Context Free Languages and Grammer
Unit 3

Unit: 3

TAFL

Course Details
(B Tech 4th Sem)

Evaluation Scheme

EVALUATION SCHEME
SEMESTER -IV

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Schemes				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
1	AAS0404	Optimization and Numerical Techniques	3	1	0	30	20	50		100		150	4
2	AASL0401	Technical Communication	2	1	0	30	20	50		100		150	3
3	ACSE0403A	Operating Systems	3	0	0	30	20	50		100		150	3
4	ACSAI0402	Database Management Systems	3	1	0	30	20	50		100		150	4
5	ACSAI0403	Introduction to Information Security and Cryptography	3	0	0	30	20	50		100		150	3
6	ACSE0404	Theory of Automata and Formal Languages	3	0	0	30	20	50		100		150	3
7	ACSE0453A	Operating Systems Lab	0	0	2				25		25	50	1
8	ACSAI0452	Database Management Systems Lab	0	0	2				25		25	50	1
9	ACSAI0453	Introduction to Cryptography Lab	0	0	2				25		25	50	1
10	ACSE0459	Mini Project using Open Technology	0	0	2				50			50	1
11	ANC0402 / ANC0401	Environmental Science / Cyber Security	2	0	0	30	20	50		50		100	
12		MOOCs (For B.Tech. Hons. Degree)											
		GRAND TOTAL										**1100**	**24**

Syllabus

UNIT-I	Basic Concepts of Formal Language and Automata Theory	8 Hours
Introduction to Theory of Computation- Alphabet, Symbol, String, Formal Languages, Grammar, Derivation and Language generation by Grammar, Chomsky Hierarchy, Finite Automata, Deterministic Finite Automaton (DFA)- Definition, Representation, Acceptability of a String and Language, Non-Deterministic Finite Automaton (NFA), Equivalence of DFA and NFA, NFA with ∈-Transition, Equivalence of NFA's with and without ∈-Transition, Finite Automata with output- Moore Machine, Mealy Machine, Equivalence of Moore and Mealy Machine, Minimization of Finite Automata, Myhill-Nerode Theorem, Simulation of DFA and NFA.		
UNIT-II	**Regular Language and Finite Automata**	**8 Hours**
Regular Expressions, Transition Graph, Kleen's Theorem, Finite Automata and Regular Expression- Arden's theorem, Algebraic Method Using Arden's Theorem, Regular Grammars-Right Linear and Left Linear grammars, Conversion of FA into Regular grammar and Regular grammar into FA, Regular and Non-Regular Languages- Closure properties of Regular Languages, Pigeonhole Principle, Pumping Lemma, Application of Pumping Lemma. Decidability- Decision properties, Finite Automata and Regular Languages, Simulation of Transition Graph and Regular language.		
UNIT-III	**Context Free Language and Grammar**	**8 Hours**
Context Free Grammar (CFG)-Definition, Derivations, Languages, Derivation Trees and Ambiguity, Simplification of CFG, Normal Forms- Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Pumping Lemma for CFL, Closure properties of CFL, Decision Properties of CFL		

Syllabus

UNIT-IV	Push Down Automata	8 Hours
Pushdown Automata- Definition, Representation, Instantaneous Description (ID), Acceptance by PDA, Nondeterministic Pushdown Automata (NPDA)- Definition, Moves, Pushdown Automata and Context Free Language, Pushdown Automata and Context Free Grammar, Two stack Pushdown Automata.		
UNIT-V	**Turing Machine and Undecidability**	**8 Hours**
Turing Machine Model, Representation of Turing Machines, Language Acceptability of Turing Machines, Techniques for Turing Machine Construction, Variations of Turing Machine, Turing Machine as Computer of Integer Functions, Universal Turing machine, Linear Bounded Automata, Church's Thesis, Recursive and Recursively Enumerable language, Closure Properties of Recursive and Recursively Enumerable Languages, Non-Recursively Enumerable and Non-Recursive Languages, Undecidability, Halting Problem, Undecidability of Halting Problem, Post's Correspondence Problem.		

Branch wise Application

The Applications of these Automata are given as follows:

1. Finite Automata (FA) –
For the designing of lexical analysis of a compiler.
For recognizing the pattern using regular expressions.
Used in text editors.
For the implementation of spell checkers.

2. Push Down Automata (PDA) –
For designing the parsing phase of a compiler (Syntax Analysis).
For implementation of stack applications.
For evaluating the arithmetic expressions.
For solving the Tower of Hanoi Problem.

4. Turing Machine (TM) –
For solving any recursively enumerable problem.
For implementation of neural networks.
For implementation of Robotics Applications.
For implementation of artificial intelligence.

Course Objectives

The primary objective of this course is to introduce students to the foundations of computability theory. The other objectives include:

- Introduce concepts in automata theory and theory of computation
- Identify different formal language classes and their relationships
- Design grammars and recognizers for different formal languages
- Prove or disprove theorems in automata theory using its properties
- Determine the decidability and intractability of computational problems

NIET Greater Noida
GET FUTURE READY

Content

Context Free Grammar (CFG)-Definition, Derivations, Languages, Derivation Trees and Ambiguity, Simplification of CFG, Normal Forms- Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Pumping Lemma for CFL, Closure properties of CFL, Decision Properties of CFL

Course Outcome

Theory of Automata and Formal Languages (ACSE0404)		
Course Outcome (CO)		Bloom's Knowledge Level (KL)
At the end of the course, the student will be able to understand		
CO 1	Design and Simplify automata for formal languages and transform non-deterministic finite automata to deterministic finite automata.	K6
CO 2	Identify the equivalence between the regular expression and finite automata and apply closure properties of formal languages to construct finite automata for complex problems.	K3
CO 3	Define grammar for context free languages and use pumping lemma to disprove a formal language being context- free.	K3
CO 4	Design pushdown automata (PDA) for context free languages and Transform the PDA to context free grammar and vice-versa.	K6
CO 5	Construct Turing Machine for recursive and recursive enumerable languages. Identify the decidable and undecidable problems.	K6

CO-PO and PSO Mapping

CO-PO correlation matrix of Theory of Automata and Formal Languages (ACSE0404)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ACSE0404.1	**2**	**2**	**3**	**3**	**2**	**2**	**-**	**-**	**2**	**1**	**-**	**3**
ACSE0404.2	**1**	**3**	**2**	**3**	**2**	**2**	**-**	**1**	**1**	**1**	**2**	**2**
ACSE0404.3	2	2	3	2	2	2	-	2	2	1	2	3
ACSE0404.4	2	2	2	3	2	2	-	2	2	1	1	3
ACSE0404.5	3	2	2	2	2	2	-	2	1	1	1	2
Average	2	2.2	2.4	2.6	2	2	-	1.4	1.6	1	1.2	2.6

Mapping of Program Specific Outcomes and Course Outcomes:

Course Outcomes	Program Specific Outcomes			
	PSO1	PSO2	PSO3	PSO4
ACSE0404.1	**2**	**2**	**2**	**2**
ACSE0404.2	**2**	**2**	**1**	**1**
ACSE0404.3	2	2	1	1
ACSE0404.4	2	2	1	1
ACSE0404.5	2	2	2	2
Average	2	2	1.4	1.4

Program Educational Objectives

PEO1: To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and solve real-life problems using state-of-the-art technologies.

PEO2: To lead a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors so that engineering graduates can face the global challenges.

PEO3: To effectively bridge the gap between industry and academia through effective communication skill, professional attitude, ethical values and a desire to learn.

PEO4: To provide highly competitive environment and solidarity to students for successful professional career as engineer, scientist, entrepreneur and bureaucrats for the betterment of society.

CO-PEO correlation matrix

CO	PEO			
	PEO1	PEO2	PEO3	PEO4
CO1	2	2	1	2
CO2	2	2	1	1
CO3	2	2	2	2
CO4	2	2	1	1
CO5	2	2	1	2
Average	2	2	1.2	2

Result Analysis

Subject Result:

Section A: 95.65%
Section B: 97.14%
Section C: 84.51%
Section D: 94.29%

End Semester Question Paper Template

B TECH

(SEM-IV) THEORY EXAMINATION 20__-20__

Theory of Automata and Formal Languages

Time: 3 Hours **Total Marks: 100**

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt all questions in brief. **2 x 10 = 20**

Q.No.	Question	Marks	CO
1		2	
2		2	
.		.	
10		2	

End Semester Question Paper Templates

SECTION B

2. Attempt any three of the following: 3 x 10 = 30

Q.No.	Question	Marks	CO
1		10	
2		10	
.		.	
5		10	

SECTION C

3. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

End Semester Question Paper Templates

4. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

5. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

6. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

Unit Objective

- To Understand the Regular Grammar.
- To understand the context free grammar.
- To understand the relationship between FA and Context Free Grammars.
- To understand the Normal Forms in Grammar .

