Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Theory of Computation
SY Btech CSE-AIDS

CONTEXT FREE GRAMMAR(CFG)

Course Objectives & Course Outcomes

Course Objectives:

- To understand the basics of automata theory and its operations.
- To understand problem classification and problem solving by machines.
- To study computing machines by describing, classifying and comparing different types of computational models.
- To understand the fundamentals of decidability and computational complexity.

Course Outcomes:

- After completion of this course students will be able:
- To construct finite state machines to solve problems in computing.
- To write mathematical expressions and syntax verification for the formal languages.
- To construct and analyze Push Down Automata and Turing Machine for formal languages.
- To express the understanding of decidability and complexity.

Text Books & Reference Books

- **Text Books**

•John C. Martin, Introduction to Language and Theory of Computation, TMH, 3rdEdition, ISBN: 978-0-07-066048-9.

•Vivek Kulkarni, Theory of Computation, Oxford University Press, ISBN-13: 978-0-19-808458-7.

- **Reference Books**

•K.L.P Mishra,N. Chandrasekaran,Theory of Computer Science (Automata, Languages and Computation), Prentice Hall India, 2nd Edition.

•Michael Sipser, Introduction to the Theory of Computation,CENGAGE Learning, 3rd Edition, ISBN:13:978-81-315-2529-6.

•Daniel Cohen, Introduction to Computer Theory, Wiley India, 2nd Edition, ISBN: 9788126513345.

•Kavi Mahesh, Theory of Computation: A Problem Solving Approach, 1st Edition, Wiley-India, ISBN: 978-81-265-3311-4.

Unit III – Context Free Grammar

CFG

Formal definition of Grammar, Chomsky Hierarchy, **CFG :** Formal definition of CFG, Derivations, Parse Tree, Ambiguity in grammars and languages, Language Specification using CFG, Normal Forms: Chomsky Normal Form and Greibach Normal Form. Closure properties of CFL.

Introduction

- Finite Automata **accept** all regular languages and only regular languages

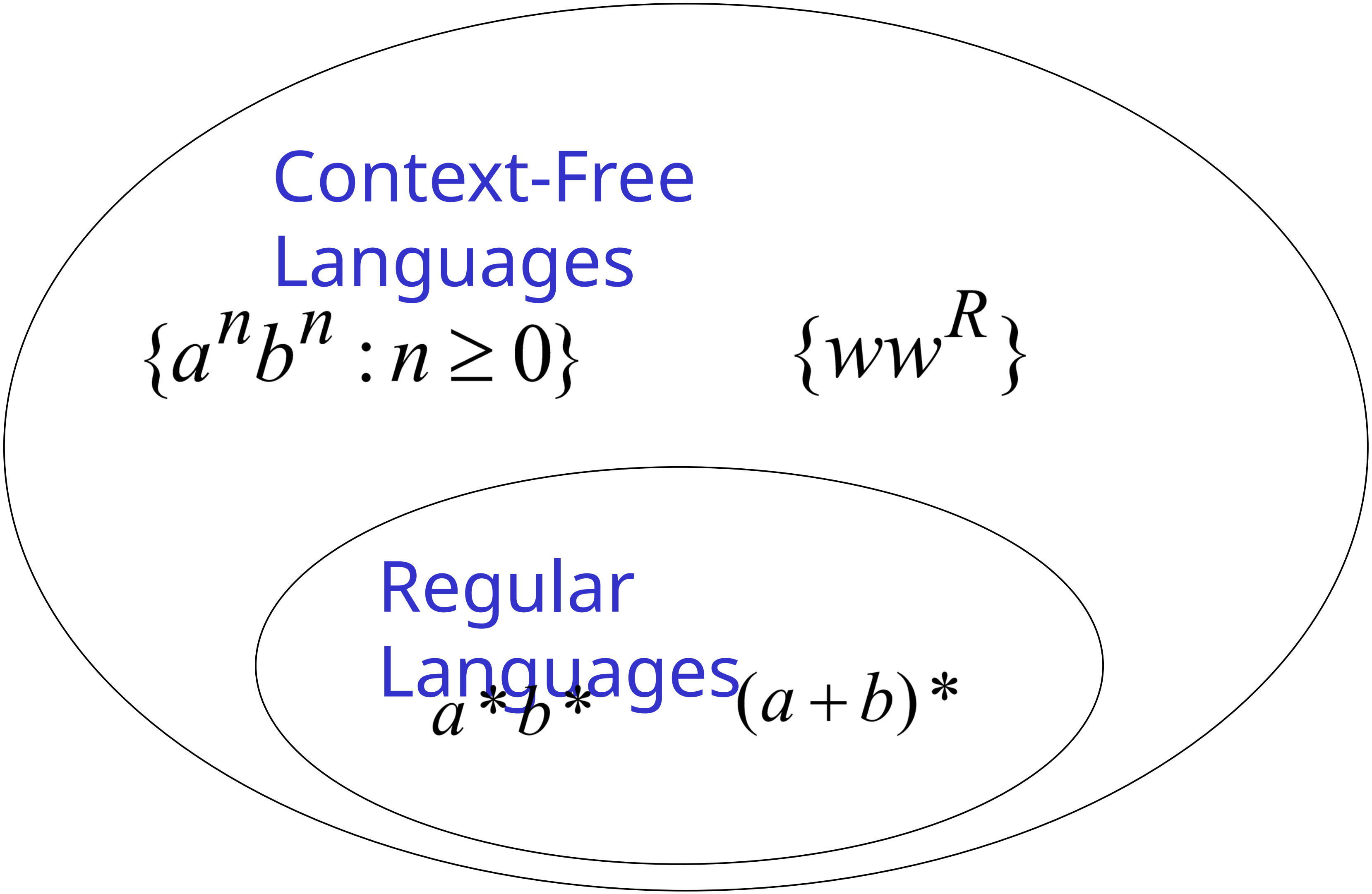
- Many simple languages are non regular:

