Imagination is more important than Knowledge.

Module : 3 Regular and Non-Regular ~~Grammer~~ Grammar

1. Grammar :-

A grammar is a formal device used to describe the syntax of a language.

A formal grammar has four important components

1 T : The set of terminals (Alphabet)
These are the finite set of symbols that actually forms the string of the language (usually small letters)

2 V : The set of variables/non-terminals / syntactic categories
Each variable represents a language
Each variable produces a set of strings. Example : The word "KEYWORD" can be considered as variable in the grammar of C language as it produces fixed 32 strings (if, else, while ...)
Similarly "VERBS" in english grammar

3 P : The set of Production Rules
Production rules represent the recursive definition of the language
Each Production consist of

<head> ⟶ <body>

Whatever is here is called head of production

compulsory symbol, known as production symbol

Whatever is here is called body of the production

Head can be any combination of terminals and non-terminals and so is the body

For example

$A \rightarrow a$

$A \rightarrow Aa$

$aA \rightarrow aaA$

$aAb \rightarrow b$

here A is non-terminal and a, b are terminal. All of the above are production

4. S : The start symbol

S is usually a non-terminal. It actually ~~defines~~ represents the language being defined.

formal

So, any grammar can be represented as a tuple of 4 elements

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

Example : The grammar for all valid mathematical formulaes will be like :-

MON TUE WED THU FRI SAT SUN

$V = \{$ Expression(E), Formulae(F), Number(N) $\}$

$T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (,), +, -, *, / \}$

$P = \{$

1. Formulae(F) $\longrightarrow$ Expression(E)
2. E $\longrightarrow$ (E)
3. E $\longrightarrow$ Number(N)
4. E $\longrightarrow$ E + E
5. E $\longrightarrow$ E − E
6. E $\longrightarrow$ E * E
7. E $\longrightarrow$ E / E
8. N $\longrightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

$\}$

Special representation in 8.
If $A \longrightarrow \alpha$ and $A \longrightarrow \beta$
we can write $A \longrightarrow \alpha \mid \beta$

$S =$ Formulae(F)

The grammar $G_1 = (V, T, P, S)$ here represents / verifies every correct mathematical formula

The Language of this grammar is like :-
$L(G_1) = \{ 2+3,\ 1,\ (1+2)*3/9,\ 1*5*4*3+2,\ (5+9)/(3+4)\ \ldots \}$

as 2+3++5 is not a valid formula it does not belong to the language of grammar.

2. **Derivation** : What does the language of a grammar means

" A derivation is the step by step process of applying grammar rules to generate a string from the start symbol of a grammar "

Points to Ponder

- Variables will be replaced by terminals and terminals make strings. All the strings that can be derived forms the language of the grammar

Example :- Consider the grammar

$$A \rightarrow aA$$
$$A \rightarrow \epsilon$$

Here $V = \{A\}$

$T = \{\epsilon, a\}$

$P = \{A \rightarrow aA,\ A \rightarrow \epsilon\}$

$S = A$

As, $A \Rightarrow aA \Rightarrow a(aA) \Rightarrow aa(\epsilon) \Rightarrow aa$

A derives aa

So aa comes in the language of grammar $G = (V, T, P, S)$

You just have to replace variables using production rules till string is formed of all non terminals

2.1 Leftmost Derivation and Rightmost Derivation in

If production rules $A \to \alpha$, the body α has more than one variable like

$A \to aBbA$

here B and A are variables, then we have options whether to produce / derive B first or A.

If the derivation is initiated by substituting leftmost variable then derivation is called leftmost derivation and if derivation is initiated by substituting rightmost variable the derivation is called rightmost derivation

Consider the grammar of pg 103

Leftmost derivation of "2+3"

$F \Rightarrow E$ $(F \to E)$
$\Rightarrow E+E$ $(E \to E+E)$
$\Rightarrow (N)+E$ $(E \to N)$
$\Rightarrow (N)+E$
$\Rightarrow (2)+E$ $(N \to 2)$
$\Rightarrow 2+(N)$ $(E \to N)$
$\Rightarrow 2+(3)$ $(N \to 3)$

*** $F \overset{*}{\underset{lm}{\Rightarrow}} 2+3$

Rightmost derivation of "2+3"

$F \Rightarrow E \Rightarrow E+E$
$\Rightarrow E+(N)$
$\Rightarrow E+(3)$
$\Rightarrow (N)+3$
$\Rightarrow (2)+3$

*** $F \overset{*}{\underset{rm}{\Rightarrow}} 2+3$

3 **Sentential Form**

A sentential form is any intermediate string of symbols that appears during a derivations from the start symbol toward a terminal string.

