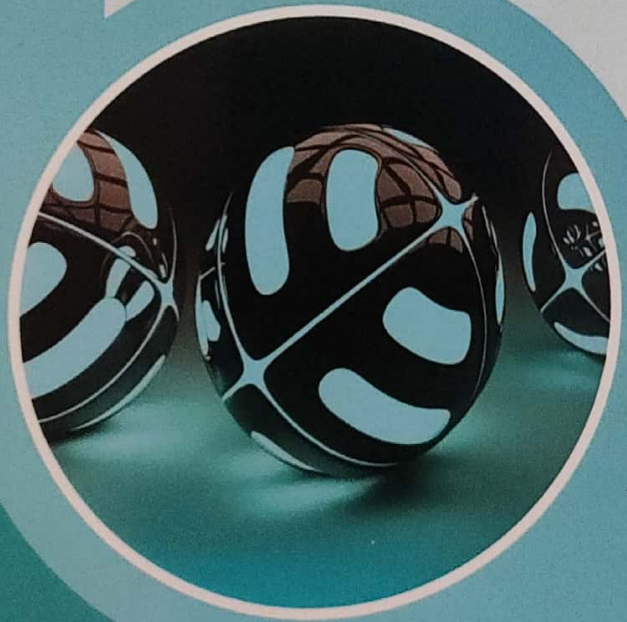
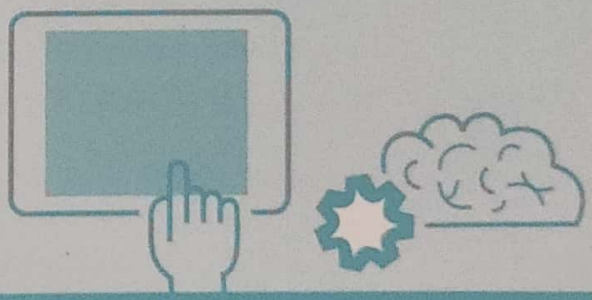
GATE
PSUs

MADE EASY
Publications

POSTAL
STUDY PACKAGE

COMPUTER SCIENCE & IT





2020

THEORY
BOOK

Theory of Computation

Well illustrated theory with solved examples

Computer Science & IT

Theory of Computation

Comprehensive Theory

with Solved Examples and Practice Questions

MADE EASY Publications

Contents

Theory of Computation

Theory of Computation

GOAL OF THE SUBJECT

Theory of Computation deals with automata theory and formal languages. It is the study of "*Abstract Model of Computation*". This subject helps to solve various problems using the following modes.

- Finite Automata
- Push Down Automata
- Linear Bound Automata
- Turing Machine

It is very important for every computer programmer to know what are the problems that can be solved and what cannot be solved. This subject explains the capabilities and limitations of computation.

INTRODUCTION

To understand the formal model of computation, this book provides you rigorous presentation of concepts, models, examples accompanied by practice problems and student assignment.

This book is organized with all models that makes it easier to read, understand and acquire quick interpretation of all models. It uses definitions and properties of mathematical models to explain the capabilities and limitations of Problems/Computation/Programs/Languages.

The following information provides a brief description of the chapters in this book.

Chapter 1 (Grammars, Languages & Automata): This chapter deals with all notations, and definitions of theory of computation such as Grammar, Equivalence of Language, Chomsky Hierarchy Model and languages with their relationships in the computation model.

Chapter 2 (Regular Languages & Finite Automata): This chapter explains indetail about type-3 formal languages acceptance by Finite Automata with representations like Regular Expressions, Regular Sets, and Regular Grammars. It also covers the Closure Properties and Decision Properties of regular languages with detailed proofs and Moore and Mealy machines.

Chapter 3 (Context free languages & Push Down Automata): This chapter covers type-2 formal languages acceptance by Push Down Automata and representations with equivalences between CFG and CFL, Closure Properties and Decision Properties of context free languages with detailed proofs and identify the CFL and non-CFL languages.

