UNIT-III

Formal Grammar:

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

* Introduction :—

Mathematically A formal Grammar is a tuple like

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

V = finite and non Empty 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

$P \rightarrow \alpha \rightarrow \beta$

$\alpha \in V$

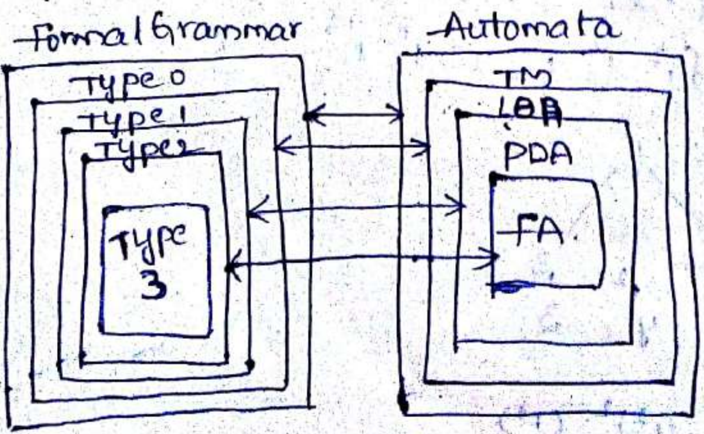
$\beta \in (V \cup T)^*$

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 → set of variables.
  T → set of Terminals
  P → set of production rules are of the form

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

(or)

$\boxed{\begin{matrix} A\to aB \\ A\to a \end{matrix}}$ — According to Right linear grammar.

Ex:— $A\to aB$
$A\to Ba$
$A\to a$
$A\to \epsilon$

where.
$(A,B)\in V$
$a\in T^*$

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

Type 2 Grammar:—

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

  where V → finite set of variables
  T → finite set of Terminals
  P → finite set of production rules are of the form

$$\alpha\to\beta$$

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

Ex:— $S\to aSa$
$S\to bSb$
$S\to ab$
$S\to\epsilon$

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

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

Type 1 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 $\alpha \rightarrow \beta$

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

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

  length of $|\alpha| \leq$ length of $|\beta|$

Ex:- $S \rightarrow aBb$

$bB \rightarrow aa$

$B \rightarrow b$

* 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 → finite set of variables

  T → finite set of Terminals

  P → set of production rules are of the form.

  $\alpha \rightarrow \beta$

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

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

  $|\alpha| \geq |\beta|$ &

Ex:- $S \rightarrow aAbB$

$aAbB \rightarrow aB$

$abB \rightarrow aA$

$A \rightarrow \epsilon$

* Recursive Grammars are used to generating recursive language (or) Recursive-Enumerable language (or) phrase structured language.

* Recursive languages are recognised are accepted by Turing machine.

Relationship b/w formal grammar and automata :—

1. Type 3 $\subseteq$ Type 2 $\subseteq$ Type 1 $\subseteq$ Type 0
2. FA $\subseteq$ PDA $\subseteq$ LBA $\subseteq$ TM

Context - Free Grammar :

* Introduction
* Design of CFL
* closure properties of CFL.

*Introduction :—

Content-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 $\alpha \rightarrow \beta$

$\alpha \in V$

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

Ex:- $S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow aa|bb$

$S \rightarrow \epsilon$

S → It is starting symbol.

Content free -language :—

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

*It is denoted by L(G).

*Context free languages are recognized by PDA.