In

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

here

☞ $\overset{*}{\Rightarrow}$ * over production symbol means one or more than one substitution has produced

α

☞ $\alpha \in (V \cup T)^*$

this, α is called sentential form.

If $\alpha \in T^*$ then α is called sentence

Leftmost sential form and Rightmost sentential form

$$A \overset{*}{\underset{lm}{\Rightarrow}} \alpha$$: Left Sential form

Left sentential form means that α is being produced using leftmost derivation

$$A \overset{*}{\underset{rm}{\Rightarrow}} \alpha$$: Right Sential form

Right sentential form means that α is being derived using rightmost derivation

MEG ≡ Grammar for Mathematical Expressions

MON TUE WED THU FRI SAT SUN

Page No.: 107
Date: __/__/____

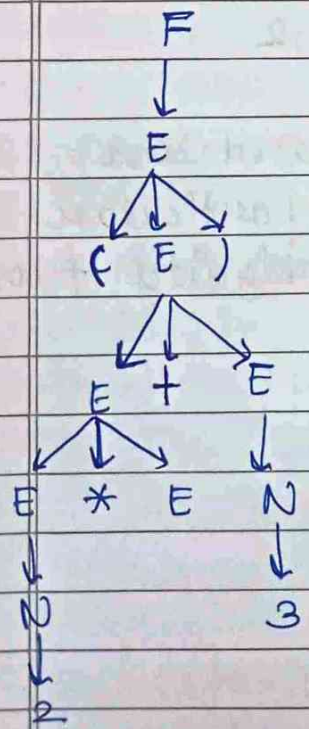
4 **Parse Tree**

The parse tree are trees for Grammar $G = (V, T, P, S)$ with following conditions :-

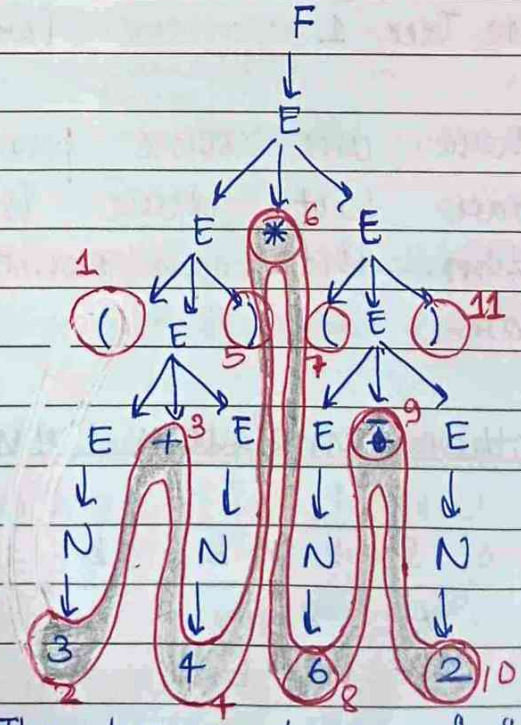
1. Each interior node is labeled by a variable V
2. Each leaf node is labeled by either a variable, a terminal or ϵ.
3. If the leaf is labeled ϵ, then it must be the only child of its parent
4. If an interior node is labeled a, and its children are labeled $X_1 X_2 X_3 \ldots X_n$, then $A \rightarrow X_1 X_2 X_3 \ldots X_n$ is a production in P

For Example :-

Consider the grammar of Pg 103 (MEG)



Valid Parse trees



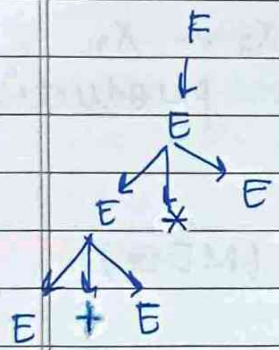
The parse tree giving yield as (3+4) * (6 - 2)

5 Ambiguous Grammar.

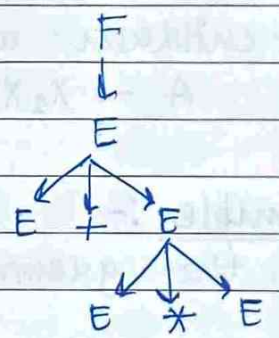
If more than one parse tree exist for any sentential form than the grammar is said to be ambiguous.

