resultant CFG is

$A \to 0B1$ $\quad B \to 1B$

$A \to 01$ $\quad B \to 1$

$A \to 1B1$

$A \to 11$

$B \to 0B$

$B \to 0$

14/11
* Normal-forms :-

* Introduction
* Types of Normal-forms
    1. chomsky Normal-form (CNF)
    2. Greiback Normal-form (GNF)

Introduction :-
In CFG each production of the form $\alpha \longrightarrow \beta$ where $\alpha \in V$ that means β contains any no.of non-terminal symbols and any no.of terminal symbols. But, we need to have a grammar in specific form i.e; we can decide the no.of non-terminals and terminals on R.H.S of the grammar. This can be implemented by using "normalization of CFG"

Normalization :-
The process of Arranging the grammar with fixed no.of

non-terminals and terminals in rule of CFG is called normalization

normal forms are classified into two types

i) chomsky normal form.

ii) Greiback normal form

chomsky normal form :-

It is defined as $\alpha \to \beta$

non-terminal $\to$ non-terminal non-terminal.   (or)   Non-terminal $\to$ Terminal.

conversion of CFG to CNF :-

procedure :-

step 1 :- simplify the CFG

step 2 :- convert the simplified CFG to CNF.

Eg:- convert the following CFG into chomsky normal form.

$S \to aaaaS$

$S \to aaaa$

Sol:- The given grammar is $S \to aaaaS$

$S \to aaaa$

Consider a non-terminal $A \to$ that derives Terminal a

$\therefore$ the production rule is $A \to a$ is in CNF

$S \to aaaaS$

$S \to A\boxed{AAAS}$ can be replaced by P_1

$S \to AP_1$ is in CNF.

$P_1 \to A\boxed{AAS}$ can be replaced by P_2

$P_1 \to AP_2$ is in CNF

$P_2 \to A\boxed{AS}$ can be replaced by P_3

$P_2 \to AP_3$ is in CNF

$P_3 \to AS$ is in CNF

$S \to aaaa$

$S \to A\boxed{AAA}$ can be replaced by P_4

$S \rightarrow AP_4$ is in CNF

$P_4 \rightarrow A\boxed{AA}$ can be replaced by P_5

$P_4 \rightarrow AP_5$ is in CNF

$P_5 \rightarrow AA$ is in CNF

The resultant grammar CNF is

$S \rightarrow AP_1$ $\quad P_5 \rightarrow AA$

$S \rightarrow AP_4$ $\quad A \rightarrow a$

$P_1 \rightarrow AP_2$

$P_2 \rightarrow AP_3$

$P_3 \rightarrow AS$

$P_4 \rightarrow AP_5$

5) Convert the given CFG to CNF $\quad S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow a$

$S \rightarrow b$

Sol: The given grammar is $S \rightarrow aSa$

$S \rightarrow bSb$.

$S \rightarrow a$

$S \rightarrow b$

It is already in simplified form

Consider a non-terminal A that derives a terminal a and the non-terminal B that derives the terminal b.

∴ the production rules is $A \rightarrow a$, $B \rightarrow b$ are in CNF

(i) $S \rightarrow aSa$

$S \rightarrow A\boxed{SA}$ can be replaced by P_1

$S \rightarrow AP_1$ is in CNF.

$P_1 \rightarrow SA$ is in CNF

(ii) $S \rightarrow bSb$

$S \rightarrow B\boxed{SB}$ can be replaced by P_2

$S \rightarrow BP_2$ is in CNF

$P_2 \rightarrow SB$ is in CNF

(iii) $S \rightarrow a$ is in CNF

$S \rightarrow b$ is in CNF.

∴ the resultant grammar in CNF is

$S \rightarrow AP_1$

$S \rightarrow BP_2$

$C \rightarrow a$

$S \rightarrow b$

$P_1 \rightarrow SA$

$P_2 \rightarrow SB$

$A \rightarrow a$

$B \rightarrow b$.

Greibach normal form (GNF):-

GNF is defined as

Nonterminal $\rightarrow$ Terminal . any no. of nonterminals

Non-terminal $\rightarrow$ Terminal.