Chapter 4 (REC, RE Languages & Turing Machines, Decidability): This chapter deals with Turing Machine Decidable and Undecidable problems of formal languages and automata theory and Closure Properties and Decision Properties of context free languages with detailed proofs.

■■■■

CHAPTER 01

Grammars, Languages & Automata

1.1 Introduction

1.1.1 Alphabet(Σ)

Definition: An Alphabet is a set of finite non-empty set of symbols.

- $\Sigma = \{a, b\}$ is alphabet with 2 symbols *a* and *b*.
- $\Sigma = \{0, 1\}$ is binary alphabet.
- $\Sigma = \{0, 1, 2, \ldots 9\}$ is decimal alphabet.

1.1.2 String

Definition: A string is any sequence of zero or more symbols over the given alphabet Σ.

"*abb*" is the string over $\Sigma = \{a, b\}$

Empty string (ε or λ): Empty string is a string with zero number of symbols in the sequence. Empty string is also called as "*Null String*".

1.1.3 Operations on Strings

1. **Length of a string:** The number of symbols in the sequence of given string is called "length of a string"
   - Length of string abb = $|abb| = 3$, where $\Sigma = \{a, b\}$
   - $|\varepsilon| = 0$. The length of empty string is zero.
   - $|a| = 1$, $|ab| = 2$
2. **Substring of a string:** A sequence of symbols from any part of the given string over an alphabet is called a "Substring of a string".

   For string abb over $\Sigma = \{a, b\}$. The possible substrings are:
   - Zero length substring: ε
   - One length substrings: a, b
   - Two length substrings: ab, bb
   - Three length substrings: abb

"Proper Substing" is a substring and its length is less than given string length. The string "abb" is not a proper substring of the string "abb" but is a substring of "abb".

3. **Prefix of a string:** A substring with the sequence of beginning symbols of a given string is called a "Prefix".

   For string "abb", the possible prefixes of abb are:
   - ε, (zero length prefix)
   - a (one length prefix)
   - ab (two length prefix)
   - abb (three length prefix)

4. **Suffix of a string:** A substring with the sequence of trailing (ending) symbols of a given string is called a "Suffix".

   For string abb, the possible suffixes are: ε, b, bb, abb.

5. **Proper Prefix of a string:** Proper prefix is a prefix except the given string.

   For string abb, the possible proper prefixes are: ε, a, ab.

6. **Proper Suffix of a string:** Proper suffix is a suffix except the given string.

   For string abb, the possible proper suffixes are: ε, b, bb.

1.1.4 Power of an Alphabet

Consider $\Sigma = \{a, b\}$. The following are powers of an alphabet over input alphabet Σ.

$\Sigma^0 = \{\varepsilon\}$: zero length string

$\Sigma^1 = \{a, b\}$: set of 1-length strings (also called as symbols)

$\Sigma^2 = \{aa, ab, ba, bb\}$: set of 2-length strings

$\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$: set of 3-length strings.

1.1.5 Kleene Star Closure (Σ^*)

- Σ^* is the set of all strings over Σ.
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ...$
- Kleene closure (*) is unary operation.

1.1.6 Positive Closure (Σ^+)

- Σ^+ is the set of strings over Σ except an empty string.
- $\Sigma^+ = \Sigma^* - \{\varepsilon\} = \Sigma^* - \Sigma^0$
- $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

1.1.7 Language

Definition: A set of strings over the given alphabet Σ is called a "language".

Let $\Sigma = \{a, b\}$. Then {ab, aab} is a language.

Language may contain finite or infinite number of strings. So language is two types: (a) Finite language, and (b) Infinite Language.

1.1.8 Grammar

Grammar has set of rules to generate the strings of a language.

Grammar is a set of 4 tuples. Grammar G = (V, T, P, S) where V is set of non-terminals, T is set of terminals, P is set of productions or rules and S is start symbol ($S \in V$).