For Example

In Grammar for mathematical expressions E+E*E can have 2 different parse tree



Parse Tree 1



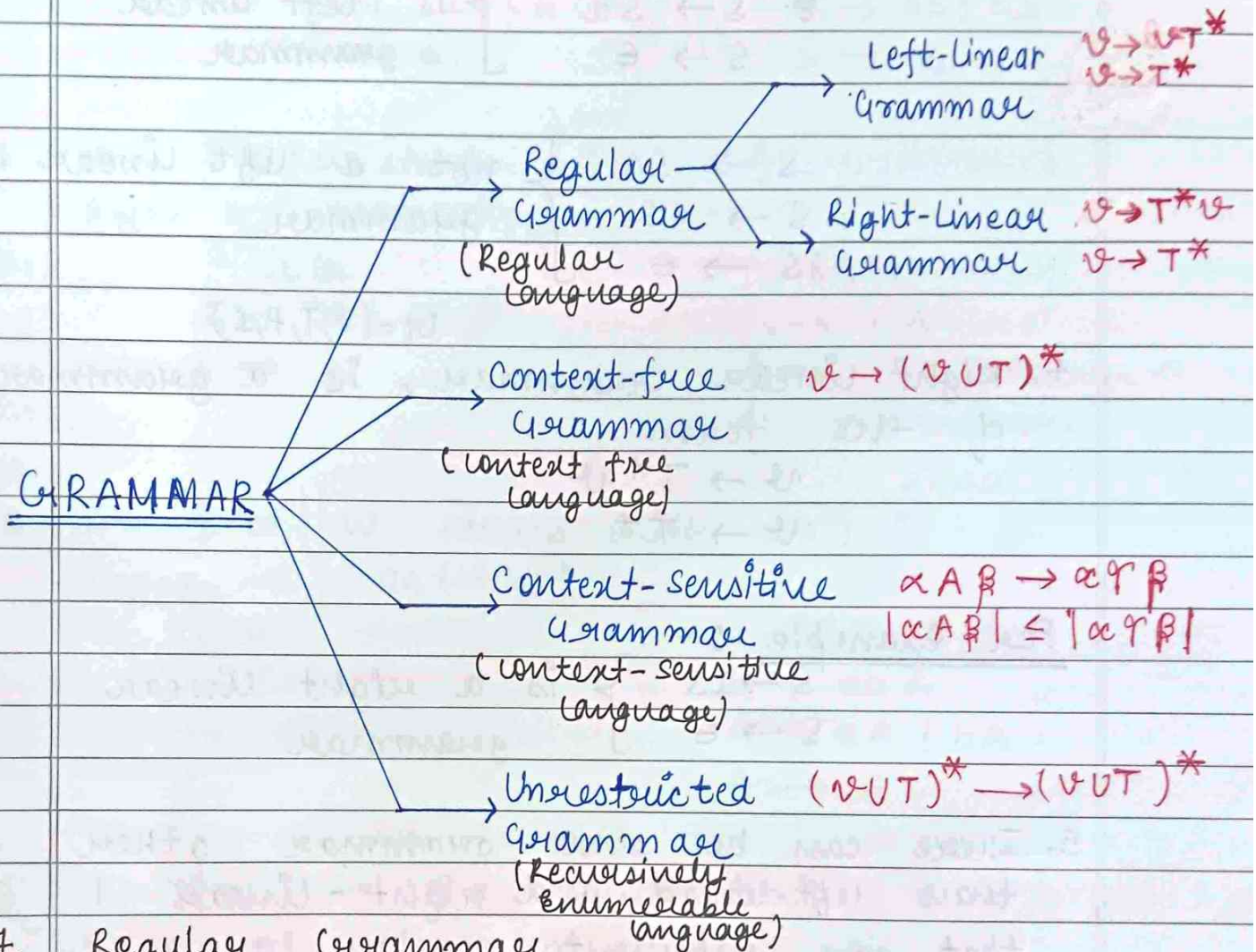
Parse Tree 2

☞ Yet there are some process based on grammar but there is no particular algorithm that can remove ambiguity from grammar