Objective of the Topic

The objective of the topic is to make the student able to:

- Understand the CFG.
- Realize the expressive power of CFG,CFL.
- Understand pumping lemma for CFL.

Prerequisite and Recap

Prerequisite:

- Basic concept of rules of grammars of various languages, concept of tree.

Recap:

- Regular expressions are the way of representation of the FA.
- RE can be converted into FA.
- FA can be converted into regular expressions.
- Any language can be regular or not which will be proved by Pumping Lemma.

Introduction (CO1, CO2)

Recap: In previous unit we have learned about RE and different Theorems.

Prerequisite: Basic knowledge about Finite Automata and Regular Expressions.

Objective: To understand the basic definition of grammar.

Introduction (CO1, CO2)

Objective: To understand the basic definition of grammar.

A grammar consists of :

- A set of variables (non terminals), one of which is represented as start symbol; It is customary to use upper case letters for variables.
- A set of terminals (customary use lower case letters), and
- A list of productions (set of rules).

Grammars (CO1, CO2)

A formal definition of a Grammar consists of :

- A finite set of rewriting rules in the form of :

$$\alpha \rightarrow \beta,$$

where α and β are the strings of symbols.

- A special "initial" symbol **S** (S for Sentence);
- A finite set of symbols stand for 'words' of language called terminal;
  [EXAMPLE: a,b,c……+,-…….etc]
- Other symbols stand for 'phrases' and are known as non terminals.
  [EXAMPLE: A,B,C……]

EXAMPLE of GRAMMAR

Take an example of 0^n1^n:

Here grammar can be represented by following productions :

- S→0S1
- S→ε

Where S is the only variable. The terminals are 0 and 1. there are two productions.

For Example we can generate any string like 0011 using grammar:

S→0S1 [By Using Production S→0S1]
S →0(0S1)1 [By Using Production S→0S1]
S →00(ε)11 [By Using Production S→ε]

Finally we get 0011

Types of Grammars

Depending on the rewriting rules we can characterize the grammars in the following four types:

1. type 0 grammars with no restriction on rewriting rules;
2. type 1 grammars have the rules of the form

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

where A is a nonterminal, α, β, γ are strings of terminals and nonterminals, and γ is non empty.

3. type 2 grammars have the rules of the form

$$A \rightarrow \gamma$$

where A is a nonterminal, and γ is a string (potentially empty) of terminals and nonterminals.

4. type 3 grammars have the rules of the form

$$A \rightarrow aB \text{ or } A \rightarrow a$$

where A, B are nonterminals, and a is a string (potentially empty) of terminals.

Types of Grammars (Continued)

Depending on the rewriting rules we can characterize the grammars in the following four types:

1. Unrestricted grammars with no restriction on rewriting rules;
2. Context-sensitive grammars have the rules of the form

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

where A is a nonterminal, α, β, γ are strings of terminals and nonterminals, and γ is non empty.

3. Context-free grammars have the rules of the form

$$A \rightarrow \gamma$$

where A is a nonterminal, and γ is a string (potentially empty) of terminals and nonterminals.

4. Regular grammars have the rules of the form

$$A \rightarrow aB \text{ or } A \rightarrow a$$

where A, B are nonterminals, and a is a string (potentially empty) of terminals.

Context-Free Languages: Syntax

Definition (Context-Free Grammar)

A context-free grammar is a tuple $G = (V, T, P, S)$ where

- V is a finite set of variables (nonterminals, nonterminals vocabulary);
- T is a finite set of terminals (letters);
- $P \subseteq V \times (V \cup T)^*$ is a finite set of rewriting rules called productions,
  - We write $A \to \beta$ if $(A, \beta) \in P$;
- $S \in V$ is a distinguished start or "sentence" symbol.

Context-Free Languages: Example

Example: G: $\mathbf{0^n 1^n}$ = (V,T,P,S), where

- V = {S};
- T = {0,1};
- P is defined as:
  - S → ε
  - S → 0S1

Generate string 0011 with above grammar

- S → 0S1
- S → 00S11 [S → 0S1]
- S → 0011 [S →ε]

Examples of CFG

Q1: Write CFG which generates palindrome for binary string.

Palindrome is a string which will be same while reading from LHS or RHS example 101, 010…..

- S → **0S0/1S1**
- S → **0/1/ε**

Generate 0101010

- S → **0S0** [S → **0S0**]
- S → **01S10** [S → **1S1**]
- S → **010S010** [S → **0S0**]
- S → **0101010**[S → **1**]

Examples of CFG

Q2: Write CFG for regular expression r = 0*1(0+1)*

- r = 0*1(0+1)*
- 0* 1 (0+1)*
- A B

- S →**A1B**
- **A →0A/ε**
- B →**0B/1B/ε**

Examples of CFG (CO1, CO2)

Q3: Write CFG for regular expression r = (a+b)*aa(a+b)*

- **S →AaaA**
- **A →aA/ε**
- **A →bA/ε**

Q4: Set of al strings of length 2

- R=(a+b)(a+b)
- S →**AA**
- **A →a/b**

Examples of CFG

Q5: Set of all strings of L=(a+b)*

- S →aS/bS/ε

Q6: Set of all strings of length at least 2

- R= (a+b)(a+b)(a+b)*
- S →AAB
- A →a/b
- B →aB/bB/ε

Q7: Set of all strings of length at most 2

- R=(a+b+ε)(a+b+ε)
- S →AA
- A →a/b/ε

Examples of CFG

Q8: For R= a(a+b)*b

- S →aAb
- A →aA/bA/ε

Left Most ad Right Most derivations

- A string can be derived in many ways . But we restrict ourselves to :
    1. Rightmost Derivations
    2. Leftmost Derivations

- In leftmost derivations the left most variable of **β** from **(α→β)** , is picked for expansion.

- In rightmost derivations the right most variable of **β** from **(α→β)** , is picked for expansion.

Left Most ad Right Most derivations [Example]

For the grammar given below :

S →A1B

A →0A|ε

B →0B|1B|ε

Give Leftmost derivation and rightmost derivation of string 1001.

Solution

Leftmost derivation of 1001:

S →A1B

S →1B [A →ε]

→10B [B →0B]

→100B [B →0B]

→1001B [B →1B]

→1001 [B →ε]

Left Most ad Right Most derivations [Example] (Continued)

Rightmost derivation of 1001:

S →A1B

S →A10B [B →0B]

→A100B [B →0B]

→A1001B [B →1B]

→A1001 [A →ε]

→1001 [B →ε]

Parse Tree (CO1, CO2)

Recap: Till now we have learned about basic definition of regular grammar and its types.

Prerequisite: Basic knowledge about Finite Automata and Trees.

Objective: To understand the design of parse tree.

Parse Tree (CO1, CO2)

Objective: To understand the design of parse tree

A set of derivations applied to generate a word can be represented using a tree. Such a tree is known as a parse tree. A parse tree representation gives us a better understanding of:

- Recursion
- Grouping of symbols

A parse tree is constructed with the following conditions:

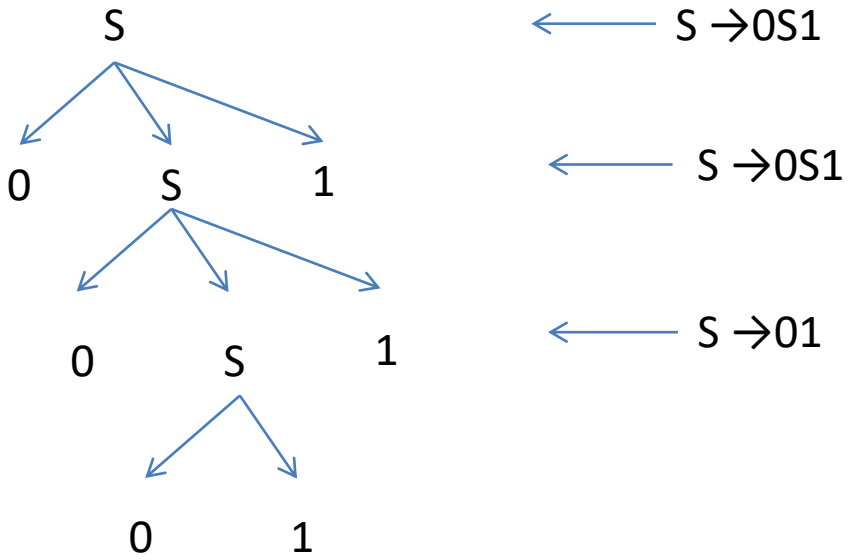
- Root of the tree is represented by start symbol.
- Each interior mode is represented by a variable belonging to V.
- Each leaf mode is represented by a terminal or ε.