  - $\{a^n b^n : n = 0, 1, 2, \ldots\}$
  - $\{w : w$ a is palindrome$\}$

  and there is no finite automata that accepts them.

- **Context-Free Languages** are a larger class of languages that encompasses all regular languages and many others, including the two above.

Context-Free Languages and Regular Languages



Formal Definition of a Grammar

Gramma
r:

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

Set of
variable
s

Set of
terminal
symbols

Start
variabl
e

Set of
production
s

Language for the Grammar

Grammar:

$S \rightarrow aSb$

$S \rightarrow \epsilon$

Language of the grammar:

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

A Convenient Notation

We write: $S \overset{*}{\Rightarrow} aaabbb$

for zero or more derivation steps

Instead of:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Generalizing:

$$w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \cdots \Rightarrow w_n$$

Trivially:

$$w \overset{*}{\Rightarrow} w$$

Formal Definition-CFG

Definition: A context-free grammar is a 4-tuple (V, T, P, S), where:

- V is a set (each element in V is called **nonterminal**)
- T is an alphabet (each character in T is called **terminal**)
- P, the set of rules, is a subset of $V \times (T \cup V)^*$

    If $(\alpha,\beta) \in P$, we write production $\alpha \to \beta$

    β is called a **sentential form**

- S, the **start symbol**, is one of the symbols in V

Example of Context-Free Grammar

productions

$P = \{S \to aSb, S \to \epsilon\}$

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

$V = \{S\}$

variables

$T = \{a, b\}$

terminals

start variable

Context-Free Language

A language L is context-free

if there is a context-free grammar G

$$L = L(G)$$

with

Context-Free Language-Example 1

$$L = \{a^n b^n : \quad n \geq 0\}$$

is a context-free language

since context-free grammar : G

$$\{S \to aSb \mid \epsilon\}$$

generates $L(G) = L$

Context-Free Language – Example 2

Context-free grammar : G

{S✉aSa | bSb | ϵ}

Example derivations:

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$$

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaSaba \Rightarrow abaaba$$

$$L(G) = \{ww^R : \; w \in \{a,b\}^*\}$$

Palindromes of even length

Derivation and Parse Trees

Consider the following example grammar with 5 productions:

$S \rightarrow AB$ $\quad A \rightarrow aaA|\epsilon$ $\quad B \rightarrow Bb|\epsilon$

Leftmost and Rightmost Derivation

Consider the following example grammar with 5 productions:

S → AB   A → aaA|ϵ   B → Bb|ϵ

Leftmost derivation: for the string aab

$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$$

Rightmost derivation: for the string aab

$$S \Rightarrow AB \Rightarrow ABb \Rightarrow Ab \Rightarrow aaAb \Rightarrow aab$$

Leftmost and Rightmost Derivation

$S \rightarrow AB$ $\quad A \rightarrow aaA \mid \epsilon$ $\quad B \rightarrow Bb \mid \epsilon$