Design of CFL :—

1) construct a CFL for the following set. $\{\epsilon, a, aa, aaa, \dots a^n\}$

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

minimum string $= \epsilon$

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:-

$S \rightarrow aS$

$S \rightarrow \epsilon$

$S \rightarrow a$

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

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

sol:- Minimum string $= \epsilon$

Next minimum string $= ab$

Maximum string $= a^n b^n$

$S \rightarrow a^n b^n$

$S \rightarrow a \underline{a^{n-1} \cdot b^{n-1}} b$

$S \rightarrow a a \underline{a^{n-2} b^{n-2}} bb$

$\therefore S \rightarrow aSb$

$S \rightarrow \epsilon$

$S \rightarrow ab$

$\therefore$ 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, ---}

sol:- Minimum string = a/b

Maximum string $= a^n b^n$

$S \rightarrow a^n b^n$

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

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

$S \rightarrow b.$

$\therefore$ CFG $S \rightarrow aSb$

$S \rightarrow a$

$S \rightarrow b$

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

4) construct a CFG to generate the language $L = \{a^n b^{2n} / n \geq 1\}$

sol:- Minimum string = abb

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

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

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

$S \rightarrow abb$

$\therefore$ CFG $= S \rightarrow aSbb$

$S \rightarrow abb.$

5) Construct CFG for the following CFL

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

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

$= 0^i 1^i . 1$

$A \rightarrow 0^i 1^i$

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

$S \rightarrow A1$

$\rightarrow 0A1$

$A \rightarrow 0A1$

$A \rightarrow \epsilon$

$A \rightarrow 01$

CFG: $S \rightarrow A1$

$A \rightarrow 0A1$

$A \rightarrow \epsilon$

$A \rightarrow 01$

6) construct a CFL from the following Language

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

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

$A \rightarrow a^m b^m$

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

$A \rightarrow aAb$

$A \rightarrow \epsilon$

$A \rightarrow ab$

$B \rightarrow c^n$

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

$B \rightarrow cB$

$B \rightarrow \epsilon$

$B \rightarrow c$

CFG :- $S \rightarrow AB$

$A \rightarrow aAb$

$A \rightarrow \epsilon$

$A \rightarrow ab$

$B \rightarrow cB$

$B \rightarrow \epsilon$

$B \rightarrow c$

Closure properties of CFL :-

- content free languages are closed under union
- " " " " " " concatenation.
- " " " " " " Kleene closure
- " " " " " " Reversal
- content free languages are not closed under Complement
- " " " " " " " Intersection
- " " " " " " " difference.

*** 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 * Rightmost 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, at each step we can replace a Right most variable to obtain the given input string.

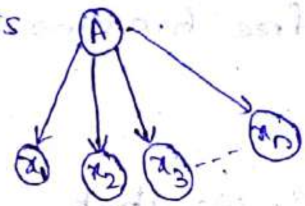
Derivation tree :—

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

* The root node of the Tree is labelled with start symbol of G
* All leaf nodes of Tree are labelled by Terminals (or) special symbols of G.
* The interior nodes are labelled by variables of G.
* If any production rule in G. is are the form.

$A \rightarrow x_1 x_2 x_3 --- x_n$ 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 $E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow id$.

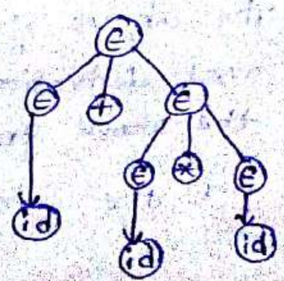
Sol:- the given grammar is $E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow id$.

Input string id+id*id.

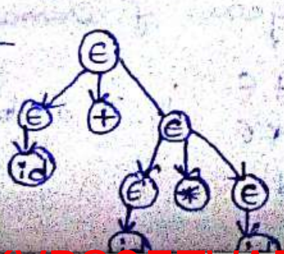
LMD:-
$E \rightarrow E+E$
$\rightarrow \cancel{E+E*E}$
$\rightarrow id+E$
$\rightarrow id+E*E$
$\rightarrow id+id*E$
$\rightarrow id+id*id$.

RMD:-
$E \rightarrow E+E$
$\rightarrow \cancel{id}\ E+E*E$
$\rightarrow E+E*id$
$\rightarrow E+id*id$
$\rightarrow id+id*id$

Parse tree:—



Parse tree:—



Ambiguious Grammar:-

* A CFG G(V,T,P,S) which generates two (or) more parse trees for given i/p string is called Ambiguious grammar.

* That means an Ambegeous grammar has two or more left most derivations (or) right most derivation (or) parse tree.

Ex:- Prove that $S \to aSbS$ is ambegeous for the i/p
$S \to bSaS$
$S \to \epsilon$

string abab

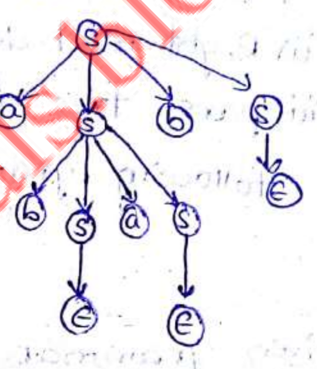
Sol:- The given Context free Grammar is $S \to aSbS$
$S \to bSaS$
$S \to \epsilon$

the input string is $\omega = abab$

① LMD:

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

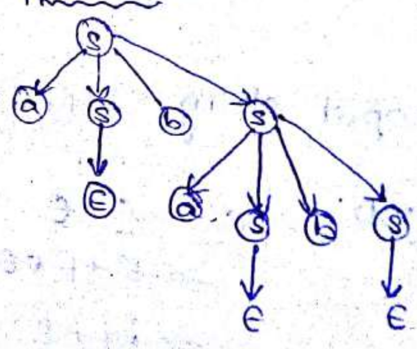
parsetree



② LMD:

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

Parsetree



∴ The above grammar generates two parsetrees (or) two left most derivation for the same i/p string $\omega = abab$. Hence the above grammar is ambegeous grammar.