Parse Tree [Example]

For a grammar given below S →0S1|01

- Give Parse tree derivation of 000111

- Derivation using parse tree:



Ambiguous Grammar (CO1, CO2)

Recap: We have studied about Grammar and its type and Parse Tree.

Prerequisite: Understanding of Grammar.

Objective: To understand the ambiguity of grammar.

Ambiguous Grammar (CO1, CO2)

Objective: To understand the ambiguity of grammar.

Context **F**ree **G**rammars(CFGs) are classified based on:

- Number of Derivation trees
- Number of strings

Depending on Number of Derivation trees, CFGs are sub-divided into 2 types:

- Ambiguous grammars
- Unambiguous grammars

Ambiguous grammar:

A CFG is said to ambiguous if there exists more than one derivation tree for the given input string i.e., more than one **L**eft **M**ost **D**erivation **T**ree (LMDT) or **R**ight **M**ost **D**erivation **T**ree (RMDT).

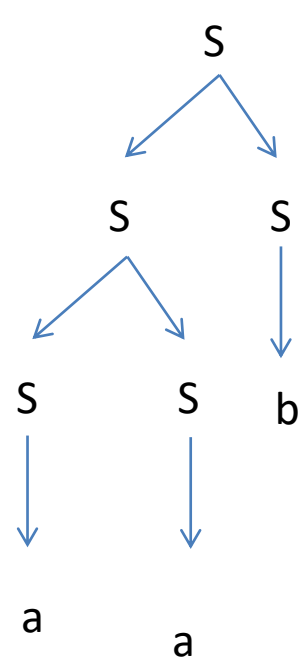
Definition:

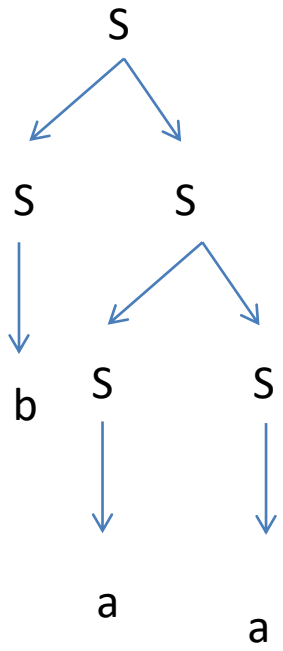
G = (V,T,P,S) is a CFG is said to be ambiguous if and only if there exist a string in T* that has more than on parse tree.

where V is a finite set of variables.
T is a finite set of terminals.
P is a finite set of productions of the form, A -> α, where A is a variable and α ∈ (V ∪ T)* S is a designated variable called the start symbol.

Ambiguous grammar Example:

- **S →SS|a|b is ambiguous or not?**
- Two parse trees can be generated for the string ‘aab’
-





Regular Grammar (CO1, CO2)

Recap: We have studied about Grammar and its type, Parse Tree and Ambiguity of grammar.

Prerequisite: Understanding of Grammar.

Objective: To understand the definition of regular grammar.

Regular Grammar (CO1, CO2)

Objective: To understand the definition of regular grammar.

The language accepted by finite automata can be described using a set of productions known as regular grammar. The productions of a regular grammar are of the following form:

- $A \rightarrow a$
- $A \rightarrow aB$
- $A \rightarrow Ba$
- $A \rightarrow \varepsilon$

Where $a \in T$ and $A,B \in V$.

- A language generated by a regular grammar is known as regular language.

Regular Grammar(Continued)

A regular grammar could be written in two forms

- Right Linear Form
- Left Linear Form

Right Linear Form:

A right Linear Form has the productions of form:

- $A \rightarrow a$
- $A \rightarrow aB$
- $A \rightarrow \varepsilon$

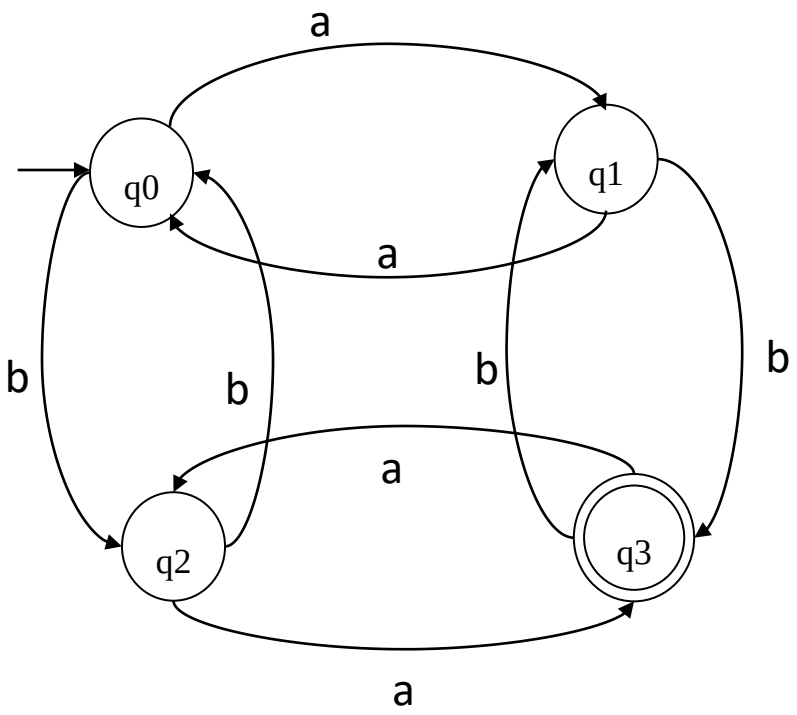
Left Linear Form:

A left Linear Form has the productions of the form:

- $A \rightarrow a$
- $A \rightarrow Ba$
- $A \rightarrow \varepsilon$

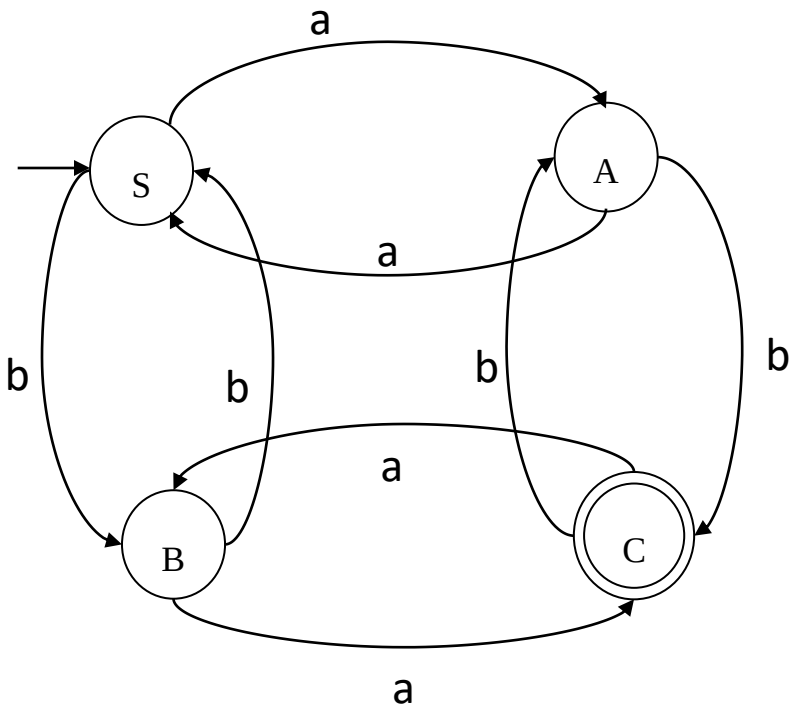
DFA to Right Linear Regular Grammar

- Given a DFA:



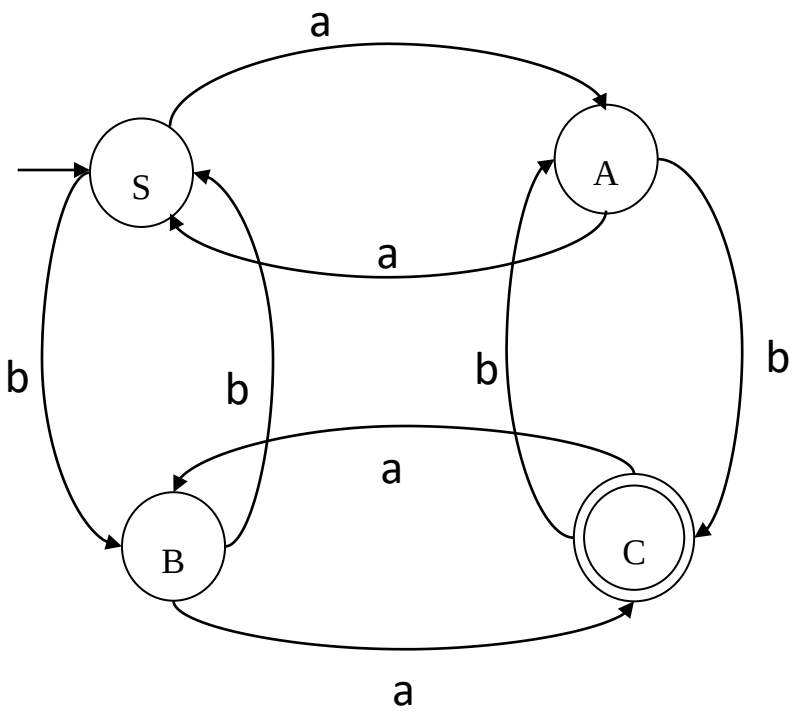
DFA to Right Linear Regular Grammar (Continued)

- **Step 1:** Rename the states: for q0 which is start state make it as S. rest as a non terminals like A,B,C…



DFA to Right Linear Regular Grammar (Continued)

- Step 2: Set of productions are given by:



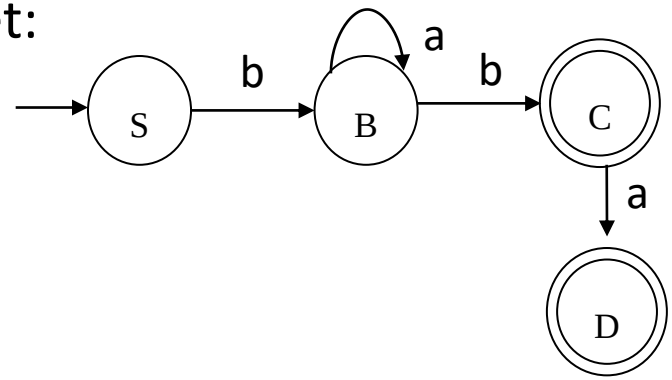
- S → aA|bB
- A → aS|bC|b
- B → bs|aC|a
- C → aB|bA

Right Linear Grammar to DFA

Convert the given Right Linear Grammar to DFA :

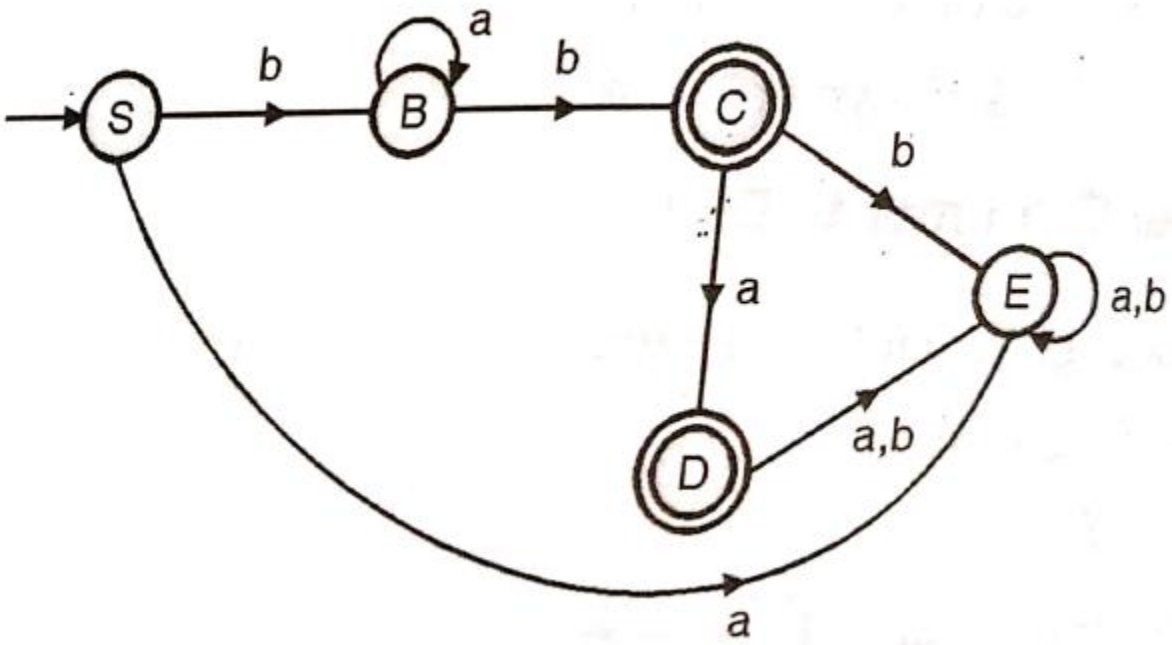
- S →bB
- B →bC|b
- B →aB
- C →a

STEP 1: Adding transactions corresponding to every production ,we get:



Right Linear Grammar to DFA (Continued)

- **STEP 2:** Adding a state E to handle Φ transitions , final DFA will be:



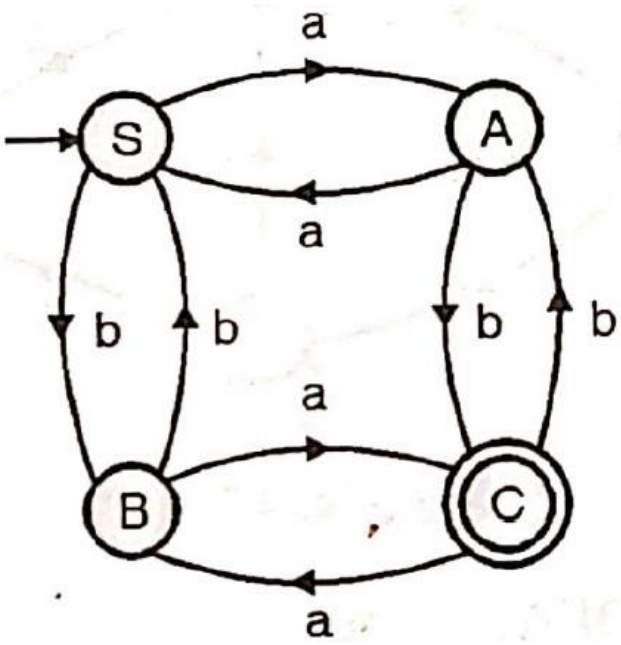
DFA to Left Linear Grammar

Following steps are required to write a left linear grammar corresponding to a DFA.

- Interchange starting state and the final state.
- Reverse the direction of all the transactions.
- Write the grammar from the transaction graph in left linear form.

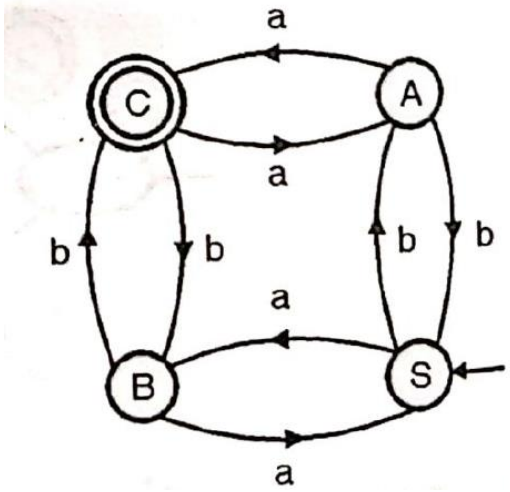
DFA to Left Linear Grammar

- Following DFA is Given:



DFA to Left Linear Grammar(Continued)

- Interchange the starting state and the final state and reverse the direction of transactions, we get the transaction graph as following:



- Writing an equivalent left linear grammar, we get:
- S →Ba|Ab, A →Sb|Ca|a
- B →Sa|Cb|b, C →Bb|Aa

Left Linear Grammar to DFA

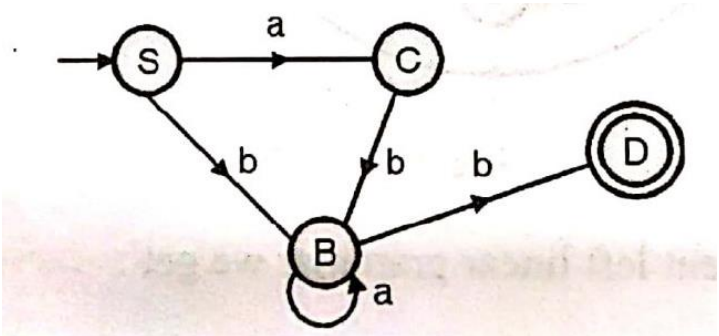
Every left linear grammar can be represented using an equivalent DFA. Following steps are required to draw a DFA for a given left linear grammar.