Each rule appears as: $X \rightarrow Y$, where X and Y are any sequence of terminals and non-terminals.

1.2 Chomsky Hierarchy

- All formal languages are divided into four classes by chomsky and this hierarchy known as "Chomsky-Hierarchy".
- Type 3 ⊆ Type 2 ⊆ Type 1 ⊆ Type 0.
- Regular languages ⊆ CFLs ⊆ CSLs ⊆ RELs.

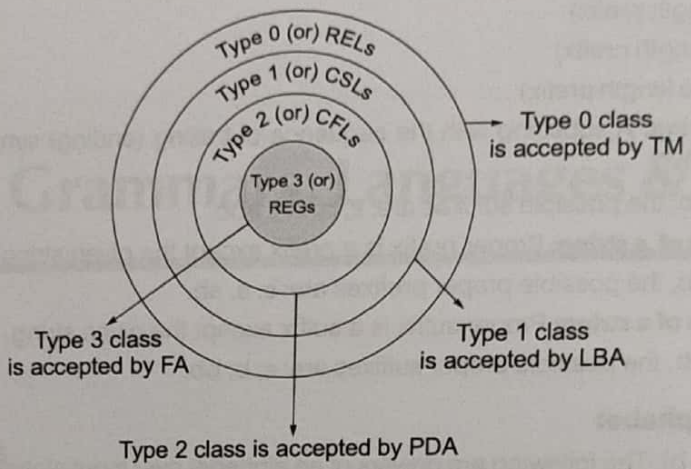


***Figure*-1.1 :** *Chomsky Hierarchy*

1.2.1 Formal Languages

Definition: Formal language is an abstraction of the generalized characteristics of programming languages.

or

Formal language is a set of all strings where each string is restricted over a particular form.

or

Formal language is a set of all strings permitted by the rules of formation.

1.2.2 *Types of Languages*

1. **Regular Language (REG):** A language accepted by a finite automaton is called a "regular language".

   or

   A language generated by regular grammar is called a "regular language".
2. **Deterministic Context Free Language (DCFL):** A language accepted by a deterministic push down automaton is called a DCFL.
3. **Context Free Language (CFL):** A language accepted by a push down automaton (non-deterministic) is called a "CFL".

   or

   A language generated by a context free grammar is called a "CFL".
4. **Context Sensitive Language (CSL):** A language accepted by a linear bound automaton is called a "CSL".

   or

   A language generated by a context sensitive grammar is called a "CSL".
5. **Recursive Language (REC):** A language accepted by the Halting Turing Machine is called a *recursive language* or *REC*.

or

If a language can be enumerated in a lexicographic order (Particular order) by some turing machine, then such a language is called recursive or *REC*.

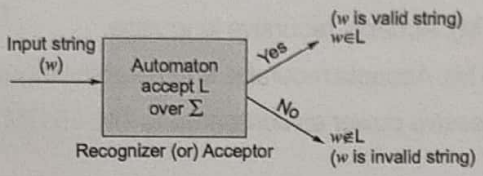
6. **Recursive Enumerable Language (REL):** A language accepted by turing machine is called a *recursive enumerable language* or REL.

or

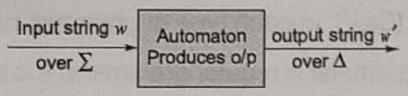
The language enumerated by the turing machine is called as *recursive enumerable* or *REL*.

1.3 Automaton

1. **Automaton Acts as Recognizer or Acceptor:** An automaton is a machine that accepts or recognizes the strings of a language (L) over an input alphabet Σ.



2. **Automaton Acts as Generator or Enumerator or Transducer:** An automaton can also produce the output over any alphabet Δ for the given input alphabet Σ.



1.3.1 Types of Automaton

1. **Finite Automaton (FA):** An automaton that accepts a regular language is called a "FA".