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

string id+id*id.

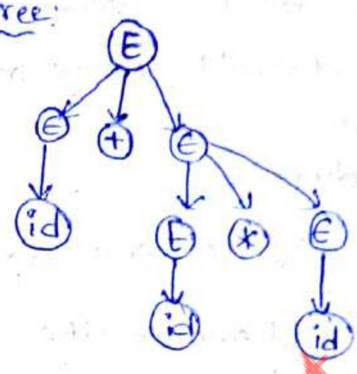
sol:- The given context free grammar is $E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow id$

The input string is $w = id+id*id$.

① LMD:

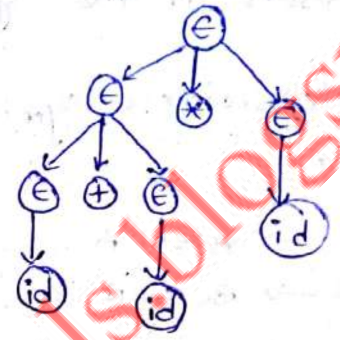
Parse tree:

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



② RMD:-

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



** (7m)
Simplification of CFG:

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

Introduction:-

It's 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:- $S \rightarrow AB$
$A \rightarrow a$
$A \rightarrow aA$
$B \rightarrow SB$

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 generates 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.

$S \rightarrow AB$
$S \rightarrow aB$
$\rightarrow aBc$
$\rightarrow aABc$
$\rightarrow aaBc$
$\rightarrow aaBb$
$\rightarrow aaABb$
$\rightarrow aaaBb$
$\vdots$

so, $\therefore$ 'B' is useless symbol.

so, we can eliminate the productions which contains 'B'.

$\therefore$ The reduced CFG is

$S \rightarrow cA$ | $c \rightarrow b$.
$A \rightarrow a$

2) elimination of ∈-production:-

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

procedure:-

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

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

Ex:- Remove ∈-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

$A \to 0B1$
$A \to 01$
$A \to 1B1$
$A \to 11$
$B \to 0B$
$B \to 0$
$B \to 1B$
$B \to 1$

14M
** 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 $\alpha \to \beta$ where $\alpha \in V$ that means β contains any no. of non-terminal symbols and any no. of terminal symbols. $\beta \in (V \cup T)$ 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 on R.H.S 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.

Ex:- 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 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

$S \to AP_4$ is in CNF

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

$P_4 \to AP_5$ is in CNF

$P_5 \to AA$ is in CNF.

The resultant Grammar CNF is

$S \to AP_1$ $\quad$ $P_5 \to AA$

$S \to AP_4$ $\quad$ $A \to a$

$P_1 \to AP_2$

$P_2 \to AP_3$

$P_3 \to AS$

$P_4 \to AP_5$

2) Convert the given CFG to CNF. $S \to aSa$

$S \to bSb$

$S \to a$

$S \to b$.

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

$S \to bSb$.

$S \to a$

$S \to b$

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.

$\therefore$ The production rules is $A \to a$ $\quad$ are in CNF.

$B \to b$.

(i) $S \to aSa$

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

$S \to AP_1$ is in CNF.

$P_1 \to SA$ is in CNF.

(ii) $S \to bSb$

$S \to B\boxed{SB}$ can be Replaced by P_2

$S \to BP_2$ is in CNF

$P_2 \to SB$ is in CNF

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

$S \to b$ is in CNF.

$\therefore$ The resultant grammar in CNF is

$S \to AP_1$

$S \to BP_2$

$S \to a$

$S \to b$

$P_1 \to A$

$P_2 \to SB$

$A \to a$

$B \to b.$

Greibach Normal Form (GNF) :-

GNF is defined as

Non-terminal $\to$ Terminal. any no. of nonterminals

Non-terminal $\to$ Terminal.