☞ **Some other ambiguous grammar**

$$S \rightarrow AB \mid ab$$
$$A \rightarrow a$$
$$B \rightarrow b$$

6 Classification of Grammar



7. Regular Grammar

1. A grammar is regular ~~if and only~~ if it is either right-linear or left-linear

2. The language represented/derived by a regular grammar is always a regular language.

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

3. Left-linear grammar is a grammar which is of the form

$$v \to vT^*$$
$$v \to T^*$$

For example :-

$S \to Sa$
$S \to \epsilon$
} is left linear grammar

$S \to Sa$
$S \to aS$
$S \to \epsilon$
} not a left linear grammar

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

4. Right linear grammar, is a grammar of the form

$V \to T^* V$
$V \to T^*$

For Example :-

$S \to aS$
$S \to \epsilon$
} is a right linear grammar

5. There can be some grammar other than left-linear and right-linear that can represent regular language.

Some important Regular Grammars

1 $L = \{\epsilon\}$

$S \to \epsilon$

2 $L = \{$all strings$\}$ $\Sigma = \{a\}$

$S \to Sa$
$S \to \epsilon$

OR

$S \to aS$
$S \to \epsilon$

3 $L = \{$ all strings $\}$ $\quad \Sigma = \{a, b\}$

$S \to Sa \mid Sb$
$S \to \epsilon$

OR

$S \to aS \mid bS$
$S \to \epsilon$

4 $L = \{ w \mid w$ starts with $a \}$

Regex = $a(a+b)^*$

LL Gr
$S \to Sa \mid Sb$
$S \to a$

RL Gr
$S \to aA$
$A \to aA \mid bA$
$A \to \epsilon$

5 $L = \{ w \mid w$ starts with $aa \}$

Regex = $aa(a+b)^*$

LL Gr
$S \to Sa \mid Sb$
$S \to aa$

RL Gr
$S \to aaA$
$A \to aA \mid bA$
$A \to \epsilon$

6 $L = \{ w \mid w$ ends with $bb \}$

Regex = $(a+b)^* bb$

LL Gr
$S \to Abb$
$A \to Aa \mid Ab$
$A \to \epsilon$

RL Gr
$S \to aS \mid bS$
$S \to bb$

7 $L = \{ w \mid w$ contains $aa \}$

Regex = $(a+b)^* aa (a+b)^*$

LL Gr
$S \to Sa \mid Sb$
$S \to Aaa$
$A \to Aa \mid Ab$
$A \to \epsilon$

RL Gr
$S \to aS \mid bS$
$S \to aaA$
$A \to aA \mid bA$
$A \to \epsilon$

8 $L = \{ w \mid w$ starts with aa or ab $\}$

Regex $= (a+b)^*(aa+ab)(a+b)^*$

LL Gr	RL Gr
$S \to Sa \mid Sb$	$S \to aS \mid bS$
$S \to Aaa \mid Bab$	$S \to aaA \mid abB$
$A \to Aa \mid Ab \mid \epsilon$	$A \to aA \mid bA \mid \epsilon$
$B \to Ba \mid Bb \mid \epsilon$	$B \to aB \mid bB \mid \epsilon$

9 $L = \{ w \mid |w| = 3 \}$

Regex $= (a+b)^3$

LL Gr	RL Gr
$S \to Aa \mid Ab$	$S \to aA \mid bA$
$A \to Ba \mid Bb$	$A \to aB \mid bB$
$B \to a \mid b$	$B \to a \mid b$

10 $L = \{ w \mid |w| \leq 3 \}$

Regex $= (\epsilon+a+b)^3$

LL Gr	RL Gr
$S \to Aa \mid Ab \mid \epsilon$	$S \to aA \mid bA \mid \epsilon$
$A \to Ba \mid Bb \mid \epsilon$	$A \to aB \mid bB \mid \epsilon$
$B \to a \mid b \mid \epsilon$	$B \to \epsilon \mid a \mid b$

11 $L = \{ w \mid |w| \geq 3 \}$

Regex $= (a+b)^3(a+b)^*$

LL Gr	RL Gr
$S \to Aa \mid Ab$	$S \to aA \mid bA$
$A \to Ba \mid Bb$	$A \to aB \mid bB$
$B \to Ca \mid Cb$	$B \to aC \mid bC$
$C \to Ca \mid Cb \mid \epsilon$	$C \to aC \mid bC \mid \epsilon$

12 $L = \{ w \mid \#_a(w) = 3 \}$

Regex = $b^*ab^*ab^*ab^*$

LLG	RLG
$S \to Sb$	$S \to bS$
$S \to Aa$	$S \to aA$
$A \to Ab$	$A \to bA$
$A \to Ba$	$A \to aB$
$B \to Bb$	$B \to bB$
$B \to Ca$	$B \to aC$
$C \to Cb \mid \epsilon$	$C \to bC \mid \epsilon$