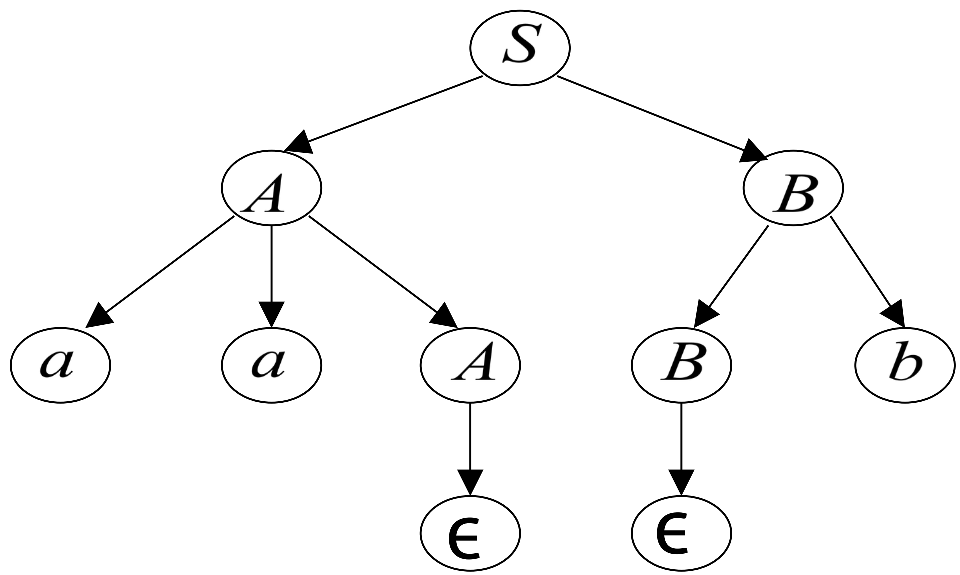
Leftmost derivation:

$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$$

Rightmost derivation:

$$S \Rightarrow AB \Rightarrow ABb \Rightarrow Ab \Rightarrow aaAb \Rightarrow aab$$

Give same derivation tree



Context Free Language for a CFG

Find the CFL generated for the given CFG.

1. S→aSb | ab

L(G)={$a^n b^n$ | n>=1}

2. S→aB|bA

   A→a|aS|bAA

   B→b|bS|aBB

L(G)={x|x containing equal no of a’s and b’s}

CFG for a CFL

1. Write the grammar for generating strings over $\Sigma=\{a\}$, containing any(zero or more) number of a's.

   $S \rightarrow a \mid aS \mid \epsilon$

2. Find the CFG represented by the RE $(a+b)^*$

   $S \rightarrow aS \mid bS \mid \epsilon$

3. Write the grammar for all strings consisting of a's and b's with atleast 2 a's.

   $S \rightarrow AaAaA$

   $A \rightarrow aA \mid bA \mid \epsilon$

Ambiguous Grammar

A context-free grammar G is ambiguous if there is a string $w \in L(G)$ which has:

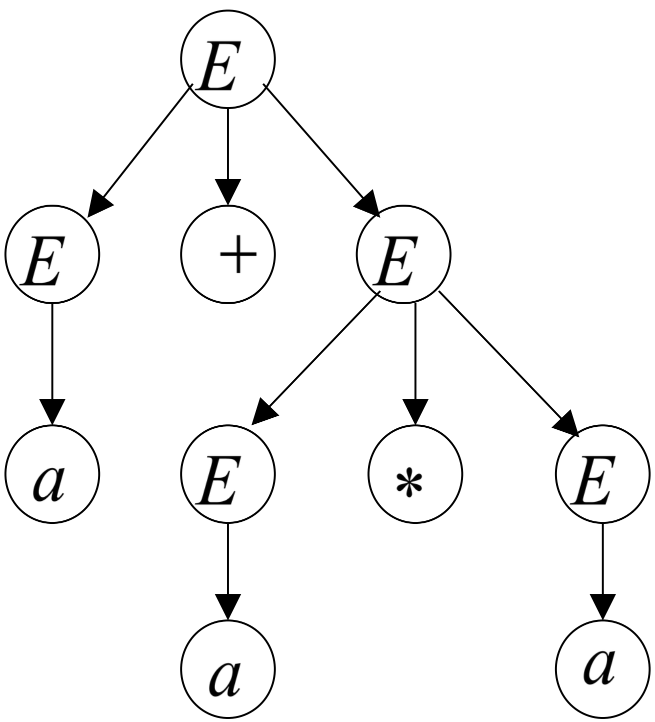
two different derivation trees
or
two leftmost derivations