Lemma 1:

Let CFG be $G = (V, T, P, S)$ and there is a production rule $A \rightarrow aB$ and $B \rightarrow \beta_1 | \beta_2 | \beta_3 | \cdots | \beta_n$ then add the new production rule $A \rightarrow a\beta_1 | a\beta_2 | a\beta_3 | \cdots | a\beta_n$ to GNF

$\therefore$ B is replaced by $B \rightarrow \beta_1 | \beta_2 | \cdots | \beta_n$

Lemma 2:

Let CFG be $G = (V, T, P, S)$ and there is production rule

$A \rightarrow A\alpha_1 | A\alpha_2 | \cdots | A\alpha_n | \beta_1 | \beta_2 | \cdots | \beta_n$ then the production rules are added to GNF.

$A \rightarrow \beta_1 | \beta_2 | \beta_3 | \cdots | \beta_n$

$A \rightarrow \beta_1 Z | \beta_2 Z | \beta_3 Z | \cdots | \beta_n Z$

$Z \rightarrow \alpha_1 | \alpha_2 | \alpha_3 | \cdots | \alpha_n$

$Z \rightarrow \alpha_1 Z | \alpha_2 Z | \alpha_3 Z | \cdots | \alpha_n Z$

Converting ~~CRG~~ CFG into GNF:-

Procedure:-

step 1:- Simplify the CFG.

step 2:- Converting simplified CFG into GNF

Ex:- Convert the given CFG to GNF $S \rightarrow ABA$

$A \rightarrow aA | \epsilon$

$B \rightarrow bB | \epsilon$

Sol:- The Given CFG is $S \rightarrow ABA$

$A \rightarrow aA$

$A \to \epsilon$

$B \to bB$

$B \to \epsilon$

Simplified of given CFG :-

③ elimination of ε-productions :-

$A \to \epsilon \qquad B \to \epsilon$

① $S \to \underline{A}BA$
$S \to \epsilon BA$
$S \to BA$

② $S \to AB\underline{A}$
$S \to AB\epsilon$
$S \to AB$

③ $S \to A\underline{B}A$
$S \to A\epsilon A$
$S \to AA$

④ $S \to \underline{AB}A$
$S \to \epsilon\epsilon A$
$S \to A$

⑤ $S \to \underline{ABA}$
$S \to \epsilon B\epsilon$
$S \to B$

$A \to a\underline{A} \qquad B \to bB$
$A \to a\epsilon \qquad B \to b\epsilon$
$A \to a \qquad B \to b$

∴ After eliminating $A \to \epsilon$, $B \to \epsilon$ from the grammar the resultant grammar is

$S \to ABA \qquad A \to aA$
$S \to BA \qquad A \to a$
$S \to AB \qquad B \to bB$
$S \to AA \qquad B \to b$
$S \to A$ ✓
$S \to B$ ✓

Elimination of unit productions :-

The above grammar has two unit productions like

$S \to A$ × $\qquad S \to B$ ×
$S \to aA \qquad S \to bB \qquad$ [∵ since $A \to aA$, $A \to a$; $B \to bB$, $B \to b$]
$S \to a \qquad S \to b$

∴ After elimination unit productions $S \to A$, $S \to B$ from the grammar the resultant grammar is

$S \to ABA \qquad A \to aA$
$S \to BA \qquad A \to a$
$S \to AB \qquad B \to bB$
$S \to AA \qquad B \to b$
$S \to aA$
$S \to a$
$S \to bB$
$S \to b$

there is no useless production.

the simplified CFG is

$S \to ABA$ $\quad A \to aA$
$S \to BA$ $\quad A \to a$
$S \to AB$ $\quad B \to bB$
$S \to AA$ $\quad B \to b$
$S \to aA$
$S \to a$
$S \to bB$
$S \to b$

Converting simplified CFG to GNF:

i) $S \to ABA$
$S \to aABA$ ✓
$S \to aBA$ ✓

$A \to aA$ ✓
$A \to a$ ✓
$B \to bB$ ✓
$B \to b$ ✓

ii) $S \to BA$
$S \to bBA$ ✓
$S \to bA$ ✓

iii) $S \to AB$
$S \to aAB$
$S \to aB$ ✓

iv) $S \to AA$
$S \to aAA$
$S \to aA$

v) $S \to aA$ ✓
$S \to a$ ✓

vi) $S \to bB$ ✓
$S \to b$ ✓

∴ The resultant grammar is in GNF is

$S \to aABA \mid aBA \mid bBA \mid bA \mid aAB \mid aB \mid aAA \mid aA \mid bB \mid a \mid b$
$A \to aA \mid a$
$B \to bB \mid b$