13 $L = \{ w \mid \#_a(w) \leq 2 \}$

Regex = $b^*(\epsilon + a)\, b^*(\epsilon + a)\, b^*$

LLG	RLG
$S \to Sb$	$S \to bS$
$S \to Aa \mid A$	$S \to aA \mid A$
$A \to Ab$	$A \to bA$
$A \to Ba \mid B$	$A \to B \mid aB$
$B \to Bb \mid \epsilon$	$B \to bB \mid \epsilon$

14 $L = \{ a^m b^n \mid m \geq 0, n \geq 0 \}$

Regex = a^*b^*

LLG	RLG
$S \to Sb$	$S \to aS$
$S \to Aa$	$S \to Ab$
$A \to Aa \mid \epsilon$	$A \to Ab \mid \epsilon$

MON TUE WED THU FRI SAT SUN

Page No.: 114
Date: __/__/____

8 FA to Regular Grammar

8.1 Finite Automata to Left-Linear Grammar

Let a FA M be defined as $(Q, \Sigma, \delta, q_0, F)$
We construct a grammar $G = (V, T, P, S)$
The rules are

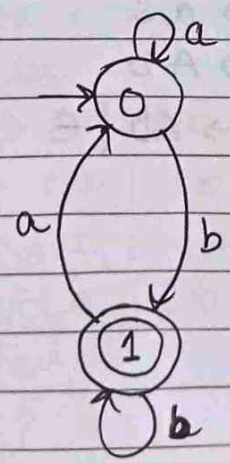
Rule 1: Each state becomes a non-terminal
Suppose A_q is non-terminal for $q \in Q$

Rule : 2 Transitions become left linear rules
$\delta(q, a) = p \iff A_p \to A_q a$
Destination → Origin + Symbol (DOS)

Rule : 3 Final state Transition
For every final state $f \in F$ add
$S \to A_f$ (Start symbol → Final)

Rule : 4 Start state Remedy :-
For the start state q_0 add
$A_{q_0} \to \epsilon$ (Start state variable → ϵ)

Example :



$G = (V, T, P, S)$ where $V = \{S, A_0, A_1\}$
$T = \{a, b\}$,
$P = \{ S \to A_1$ Rule 3
$A_0 \to \epsilon$ Rule 4
$A_0 \to A_0 a$ R2
$A_1 \to A_0 b$ R2
$A_0 \to A_1 a$ R2
$A_1 \to A_1 b \}$ R2

8.2 Finite Automata to Right Linear Grammar

Let a FA M be defined as $(Q, \Sigma, \delta, q_0, F)$
We construct a grammar $G = (V, T, P, S)$ which is right linear using following rules

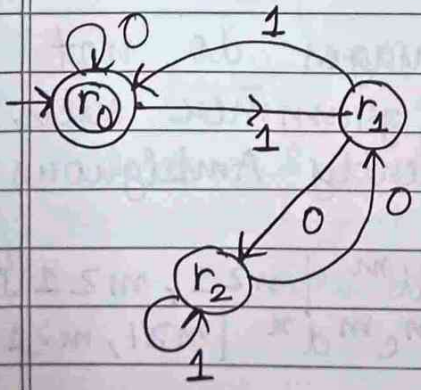
Rule 1 : Each state becomes a non-terminal
Suppose A_q is non-terminal for $q \in Q$

Rule 2 : Transitions becomes Right Linear Production
$\delta(q, a) = p \Leftrightarrow A_q \rightarrow a A_p$
Origin → Symbol + Destination (OSD)

Rule 3 : Final State Remedy
For every final state $f \in F$
$A_f \rightarrow \epsilon$ (Final State variable → ϵ)

Rule 4 : Start State Transition
For the start state q_0 add
$S \rightarrow A_{q_0}$ (Start symbol → start state variable)

Example :-



$G = (V, T, P, S)$
$V = \{0, 1\}$, $T = \{S, A_0, A_1, A_2\}$
$P = \{$
$S \rightarrow A_0$ Rule 4
$A_0 \rightarrow \epsilon$ Rule 3
$A_0 \rightarrow 0A_0 \mid 1A_1$ R2
$A_1 \rightarrow 0A_2 \mid 1A_0$ R2
$A_2 \rightarrow 0A_1 \mid 1A_2 \}$ R2