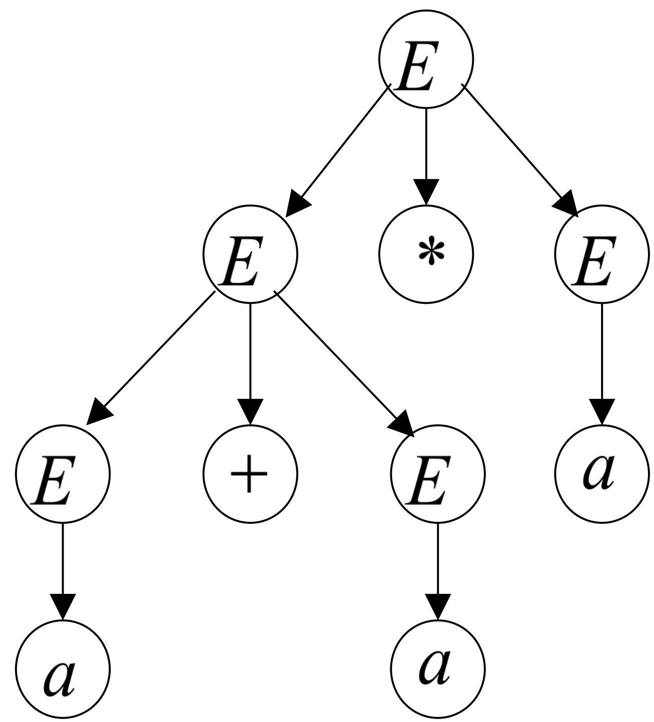
(Two different derivation trees give two different leftmost derivations and vice-versa)

Ambiguous Grammar – An Example

$$E \to E + E \;\mid\; E * E \;\mid\; (E) \;\mid\; a$$

string $a + a * a$ has two derivation trees





Ambiguous Grammar – An Example

$$\boxed{E \to E + E \mid E * E \mid (E) \mid a}$$

string $a + a * a$ has two leftmost derivations

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

Removing Ambiguity

Ambiguous Grammar

$$E \to E + E$$
$$E \to E * E$$
$$E \to (E)$$
$$E \to a$$

Equivalent Non-Ambiguous Grammar

$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid a$$

generates the same language

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T*F$$
$$\Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow a+a*a$$