- Draw a transaction graph from the given left linear grammar.
- Reverse the direction of all the transactions.
- Interchange starting state and the final state.
- Carry out conversion from FA to DFA.

Left Linear Grammar to DFA (Example)

- Following Linear grammar is given:
- S →Ca|Bb
- C →Bb
- B →Ba|b

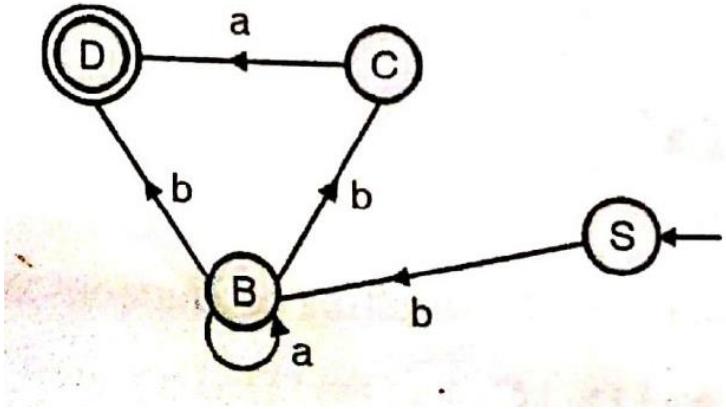
- Step 1: Draw a transaction graph from the given left linear grammar:



- D is an accepting state. It is added for the production B → b

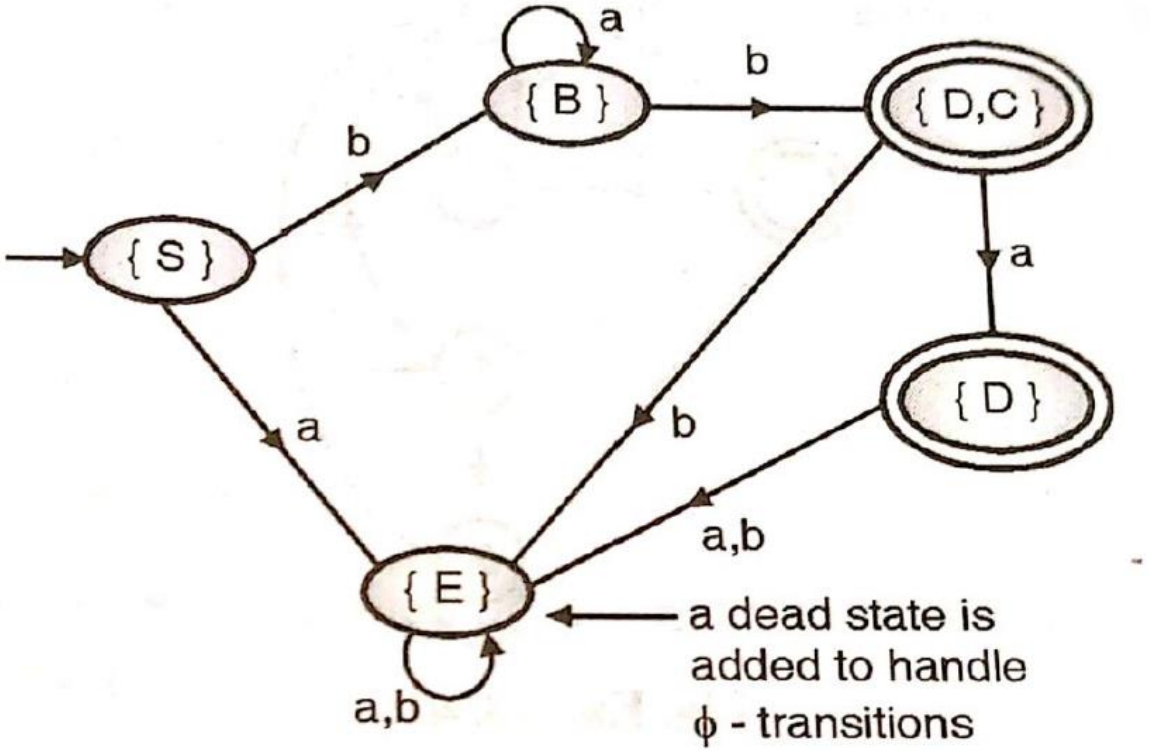
Left Linear Grammar to DFA (Example)

- **Step 2:** Reverse the direction of transaction and interchange starting state and the final state.



Left Linear Grammar to DFA (Example)

- Conversion from FA to DFA:

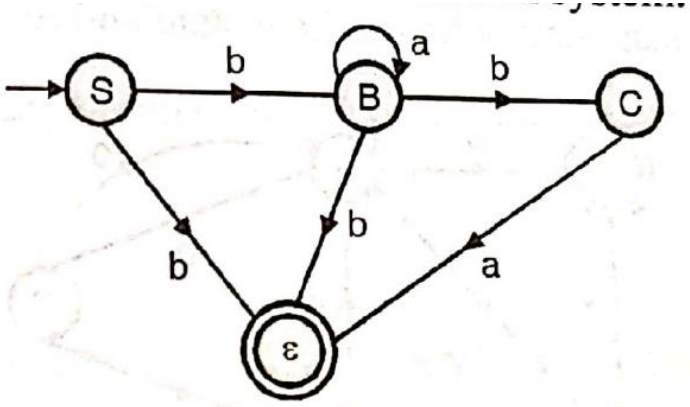


Right Linear Grammar to Left Linear Grammar (CO1, CO2)

- Following Right Linear grammar is given:

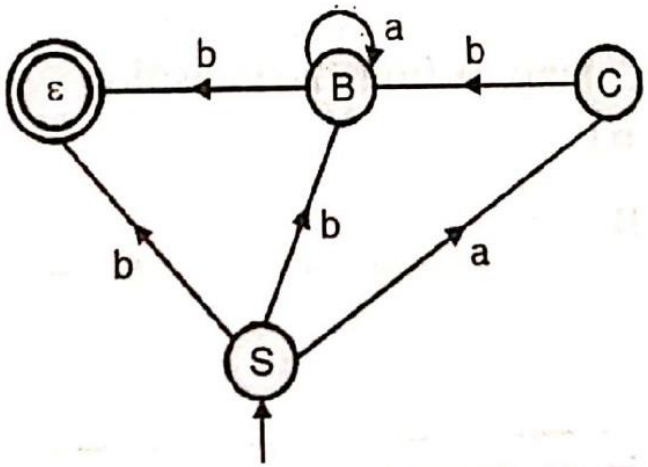
$S \rightarrow bB \mid b$
$B \rightarrow bC$
$B \rightarrow aB$
$C \rightarrow a$
$B \rightarrow b$

- **Step 1:** Conversion of Right Linear grammar to transaction system.
-



Right Linear Grammar to Left Linear Grammar (Continued)

- Step 2: Interchange the start state with the final state and reversing direction of transitions, we get:



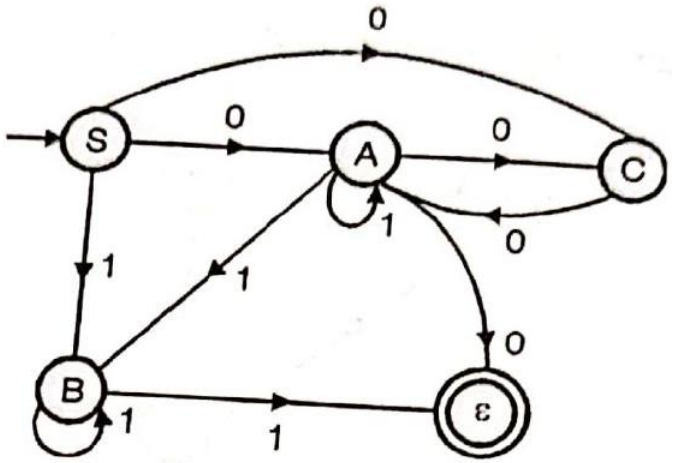
- Write the left linear grammar from the transaction diagram:
- S →b|Bb|Ca
- B →Ba|b
- C →Bb