Point to Ponder

LL(k) parses the string backward while RL(k) parses the string forward

9 Context Free Grammar

1. A grammar 'G' is context-free if the productions of G are of the form
$$V \longrightarrow (V \cup T)^*$$
where V refers to any non-terminals and $(V \cup T)^*$ refers to any combination of terminal and non-terminal

2. The language represented by a context free grammar is context free language

3. CFL's are more powerful that regular language and apart from recognition of particular words they can represent nested structures like balanced parenthesis or arithmetic expression

4. Some context free languages do not have any unambiguous grammer such CFL's are called Inherently Ambiguous Language.
For Example : $L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

UG = Unambiguous Grammar
AG = Ambiguous Grammar

MON TUE WED THU FRI SAT SUN

5 Example of CFG is

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

where $V = \{S, A\}$

$T = \{a, b\}$

$P = \{ S \to aSb \mid A$

$A \to aA \mid a$

$\}$

10 Some Important CFGs

1 $L = \{\epsilon\}$

$S \to \epsilon$ (UG)

$S \to SS \mid \epsilon$ (AG)

2 $L = \{a\}$

$S \to a$

$S \to A$, $A \to a$

$S \to AB$, $A \to a$, $B \to \epsilon$

3 $L = \{a^* b^*\}$

$S \to AB$

$A \to Aa \mid \epsilon$

$B \to Bb \mid \epsilon$

Pg 110 Point 5

Not a Regular Grammar but represents regular language.

4 $L = \{a^m b^n \mid m \geq 0, n \geq 0\}$

$S \to aS \mid Sb \mid \epsilon$

$a^* S\, b^*$

5 $L = \{ b^m a^n \mid m \geq 0 , n \geq 0 \}$

$S \rightarrow bS \mid Sa \mid \epsilon$

6 $L = \{ b^m a^n \mid m \geq 1 , n \geq 0 \}$

$S \rightarrow bS \mid Sa \mid b$

7 $L = \{ a^m b^n \mid m \geq 1, n \geq 0 \text{ or } m \geq 0, n \geq 1 \}$

$S \rightarrow aS \mid Sb \mid a \mid b$

***8 $L = \{ a^m b^m \mid m \geq 0 \}$

$S \rightarrow aSb \mid \epsilon$

9 $L = \{ a^{m+1} b^m \mid m \geq 0 \}$

$S \rightarrow aSb \mid a$

10 $L = \{ a^m b^{m+1} \mid m \geq 0 \}$

$S \rightarrow aSb \mid b$

11 $L = \{ a^{2m} b^n \mid m \geq 0 \}$

$S \rightarrow aaSb \mid \epsilon$

12 $L = \{ a^{2m} b^{3m} \mid m \geq 0 \}$

$S \rightarrow aaSbbb \mid \epsilon$

***13 $L = \{ a^n a^* b^n \mid n \geq 0 \}$ or $L = \{ a^m b^n \mid m \geq n \}$

$S \rightarrow aSb \mid A$

$A \rightarrow Aa \mid \epsilon$

14 $L = \{ a^m b^n \mid m \leq n \}$

$S \rightarrow aSb \mid A$

$A \rightarrow bA \mid \epsilon$

***
15 $L = \{a^n b^n \mid n \geq 0\}^*$

$L = \{ \epsilon, ab, abaabb, abab, aabbaaabbbab \ldots \}$

$S \rightarrow SA \mid \epsilon$ (means SA^*)

$A \rightarrow aAb \mid \epsilon$

Point to Ponder

$L = \{(a^n b^n)^* \mid n \geq 0\}$ is not a CFL

This language means select repetitions first and then the value of n as $L = \{ababab, aabbaabb, aaabbbaaabbbaaabbb \ldots\}$

*****
16 $L = \{ww^R \mid w \in (a,b)^*\}$ (Even length palindrome)

$S \rightarrow aSa \mid bSb \mid \epsilon$

**
17 $L = \{ww^R \mid w \in a^*b^*\}$ (Palindrome with no a between b's)

$S \rightarrow ASA \mid B \mid \epsilon$

$A \rightarrow a$

$B \rightarrow CBC \mid \epsilon \mid b$

$C \rightarrow b$

18 $L = \{w(a+b)w^R \mid w \in (a,b)^*\}$ (Odd length palindromes)

$S \rightarrow aSa \mid bSb \mid \epsilon$

19 $L = \{$ Set of all palindromes even or odd $\}$

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

***
20 $L = \{a^m b^n c^l \mid m = n+l\}$

$S \rightarrow aSc \mid A$ (mean $a^n A c^n$)