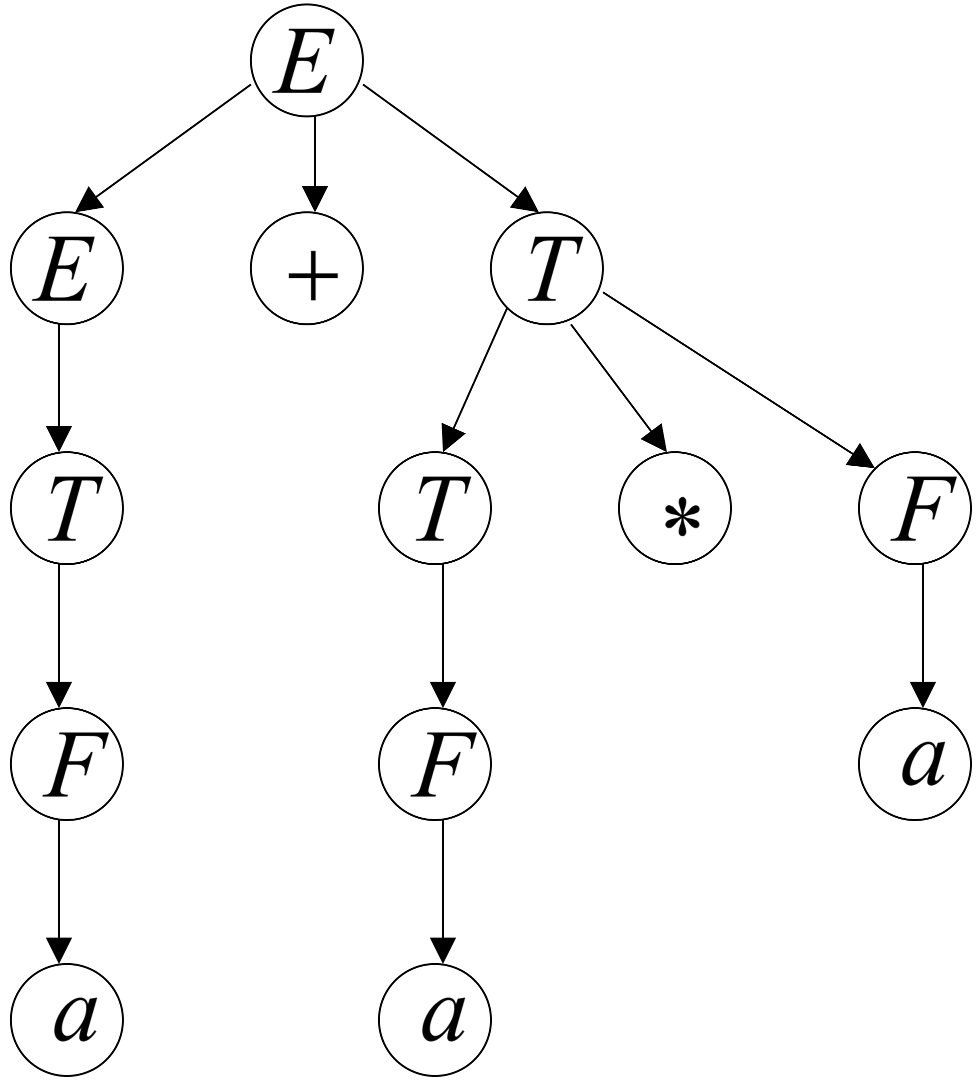
$$E \rightarrow E+T \mid T$$

$$T \rightarrow T*F \mid F$$

$$F \rightarrow (E) \mid a$$

Unique derivation tree and leftmost derivation

For the string a+a*a



Simplification of Context Free Grammar

A CFG G can be simplified as:

1. Each variable and terminal of the CFG should appear in the derivation if at least one word in the L(G).
2. There should not be any production of the form A→B where A and B are both non-terminals.

Simplification techniques are:

1. Removal of Useless Symbols
2. Removal of Unit Productions
3. Elimination of ∈ production

Removal of Useless symbol

Ex.1 Consider the following grammar:

G={ (S,A),(1,0),P,S}

Where P consists of the following productions:

S✉1 0|0 S 1|1 S 0|A|S S

Simplify the grammar by removing the useless symbols if any.

Remove S✉A, as A is useless symbol.

S✉1 0|0 S 1|1 S 0|S S

Removal of Useless symbol

Ex.2 Consider the following grammar:

G={ (S,A,B),(a),(1,0),P,S}

Where P consists of the following productions:

S✉ A B | a

A✉a

Simplify the grammar by removing the useless symbols if any.

B is useless, Remove B

S✉A | a

A✉ a

Removal of Unit Production

- A production rule of the form $A \rightarrow B$ where A and B are both non-terminals is called a unit production.
- All the other productions (including $\in$ productions) are non-unit productions.

Ex.1 Consider the following grammar:

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

Where P consists of :

$A \rightarrow B$, $B \rightarrow a \mid b$

Simplify the grammar by removing the unit productions if any.

On eliminating unit production $A \rightarrow B$

$A \rightarrow a|b$

Removal of Unit Production

Ex.2 Consider the following grammar:

S→ a | X b | a Y a | b | a a

X →Y

Y → b | X

Simplify the grammar by removing the unit productions if any.

Simplified grammar is:

S→a | Yb | aYa | b | aa

Y→ b

Removal of ∈ - Production

A production of the form A✉ ∈ where A is non-terminal is known as ∈ - Production.

Ex.1 Consider the following grammar,

S✉ a S a | b S b | ∈

Simplify the grammar by eliminating ∈ - Productions if any.

Simplified grammar is G’:

S ✉ aSa | bSb | aa | bb

G’=G-ϵ rules.

L(G’)=L(G)-{ϵ}

Removal of ∈ - Production

Ex.2 Consider the following grammar,

S $\rightarrow$ a | X b | a Y a

X $\rightarrow$ Y | ∈

Y $\rightarrow$ b | X

Simplify the grammar by eliminating ∈ - Productions if any.