Left Linear Grammar to Right Linear Grammar

A left Linear Grammar is given:

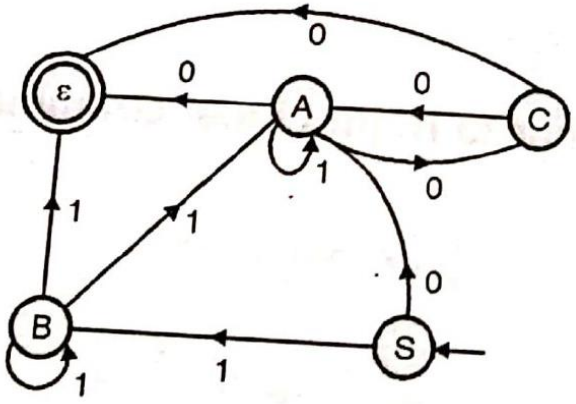
- S →C0|A0|B1
- A →A1|C0|B1|0
- B →B1|1
- C →A0

Step 1: Transition system for the left linear grammar if constructed:



Left Linear Grammar to Right Linear Grammar (Continued)

- **Step 2:** Interchange the start state and the final state and changing direction of all transactions:



- Step 3: Following grammar we get:
- S →1B|0A
- A →0C|1A|0
- B →1B|1A|1
- C →0C|0

Removal of Unit Productions

- Any production rule in the form A → B where A, B ∈ Non-terminal is called **unit production.**

Removal Procedure –

- **Step 1** – To remove **A → B**, add production **A → x** to the grammar rule whenever **B → x** occurs in the grammar. [x ∈ Terminal, x can be Null]
- **Step 2** – Delete **A → B** from the grammar.
- **Step 3** – Repeat from step 1 until all unit productions are removed.

Removal of Unit Productions

Problem

Remove unit production from the following −

S → XY,

X → a,

Y → Z | b,

Z → M,

M → N,

N → a

Solution −

There are 3 unit productions in the grammar −

- Y → Z, Z → M, and M → N

At first, we will remove M → N.

- As N → a, we add M → a, and M → N is removed.

The production set becomes

- S → XY, X → a, Y → Z | b, Z → M, M → a, N → a

Removal of Unit Productions (Continued)

Now we will remove Z → M.

- As M → a, we add Z→ a, and Z → M is removed.

The production set becomes

- S → XY, X → a, Y → Z | b, Z → a, M → a, N → a

Now we will remove Y → Z.

- As Z → a, we add Y→ a, and Y → Z is removed.

The production set becomes

- S → XY, X → a, Y → a | b, Z → a, M → a, N → a
- Now Z, M, and N are unreachable, hence we can remove those.

The final CFG is unit production free −

- S → XY, X → a, Y → a | b

Removal of ε Productions

Given Grammar:

- S→aA
- A →b|ε

Solution:

- Substitute ε on production S→aA
- S→a[ε]
- S→a

- Then add this production into main one:
- S→aA|a
- A →b

Chomsky Normal Form (CO1, CO2)

Recap: Till now we have studied about grammar, Context Free Grammar and Regular Grammar.

Prerequisite: Basic knowledge of CFG and Regular Grammar.

Objective: To understand the definition on Chomsky Normal Form.

Chomsky Normal Form (CO1, CO2)

Objective: To understand the definition on Chomsky Normal Form.

A CFG is in Chomsky Normal Form if the Productions are in the following forms −

- $A \rightarrow a$
- $A \rightarrow BC$
- $S \rightarrow \varepsilon$

where A, B, and C are non-terminals and **a** is terminal.

Algorithm to Convert into Chomsky Normal Form

Step 1 – If the start symbol **S** occurs on some right side, create a new start symbol **S'** and a new production **S'→ S**.

Step 2 – Remove Null productions. (Using the Null production removal algorithm discussed earlier)

Step 3 – Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

Step 4 – Replace each production **A → B_1…B_n** where **n > 2** with **A → B_1C** where **C → B_2 …B_n**. Repeat this step for all productions having two or more symbols in the right side.

Step 5 – If the right side of any production is in the form **A → aB** where a is a terminal and **A, B** are non-terminal, then the production is replaced by **A → XB** and **X → a**. Repeat this step for every production which is in the form **A → aB**.

Chomsky Normal Form(Continued)

Problem

- Convert the following CFG into CNF
- S → ASA | aB, A → B | S, B → b | ε

Solution

(1) Since **S** appears in R.H.S, we add a new state $\mathbf{S_0}$ and $\mathbf{S_0 \rightarrow S}$ is added to the production set and it becomes −

$S_0 \rightarrow S$, S→ ASA | aB, A → B | S, B → b | ∈

Chomsky Normal Form(Continued)

(2) Now we will remove the null productions −

- $B \rightarrow \in$ and $A \rightarrow \in$

After removing $B \rightarrow \varepsilon$, the production set becomes −

- $S_0 \rightarrow S$, $S \rightarrow ASA \mid aB \mid a$, $A \rightarrow B \mid S \mid \in$, $B \rightarrow b$

After removing $A \rightarrow \in$, the production set becomes −

- $S_0 \rightarrow S$, $S \rightarrow ASA \mid aB \mid a \mid AS \mid SA \mid S$, $A \rightarrow B \mid S$, $B \rightarrow b$

Chomsky Normal Form(Continued)

(3) Now we will remove the unit productions.

After removing S → S, the production set becomes −

- S_0→S, S→ ASA | aB | a | AS | SA, A → B | S, B → b

After removing S_0→ S, the production set becomes −

- S_0→ ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
- A → B | S, B → b

After removing A→ B, the production set becomes −

- S_0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
- A → S | b
- B → b

After removing A→ S, the production set becomes −

- S_0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA
- A → b |ASA | aB | a | AS | SA, B → b

Chomsky Normal Form(Continued)

(4) Now we will find out more than two variables in the R.H.S

Here, $S_0 \rightarrow$ ASA, S $\rightarrow$ ASA, A $\rightarrow$ ASA violates two Non-terminals in R.H.S.

Hence we will apply step 4 and step 5 to get the following final production set which is in CNF –

- $S_0 \rightarrow$ AX | aB | a | AS | SA
- S $\rightarrow$ AX | aB | a | AS | SA
- A $\rightarrow$ b |AX | aB | a | AS | SA
- B $\rightarrow$ b
- X $\rightarrow$ SA

Chomsky Normal Form(Continued)

(5) We have to change the productions $S_0 \rightarrow aB$, $S \rightarrow aB$, $A \rightarrow aB$

And the final production set becomes −

- $S_0 \rightarrow AX \mid YB \mid a \mid AS \mid SA$
- $S \rightarrow AX \mid YB \mid a \mid AS \mid SA$
- $A \rightarrow b\ A \rightarrow b \mid AX \mid YB \mid a \mid AS \mid SA$
- $B \rightarrow b$
- $X \rightarrow SA$
- $Y \rightarrow a$

Greibach Normal Form (CO1, CO2)

Recap: Till now we have studied about grammar, Context Free Grammar and Regular Grammar.

Prerequisite: Basic knowledge of CFG and Regular Grammar.

Objective: To understand the definition on Greibach Normal Form.

Greibach Normal Form (CO1, CO2)

Objective: To understand the definition on Greibach Normal Form.

A CFG is in Greibach Normal Form if the Productions are in the following forms −

- $A \rightarrow b$
- $A \rightarrow bD_1 \ldots D_n$
- $S \rightarrow \varepsilon$

where A, D_1,....,D_n are non-terminals and b is a terminal.

Algorithm to Convert a CFG into Greibach Normal Form

Step 1 – If the start symbol **S** occurs on some right side, create a new start symbol **S'** and a new production **S' → S**.

Step 2 – Remove Null productions. (Using the Null production removal algorithm discussed earlier)

Step 3 – Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

Step 4 – Remove all direct and indirect left-recursion.

Step 5 – Do proper substitutions of productions to convert it into the proper form of GNF.

Greibach Normal Form(Continued)

Question:

Convert the following CFG into CNF

- S → XY | Xn | p
- X → mX | m
- Y → Xn | o

Greibach Normal Form(Continued)

Solution

Here, **S** does not appear on the right side of any production and there are no unit or null productions in the production rule set. So, we can skip Step 1 to Step 3.

Now after replacing

X in S → XY | Xo | p

with

- mX | m

we obtain

- S → mXY | mY | mXo | mo | p.

And after replacing

- X in Y → X_n | o

with the right side of

- X → mX | m

we obtain

- Y → mXn | mn | o.