$A \rightarrow aAb \mid \epsilon$ (A can be $a^n b^n$ or ϵ)

Page No.: 120
Date: __/__/____

21 $L = \{ a^m c^n b^l \mid m = n+l \}$

$S \to aSb \mid A$ ($S = a^n A b^n$)

$A \to aAc \mid \epsilon$ ($A = a^n c^n$)

22 $L = \{ a^m b^m c^n d^n \mid m \geq 0, m \geq 0 \}$

$S \to AB$

$A \to aAb \mid \epsilon$

$B \to cBd \mid \epsilon$

23 $L = \{ a^* b^* c^* \}$

$S \to ABC$

$A \to Aa \mid \epsilon$

$B \to Bb \mid \epsilon$

$C \to Cc \mid \epsilon$

***** 24 Grammar for all balanced parentheses

$S \to SS$

$S \to (S)$

$S \to \epsilon$

***** 25 Grammar for all mathematical expressions involving +, *

$E \to E+E$

$E \to E*E$

$E \to (E)$

$E \to id$

↑ (Ambiguous)

$E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E)$

$F \to id.$

(Unambiguous)

MON TUE WED THU FRI SAT SUN

11 Normal Forms

1. For Context Free grammars Normal forms play an important role both in identifying the grammar and identify the importantance of the variables and grammar.

2. Normal Form would be more precise and informative if all unused symbols are eliminated.

11.1 Eliminating ϵ - productions

Before ϵ - elimination	After ϵ- elimination
$S \to A_1 A_2 \cdots A_K \cdots A_n$	$S \to A_1 A_2 \cdots A_K \cdots A_n$
$A_K \to \epsilon$	$S \to A_1 A_2 \cdots A_{K-1} A_{K+1} \cdots A_n$

Example

$S \to SaA \mid Bb$
$A \to Aa \mid \epsilon$
$B \to Sa \mid \epsilon$

Eliminating $A \to \epsilon$

$S \to SaA \mid Sa \mid Bb$ — Added $S \to Sa$ for $S \to SaA$
$A \to Aa \mid a$ — Added $A \to a$ for $A \to Aa$
$B \to Sa \mid \epsilon$

Eliminating $B \to \epsilon$

$S \to SaA \mid Sa \mid Bb \mid b$
$A \to Aa \mid a$
$B \to Sa$

Note :-

If the language contains ϵ i.e. $L = \{\epsilon, \ldots\}$ then the transition $S \to \epsilon$ cannot be eliminated, in contrast if $S \to \epsilon$ is a production then it means language contains ϵ and hence cannot be eliminated

In short ϵ can be eliminated completely if and only if language does not contain ϵ

11.2 Eliminating Unit Production

A production of the form $v \to v$ is called unit production where v is a variable

Example : $S \to A$ is a unit production

☞ **Construct Unit Pair**

1. For all $A \in v$ (Variable) make (A, A) as unit pair
2. For all $B \in v - \{A\}$ if $A \to B$ is a production then (A, B) is a unit pair
3. For all $C \in v - \{A, B\}$ if $B \to C$ is a production then (A, C) is a unit pair

Eliminate Unit production using unit pairs

1. For all unit pairs (A, B) add all the transitions of B to A other than unit production

Example :

E → T | E+T
T → F | T*F
F → I | (E)
I → a | b | Ia | Ib | I0 | I1

Unit pairs	Productions Added
(E, E)	E → E+T
(E, T)	E → T*F
(E, F)	E → (E)
(E, I)	E → a | b | Ia | Ib | I0 | I1
(T, T)	T → T*F
(T, F)	T → (E)
(T, I)	T → a | b | Ia | Ib | I0 | I1
(F, F)	F → (E)
(F, I)	F → a | b | Ia | Ib | I0 | I1
(I, I)	I → a | b | Ia | Ib | I0 | I1

11.3 Eliminate Non-Generating Symbol

Non-Generating symbol are like traps which does not allow any strings or terminal to get produced.

Rules for identifying generating symbols

1. All terminals are generating
2. For production $A \rightarrow \alpha_1 \alpha_2 \alpha_3 \ldots \alpha_n$ if $\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_n$ are all generating then A is generating.