③ Convert the following CFG into GNF $S \to AA \mid 0$
$A \to SS \mid 1$

Sol: Given Grammar $S \to AA$
$S \to 0$
$A \to SS$
$A \to 1$

The simplified CFG is $S \to AA$
$S \to 0$
$A \to SS$
$A \to 1$

① $S \to AA \mid 0$ ② $S \to AA \mid 0$ ③ $A \to S Z$

$S \to 0ZA \mid 0$ $S \to 1A \mid 0$ $A \to 0 Z$

$S \to 0$ $S \to 1A$ $A \to 0 Z Z$

$S \to 0Z$ $S \to 0$ $A \to 1 A Z$

$Z \to 1A$ $A \to 0 Z$

$Z \to 1AZ$

$Z \to 0Z A$ $Z \to 1AZ$

$Z \to 0A$ $Z \to 0AZ$

$Z \to 0ZA$ $Z \to 0ZAZ$

$Z \to 1AA$ $Z \to 1AAZ$

$Z \to 0A$ $Z \to 0AZ$

∴ The resultant grammar is:

$S \to 0 \mid 0Z \mid 1A$

$Z \to 0A \mid 0ZA \mid 1AA \mid 0A \mid 0AZ \mid 0ZAZ \mid 1AAZ \mid$

$A \to 0S \mid 0ZS \mid 1AS \mid 1$

③ Convert the given CFG to GNF $S \to CA$
$A \to a$
$C \to aB \mid b$

A: Given CFG is not a simplified grammar. After eliminating the useless symbols the resultant CFG is: $S \to CA$
$A \to a$
$C \to b$

By applying lemma 1 $S \to CA$
$S \to bA$

∴ The resultant GNF is $S \to bA$
$A \to a$
$C \to b$

④ Convert the given CFG to GNF $S \to S1$
$S \to 0S1 \mid 01$

The given CFG is a simplified CFG.
The resultant grammar is: $S \to S1$
$S \to 0S1$
$S \to 01$

Replaced 0 by A, 1 by B
then productions are $A \to 0$
$B \to 1$

$S \to SS$

$S \to ASB$

$S \to AB$

Applying lemma ①

① $S \to SS$
$S \to ASBS$
$S \to 0SBS$

② $S \to SS$
$S \to ABS$
$S \to 0BS$

③ $S \to ASB$
$S \to 0SB$

$S \to AB$
$S \to 0B$

The resultant grammar is

$S \to 0SBS \mid 0BS \mid 0SB \mid 0B$

$A \to 0$

$B \to 1$

Pumping lemma for CFL:—

pumping lemma is used for proving the given language is CFL (or) not

Lemma :- let 'L' be any CFL, then there is a constant n which depends only on L such that there exist a string $w = uvxyz$ such that
1. $|vy| \geq 1$
2. $|vxy| \leq n$
3. for $\forall\, i \geq 0$ uv^ixy^iz is in L

Then L is said to be CFL otherwise it is not a CFL

① prove that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not a CFL

the given language $L = \{a^n b^n c^n \mid n \geq 0\}$

$L = \{\epsilon, abc, aabbcc, \ldots\}$

consider a constant n and the string $w = a^n b^n c^n$

consider a string $w \in L$

$w = abc$ for $n = 1$

$|w| = 3n$

for $i = 1$ $w = abc$

for $i = 2$

$w = uv^ixy^iz$

$w = uv^2xy^2z$

$w = a^{n-1} a^2 b^{n-1} b^2 c^n$

$w = a^{n+1} b^{n+1} c^n \notin L$

$a^n \quad b^n \quad c^n$

$a^{n-1} \quad a \quad b^{n-1} \quad b \quad c^n$

$u \quad v \quad x \quad y \quad z$

∴ The given language is not a CFL

(5) show that the language $L = \{ SS^T / S \in \{a,b\}^* \}$

Given language $L = \{ SS^T / S \in \{a,b\}^* \}$

$L = \{ \epsilon,$