Greibach Normal Form(Continued)

Two new productions O → o and P → p are added to the production set and then we came to the final GNF as the following −

- S → mXY | mY | mXC | mC | p
- X → mX | m
- Y → mXD | mD | o
- O → o
- P → p

Pumping Lemma for CFL

Pumping Lemma for Context-free Languages (CFL) Pumping Lemma for CFL states that for any Context Free Language L, it is possible to find two substrings that can be ‘pumped’ any number of times and still be in the same language.

For any language L, we break its strings into five parts and pump second and fourth substring. Pumping Lemma, here also, is used as a tool to prove that a language is not CFL. Because, if any one string does not satisfy its conditions, then the language is not CFL. Thus, if L is a CFL, there exists an integer n, such that for all $x \in L$ with $|x| \geq n$, there exists u, v, w, x, $y \in \Sigma^*$, such that $x = uvwxy$, and (1) $|vwx| \leq n$ (2) $|vx| \geq 1$ (3) for all $i \geq 0$: $uviwxiy \in L$

Pumping Lemma for CFL (Continued)

- For above example, 0n1n is CFL, as any string can be the result of pumping at two places, one for 0 and other for 1.
- Let us prove, L012 = {0n1n2n | n ≥ 0} is not Context-free.
- Let us assume that L is Context-free, then by Pumping Lemma, the above given rules follow. Now, let x ∈ L and |x| ≥ n.
- So, by Pumping Lemma, there exists u, v, w, x, y such that (1) – (3) hold. We show that for all u, v, w, x, y (1) – (3) do not hold. If (1) and (2) hold then x = 0n1n2n = uvwxy with |vwx| ≤ n and |vx| ≥ 1. (1) tells us that vwx does not contain both 0 and 2. Thus, either vwx has no 0's, or vwx has no 2's. Thus, we have two cases to consider. Suppose vwx has no 0's. By (2), vx contains a 1 or a 2. Thus uwy has 'n' 0's and uwy either has less than 'n' 1's or has less than 'n' 2's. But (3) tells us that uwy = uv0wx0y ∈ L. So, uwy has an equal number of 0's, 1's and 2's gives us a contradiction. The case where vwx has no 2's is similar and also gives us a contradiction. Thus L is not context-free.

Properties of Context-Free Languages

Properties of Context-Free Languages

- Decision Properties
- Closure Properties

Properties of Context-Free Languages (Continued)

- As usual, when we talk about “a CFL” we really mean “a representation for the CFL, e.g., a CFG or a PDA accepting by final state or empty stack.
- There are algorithms to decide if:
  1. String w is in CFL L.
  2. CFL L is empty.
  3. CFL L is infinite.

Properties of Context-Free Languages (Continued)

- We already did this.
- We learned to eliminate variables that generate no terminal string.
- If the start symbol is one of these, then the CFL is empty; otherwise not.

Properties of Context-Free Languages (Continued)

- Want to know if string w is in L(G).
- Assume G is in CNF.
  - Or convert the given grammar to CNF.
  - w = ϵ is a special case, solved by testing if the start symbol is nullable.
- Algorithm (*CYK*) is a good example of dynamic programming and runs in time $O(n^3)$, where n = |w|.

Properties of Context-Free Languages (Continued)

- Let $w = a_1 \ldots a_n$.
- We construct an n-by-n triangular array of sets of variables.
- $X_{ij} = \{\text{variables } A \mid A \Rightarrow^* a_i \ldots a_j\}$.
- Induction on j–i+1.
  - The length of the derived string.
- Finally, ask if S is in X_{1n}.

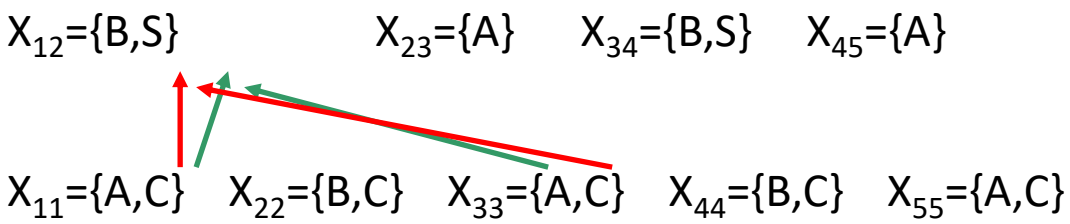
Properties of Context-Free Languages (Continued)

- Basis: $X_{ii} = \{A \mid A \rightarrow a_i$ is a production$\}$.
- Induction: $X_{ij} = \{A \mid$ there is a production $A \rightarrow BC$ and an integer k, with $i \leq k < j$, such that B is in X_{ik} and C is in $X_{k+1,j}$.

Properties of Context-Free Languages (Continued)

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

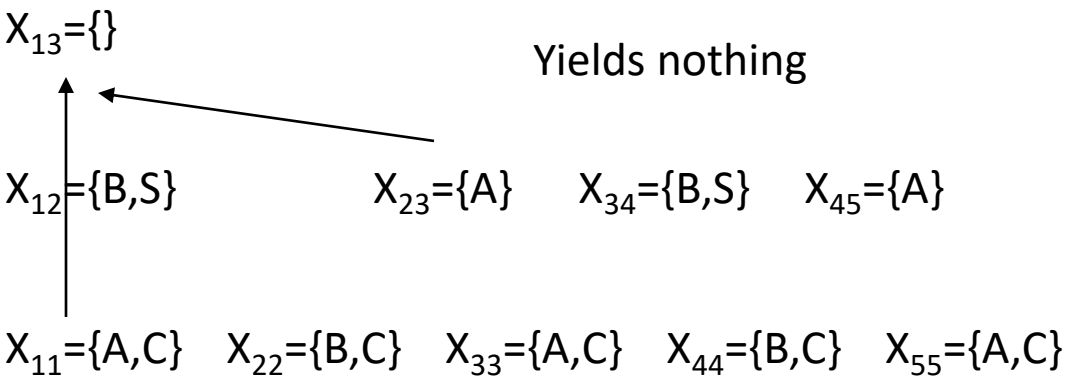
String w = ababa



Properties of Context-Free Languages (Continued)

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa



Properties of Context-Free Languages (Continued)

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa

X_{13}={A} X_{24}={B,S} X_{35}={A}

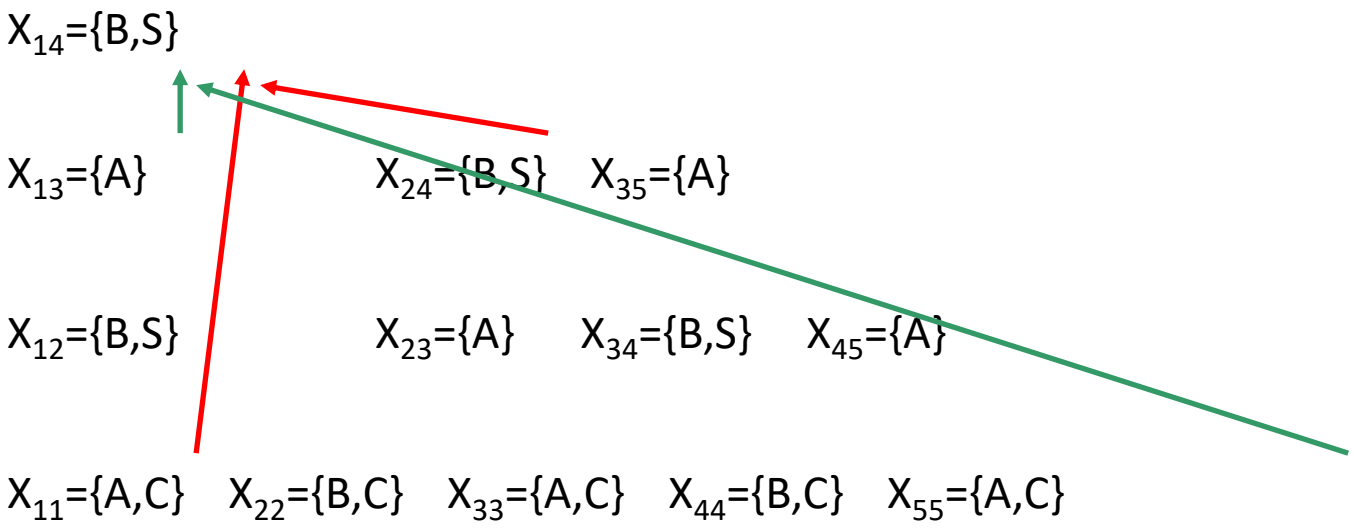
X_{12}={B,S} X_{23}={A} X_{34}={B,S} X_{45}={A}

X_{11}={A,C} X_{22}={B,C} X_{33}={A,C} X_{44}={B,C} X_{55}={A,C}

Properties of Context-Free Languages (Continued)

Grammar: S -> AB, A -> BC | a, B -> AC | b, C -> a | b

String w = ababa



Properties of Context-Free Languages (Continued)

- The idea is essentially the same as for regular languages.
- Use the pumping lemma constant n.
- If there is a string in the language of length between n and 2n-1, then the language is infinite; otherwise not.
- Let's work this out in class.