Example

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

Production	Generating Symbols
$A \rightarrow aaa$	A
$B \rightarrow a$	B
$S \rightarrow BaB$	S
$D \rightarrow \epsilon$	D

The Symbol C is non-generating

Removing Non-Generating Symbol

$S \rightarrow BaB$
$A \rightarrow aA \mid aaa$
$B \rightarrow bBb \mid a \mid D$
$D \rightarrow \epsilon$

11.4 Eliminate Reachable Symbol

A symbol X is reachable if there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta$ for some α and β

Example:

$S \rightarrow A$
$A \rightarrow aA \mid bAa \mid a$
$B \rightarrow bbA \mid aB \mid AB$
$C \rightarrow aCa \mid aD$
$D \rightarrow aD \mid bC$

It is necessary to eliminate non-generating symbols before eliminating reachable symbol

Here C and D are non-generating

$S \rightarrow A$
$A \rightarrow aA \mid bAa \mid a$
$B \rightarrow bbA \mid aB \mid AB$

As we cannot reach B from S hence B is unreachable.

11.5 Chomsky Normal Form.

1 A CFG is in Chomsky Normal Form if every production rule is of the type

1 $A \rightarrow BC$ $A, B, C \in V$ (variable)
2 $A \rightarrow a$ $a \in T$ (Terminal)
3 $S \rightarrow \epsilon$ if Language contains ϵ

2 Steps to convert CFG in CNF

Step 1: Add new start symbol
$S_0 \rightarrow S$

Step 2: Eliminate ϵ - production

Step 3: Eliminate Unit production

Step 4: Eliminate Non-Generating symbol

Step 5: Eliminate Non-Reachable symbol

Step 6: Any production of the type
$S \rightarrow ABCD$ will be transfored as
$S \rightarrow AX_1$
$X_1 \rightarrow BX_2$
$X_2 \rightarrow CD$

Step 7: Any production of the type
$S \rightarrow aA$ will be transformed as
$S \rightarrow X_1 A$
$X_1 \rightarrow a$

Example :-
$S \rightarrow AAA \mid B$
$A \rightarrow aA \mid B$
$B \rightarrow \epsilon$

Adding New start state

$S_0 \rightarrow S$
$S \rightarrow AAA \mid B$
$A \rightarrow aA \mid B$
$B \rightarrow \epsilon$

Eliminating ϵ Production

$S_0 \rightarrow S$
$S \rightarrow AAA \mid \epsilon$
$A \rightarrow aA \mid a$
$S \rightarrow AA \mid A$

Eliminating Unit Production

Unit Pair	Productions
(S_0, S_0)	No production
(S_0, S)	$S_0 \rightarrow AAA \mid AA \mid A \mid \epsilon$
(S_0, A)	$S_0 \rightarrow aA$
(S, A)	$S \rightarrow aA$
(S, S)	$S \rightarrow AAA \mid AA \mid \epsilon$
(A, A)	$A \rightarrow aA \mid a$

Eliminating Non-Generating Symbol

$S_0 \rightarrow AAA \mid AA \mid \epsilon \mid aA$
$S \rightarrow AAA \mid AA \mid \epsilon \mid aA$
$A \rightarrow aA \mid a$

Eliminating Non-Reachable Symbol

$S \to AAA \mid AA \mid aA \mid \epsilon$
$A \to aA \mid a$

Chomsky Normal Form

$S \to AB_1$
$B_1 \to AA$
$S \to AA$
$S \to B_2A$
$B_2 \to a$
$A \to B_3A$
$B_3 \to a$
$S \to \epsilon$
$A \to a$

$\Rightarrow$

$S \to AB_1 \mid AA \mid B_2A \mid \epsilon$
$A \to B_2A \mid a$
$B_1 \to AA$
$B_2 \to a$

11.6 Greibach Normal Form (GNF)

A CFG is in Greibach Normal Form if all of its production are of the form

$$A \to a\alpha$$

where $A \in$ Variable
$a \in$ Non-Terminal
α is a string of non-terminal or ϵ

Steps to convert CFG into CNF

Step 1 Convert CFG to CNF

Step 2 Eliminate Left Recursion

Step 3 If there is any production of the form

$$A \rightarrow B\gamma$$

replace it recursively with the production of B till you get a terminal

Eliminating Left Recursion

Left recursion in a production is of the form

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \cdots A\alpha_n \mid \beta_1 \mid \beta_2 \cdots \beta_k$$

means,

variable having a body whose first symbol is that variable only

After elimination of left Recursion

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \cdots \beta_k A'$$

$$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \alpha_3 A' \cdots \alpha_n A'$$