Simplified grammar is:

S $\rightarrow$ a | X b | a Y a | b | aa

X $\rightarrow$ Y

Y $\rightarrow$ b | X

Chomsky Hierarchy

1. Type 0 grammar (Unrestricted Grammar)
2. Type 1 grammar (Context Sensitive Grammar)
3. Type 2 grammar (Context Free Grammar)
4. Type 3 grammar (Regular Grammar)

Chomsky Hierarchy

- **Type 3 grammar:**

  It is also called as **regular grammar.**

  $A \rightarrow \alpha$

Recognized by **FSM.**

Two types of Regular Grammar are:

1. Left Linear Grammar 2. Right Linear Grammar

- **Type 2 grammar:**

  It is also called as **context free grammar.**

  $A \rightarrow \alpha$ where A is NT and α is sentential form.

  Start symbol of the Grammar can also appear on RHS.

Recognized by **PDA(Pushdown Automata)**

Chomsky Hierarchy

- ## Type 1 grammar:

It is **context sensitive grammar** or context dependent.

1. $\alpha \rightarrow \beta$ where length of β is at least as much as the length of α except $S \rightarrow \in$.
2. The rule $S \rightarrow \in$ is allowed only if start symbol S does not appear on RHS.
3. Productions are of the form

$$\alpha_1 A \alpha_2 \rightarrow \alpha 1\ \beta\ \alpha_2\ (\beta \neq \in)$$

Recognized by **TM(Turing Machine).**

Chomsky Hierarchy

- **Type 0 grammar:**

  It is **unrestricted grammar** that is no restriction on production.

  $\alpha \rightarrow \beta \qquad (\alpha \neq \in)$

Recognized by TM(Turing machines)

These languages are known as **Recursively Enumerable Languages.**

Normal Forms

There are certain standard ways of writing CFG.

They satisfy certain restrictions on the productions in the CFG.

Then the G is said to be in **Normal forms.**

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

Chomsky Normal Form(CNF)

A CFG is in CNF if every production is of the form

A✉ a, where A is NT and a is T

A ✉ BC where A,B and C are NTs

S ✉ ∈is in G if ∈ belongs to L(G).

When ∈ is in L(G),

we assume that S does not appear on the RHS of any production.

e.g. G is :

S✉AB| ∈

A✉ a

B ✉ b

Construction of G in CNF

Step 1. Elimination of null productions and unit productions using

previous method.

Let the G is (V,T,P,S)

Step 2. Elimination of terminals on RHS.

Step 3. Restricting the number of variables on RHS.

Reduce to CNF

1. Convert to CNF, $S \rightarrow aSa \mid bSb \mid a \mid b \mid aa \mid bb$

Adding $A \rightarrow a$ and $B \rightarrow b$ we can rewrite the grammar as

$S \rightarrow ASA|BSB|a|b|AA|BB$

$A \rightarrow a$, $B \rightarrow b$

Only $S \rightarrow ASA$ and $S \rightarrow BSB$ are not in CNF

For $S \rightarrow ASA$ we can write

$S \rightarrow AR_1$, $R_1 \rightarrow SA$

For $S \rightarrow BSB$ we can write

$S \rightarrow BR_2$, $R_2 \rightarrow SB$

Grammar in CNF is

$S \rightarrow AR_1|BR_2|a|b|AA|BB$

$A \rightarrow a$, $B \rightarrow b$

$R_1 \rightarrow SA$, $R_2 \rightarrow SB$

Reduce to CNF

2. Convert to CNF,

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

Add $R_1 \rightarrow a$, $R_2 \rightarrow b$

Rewriting the grammar,

$S \rightarrow R_2 A|\ R_1 B$,

$A \rightarrow R_2 AA|\ R_1 S|a$,

$B \rightarrow R_1 BB|\ R_2 S|b$

$R_1 \rightarrow a$, $R_2 \rightarrow b$

$A \rightarrow R_2 AA$ and $B \rightarrow R_1 BB$ are not in CNF

Converted grammar in CNF is

$S \rightarrow R_2 A|\ R_1 B$,

$A \rightarrow R_2 R_3|\ R_1 S|a$, $R_3 \rightarrow AA$

$B \rightarrow R_1 R_4|\ R_2 S|b$, $R_4 \rightarrow BB$

$R_1 \rightarrow a$, $R_2 \rightarrow b$

Greibach Normal Form

A context free grammar is said to be in Greibach Normal Form if all productions are in the following form:

$$A \rightarrow \alpha X$$

- A is a non terminal symbols
- α is a terminal symbol
- X is a sequence of non terminal symbols.

  It may be empty.

Closure properties of CFL

CFLs are closed under:

- Union
- Concatenation
- Kleene closure operator
- Substitution
- Homomorphism, inverse homomorphism
- Reversal

CFLs are *not* closed under:

- Intersection
- Difference
- Complementation