Properties of Context-Free Languages (Continued)

- CFL's are closed under union, concatenation, and Kleene closure.
- Also, under reversal, homomorphisms and inverse homomorphisms.
- But not under intersection or difference.

Properties of Context-Free Languages (Continued)

- Form a new grammar for L ∪ M by combining all the symbols and productions of G and H.
- Then, add a new start symbol S.
- Add productions S -> S_1 | S_2.

Properties of Context-Free Languages (Continued)

- In the new grammar, all derivations start with S.
- The first step replaces S by either S_1 or S_2.
- In the first case, the result must be a string in L(G) = L, and in the second case a string in L(H) = M.

Properties of Context-Free Languages (Continued)

- Let L and M be CFL's with grammars G and H, respectively.
- Assume G and H have no variables in common.
- Let S_1 and S_2 be the start symbols of G and H.

Properties of Context-Free Languages (Continued)

Closure Under Concatenation – (2)

- Form a new grammar for LM by starting with all symbols and productions of G and H.
- Add a new start symbol S.
- Add production S -> S_1S_2.
- Every derivation from S results in a string in L followed by one in M.

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

My Video :

https://www.youtube.com/feed/my_videos

Youtube/other Video Links

- https://www.youtube.com/watch?v=6b40kKe2SFg
- https://www.youtube.com/watch?v=-aIRqNnUvEg&list=PL85CF9F4A047C7BF7
- https://www.youtube.com/watch?v=Xk-MTgB68rQ

Daily Quiz

1. Let G be the grammar S → aB | bA, A →a | aS | bAA, B → b | bS | aBB. For the string aaabbabbba find
   a. Parse tree
   b. Leftmost derivation
   c. Rightmost derivation
   d. Is the grammar unambiguous?
2. Write the grammar generating the language L = $\{a^n b^m \mid \text{where } n \neq m\}$
3. Define the Chomsky Normal Form (CNF). Convert the grammar given below to its equivalent CNF:

- $S \rightarrow ABA$
- $A \rightarrow 0A \mid \epsilon$
- $B \rightarrow 1B \mid \epsilon$

4. Using pumping lemma for CFLs prove that language is not Context Free.

$$L = 0^{n^2}, n >= 1$$

Daily Quiz (Continued)

5. The following grammar generates the regular expression 0*1(0+1)*

   S→A1B

   A→0A| ε

   B→0B|1B|ε

   Give leftmost and rightmost derivation of the string 00101

6. Give the CFG of the following language:

   0(0+1)*01(0+1)*1

7. Find the CFG of following language

   $L=\{a^i b^j c^k | i=j+k\}$

8. Give CFG for matching parenthesis.

9. Give CFG for all strings with at least two 0s, over an alphabet {0,1}.

10. Write an equivalent right recursive grammar for the given left recursive grammar: S→S10|0

Weekly Assignment

1. Write Context Free Grammars for Following: **[C03]**

- L= $\{0^n1^{n+m}2^{m+p}3^p$ where n, m, p>=1$\}$
- L= {w: w consists of all strings containing a and b which are palindromes}
- L= {w: w consists of all strings containing a & b which are even length palindromes}
- L= $\{ a^nb^m$: n≠ m$\}$
- L= $\{a^na^mb^k$:n=m or m<=k $\}$
- L= $\{a^nb^mc^k$:n=m or m≠k $\}$
- L= $\{a^nb^mc^k$:k = n + 2m $\}$ where n, m>0
- L= $\{a^nb^mc^k$: k = |n – m | $\}$ where n, m, k>0
- L= $\{a^nww^Rb^n$: wϵ (a/b)* , n>=1$\}$
- L= { w: w ϵ (a/b)*, which generates string of balanced parenthesis

Weekly Assignment

- Write a CFG for language L(G) containing strings with at least one a. **[C03]**
- Write a CFG for language L(G) containing strings with at least 3 a. **[C03]**
- Consider G = ({A, B}, {a, b, c}, A, P) with productions **[C03]**

  A→a|aaA|abBc B→abbA|b

  Convert the grammar into CNF.
- Show that two grammars **[C03]**
- S→abAB|ba, A→aaa,B→aA|bb and S→abAaA|abAbb|ba, A→aaa are equivalent.

Weekly Assignment

- Eliminate the useless productions from the grammar **[C03]**

  S→aS|AB, A→bA, B→AA

  What language this grammar generates.

- Find a Context Free Grammar without λ Productions equivalent to the grammar defined by:

  S→AaB|aaB, A→ λ, B→bbA| λ **[C03]**

- Convert the following grammar into CNF **[C03]**

  S→AACD, A→aAb| λ, C→aC| λ, D→aDa|bDb| λ

- Remove all Unit productions, all useless productions, and all λ productions from the grammar

  S→aA|aBB, A→aaA| λ, B→bB|bbC, C→B **[C03]**

- Find a Context Free Grammar without λ Productions equivalent to the grammar defined by:

  S→AaB|aaB, A→ λ, B→bbA| λ **[C03]**

- Use pumping lemma to prove language L = $a^n b^n c^n$ is not context free. **[C03]**

MCQ s

1. The alphabet over which the language is defined is represented in CFG as:
   a. non-terminals
   b. terminals
   c. star symbols
   d. Productions

2. In a CFG left hand side production is

a. a single non terminal
b. a single terminal
c. two non-terminals
d. combination of terminals and non-terminals

MCQ s(Continued)

3. Which one is not a CFG?

a. S→a
b. aA→b
c. S→aA
d. S→Sb

4. Following productions represent which language: A→a, A→b

a. {ab}
b. {ba}
c. {a,b}
d. {a}

MCQ s(Continued)

5. Which one is a unit production?

a. A→B

b. A→aB

c. A→Bb

d. A→a

6. the production S→ABC can be converted to S→AX by adding

a. S→AB

b. X→BC

c. X→AB

d. X→ABC

7. In parse tree the interior nodes are:

a. starting symbol

b. terminal

c. non terminal except start symbol

d. null symbol

MCQ s(Continued)

8. Terminals can be only in ……. in case of parse tree:

a. root

b. leaves

c. nodes

d. none of these

9. In case of productions A→aB, A→a the state B is in :

a. initial

b. final

c. intermediate

d. Dead

10. in case of a CFG in regular form the productions should be in which form:

a. A→aB|a

b. A→Ba

c. A→a

d. A→CV

Expected Questions for University Exam

1. Let G = ({ S,C},{a,b},P,S) where P consistes of S→aCa, C→aCa|b. Find L(G).
2. Find L(G), where G = ({ S},{0,1},{S→0S1, S→ε},S)
3. Find out the context free language

- S→aSb|aAb
- A→bAa
- A→ba

4. Give the CFG of the following language:

   0(0+1)*01(0+1)*1

5. Give CFG for matching parenthesis.

Old Question Papers

https://drive.google.com/drive/folders/19Eia3VHCl3627foiH6V_j-p4X9ZkyyC7?usp=sharing

Summary

- There are basically two types of derivations of strings from the grammar, namely, top down derivations and bottom up derivations.
- Parse tree can be constructed for each derivations.
- Ambiguous grammars and normal forms for the grammars were also discussed in this unit.

References

1. Aho, Hopcroft and Ullman, *The Design and Analysis of Computer Algorithms,* Addison Wesley.
2. Aho, Sethi, Ullman, *Compilers Principles, Techniques and Tools,* Pearson Education, 2003.
3. Hopcroft and Ullman, *Introduction to Automata Theory, Languages and Computation,* Addison Wesley.
4. Kohavi, ZVI, *Switching And Finite Automata Theory*, Tata McGraw-Hill, 2006.
5. Lewis and Papadimitriou, *Elements of the Theory of Computation,* Prentice-Hall.
6. Martin, *Introduction to Languages and the Theory of Computation,* McGraw-Hill, 2nd edition,1996.
7. Mishra, KLP, Chandrasekaran, N. *Theory of Computer Science,* (Automata, Languages and Computation) PHI, 2002.

Thank You