Lemma 1:

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

$\therefore$ B is replaced by $B \to \beta_1|\beta_2|\ldots|\beta_n$

Lemma 2:

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

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

$A \to \beta_1|\beta_2|\beta_3|\ldots|\beta_n$

$A \to \beta_1 Z|\beta_2 Z|\beta_3 Z|\ldots|\beta_n Z$

$Z \to \alpha_1|\alpha_2|\alpha_3|\ldots|\alpha_n$

$Z \to \alpha_1 Z|\alpha_2 Z|\alpha_3 Z|\ldots|\alpha_n Z$

Converting ~~CNF~~ CFG into GNF :-

Procedure:-

step 1 :- Simplify the CFG.

step 2 :- Converting simplified CFG into GNF.

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

$A \to aA|\epsilon$

$B \to bB|\epsilon$

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

$A \to aA$

$A \rightarrow \epsilon$

$B \rightarrow bB$

$B \rightarrow \epsilon$

Simplified of given CFG :-

ⓐ elimination of ε-productions :-

$A \rightarrow \epsilon \quad B \rightarrow \epsilon$

① $S \rightarrow ABA$
$S \rightarrow \epsilon BA$
$S \rightarrow BA$

② $S \rightarrow ABA$
$S \rightarrow AB\epsilon$
$S \rightarrow AB$

③ $S \rightarrow ABA$
$S \rightarrow A\epsilon A$
$S \rightarrow AA$

④ $S \rightarrow ABA$
$S \rightarrow \epsilon\epsilon A$
$S \rightarrow A$

⑤ $S \rightarrow ABA$
$S \rightarrow \epsilon B\epsilon$
$S \rightarrow B$

$A \rightarrow aA \qquad B \rightarrow bB$
$A \rightarrow a\epsilon \qquad B \rightarrow b\epsilon$
$A \rightarrow a \qquad B \rightarrow b$

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

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

Elimination of unit productions :-

The above grammar has two unit productions like

$S \rightarrow A \qquad S \rightarrow B$

$S \rightarrow aA \qquad S \rightarrow bB$ [∴ since $A \rightarrow aA$, $A \rightarrow a$, $B \rightarrow bB$, $B \rightarrow b$]
$S \rightarrow a \qquad S \rightarrow b$

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

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

there is no useless production.

∴ The simplified CFG is

S → ABA
S → BA
S → AB
S → AA
S → aA
S → a
S → bB
S → b

A → aA
A → a
B → bB
B → b

Converting simplified CFG to GNF:

i) S → ABA
S → aABA ✓
S → aBA ✓

A → aA ✓
A → a ✓
B → bB ✓
B → b ✓

ii) S → BA
S → bBA ✓
S → bA ✓

iii) S → AB
S → aAB
S → aB ✓

iv) S → AA
S → aAA
S → aA

v) S → aA ✓
S → a ✓

vi) S → bB ✓
S → b ✓

∴ The resultant grammar is in GNF is

S → aABA | aBA | bBA | bA | aAB | aB | aAA | aA | bB | a | b
A → aA | a
B → bB | b.

(2) Convert the following CFG into GNF S → AA | 0
A → SS | 1

Sol: Given Grammar S → AA
S → 0
A → SS
A → 1

The simplified CFG is S → AA
S → 0
A → SS
A → 1

① $S \to AA \mid 0$ ② $S \to AA \mid 0$ ③ $A \to S1$

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

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

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

$Z \to SA$ $A \to 0S$

$Z \to SAZ$

$Z \to SA$ $Z \to SAZ$

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

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

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

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

∴ 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 $S \to CA$
$A \to a$
$C \to aB \mid b$

⇒ 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 $S \to SS$
$S \to 0S1 \mid 01$

The given CFG is a simplified CFG

The resultant grammar is $S \to SS$
$S \to 0S1$
$S \to 01$

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 $S \to SS$
$S \to ABS$
$S \to 0BS$

② $S \to ASB$ $\quad$ $S \to AB$
$S \to 0SB$ $\quad$ $S \to 0B$

The resultant grammar GNF 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 a part '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 $L = \{a^n b^n c^n \mid n \geq 0\}$.

$L = \{\epsilon, abc, aabbcc, \text{----}\}$

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 $i = 2$

$w = uv^ixy^iz$

$w = uv^2xy^2z$

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

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

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

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

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

$\therefore$ The given language is not a CFL.

② show that the language $L = \{ ss^T \mid s \in \{a,b\}^* \}$

Given language $L = \{ ss^T \mid s \in \{a,b\}^* \}$

$L = \{ \epsilon,$