   Types of Finite Automaton:

   (a) Deterministic Finite Automaton (DFA)

   (b) Non-deterministic Finite Automaton (NFA or NDFA)

   (c) NFA with ε-moves (ε-NFA)

   L(DFA) ≅ L(NFA) ≅ L(ε-NFA)

   where L(DFA) is the class of languages accepted by DFA's.

   i.e., all finite automata are having same expressive power.

2. **Push Down Automaton (PDA):** An automaton that accepts a context free language is called a "PDA".

   Types of Push Down Automaton:

   (a) Deterministic PDA (DPDA)

   (b) Non-deterministic PDA (NPDA or PDA)

   L(DPDA) < L(NPDA) i.e., DCFL's ⊂ CFL's

   Class of languages accepted by DPDA's are proper subset of class of languages accepted by NPDAs.

   NPDAs have more expressive power than the DPDAs.

3. **Linear Bound Automaton (LBA):** An automaton that accepts a context sensitive language is called a *linear bound automaton.*

4. **Turing Machine (TM):** An automaton that accepts a recursive enumerable language is called a *turing machine*.
   - TM can accept a recursive enumerable language. (TM acts as recognizer)
   - TM can enumerate a recursive enumerable language. (TM acts as enumerator)
   - Types of turing machine (based on configuration):
     (a) Deterministic Turing Machine (DTM)
     (b) Non-deterministic Turing Machine (NTM)
   - NTM and DTM are having same expressive power; $L(NTM) \equiv L(DTM)$
   - Standard turing machine is a deterministic turing machine.
   - Types of TM (Based on acceptance/enumeration):
     (a) Halting TM (HTM): Accepts recursive language.
     (b) Standard TM (TM): Accepts recursive enumerable language.
   - HTM has less expressive power as compared to TM; $L(HTM) \subset L(TM)$.

1.4 Grammars

Let G be a grammar. G = (V, T, P, S) where V is set of non-terminals, T is set of terminals, P is set of rules or productions and S is a start symbol ($S \in V$). In each type of grammar, rules of P are restricted.

1. **Regular Grammar (RG):** A grammar is regular grammar iff it is either left linear grammar or right linear grammar.
   Left linear grammar: $V \to VT^* \mid T^*$
   Right linear grammar: $V \to T^*V \mid T^*$ (where V is any variable and T is any terminal)
   - Left linear grammar : $S \to Aa, A \to Aa \mid a$
   - Right linear grammar : $S \to aB, B \to bB \mid a$
2. **Context Free Grammar (CFG):** Every rule of CFG is restricted as: $V \to (V \cup T)^*$
   - $\{S \to aSb \mid \varepsilon\}$ is CFG
   - $\{S \to B, A \to aAbA \mid a, B \to AB \mid \varepsilon\}$ is CFG
3. **Context Sensitive Grammar (CSG):** Every rule of CSG is restricted as following.
   $X \to Y$ where $|X| \leq |Y|$, $X \in (V \cup T)^*$ and $Y \in (V \cup T)^*$ and X must contain atleast one variable.
   - If NULL productions are present in the grammar, it may not be a CSG.
   - $\{S \to aAb \mid a,\ aA \to cd\}$ is CSG
   - $\{S \to aAb \mid \varepsilon,\ aA \to cd\}$ is CSG
   - $\{S \to aAb \mid aS,\ aAb \to cdef\}$ is CSG
4. **Recursive Enumerable Grammar (REG):** Every rule of REG is restricted as following.
   $X \to Y$ where $X \in (V \cup T)^*$, $Y \in (V \cup T)^*$; X must contain atleast one variable.
   - REG is also called as unrestricted grammar or phrase-structure grammar.
   - $\{S \to aAb \mid bS,\ aA \to b\}$ is REG

1.5 Equivalence of Languages, Grammars and Automata

1. Regular Language (REG)
   ≅
   Finite Automaton (DFA ≅ NFA ≅ ε-NFA)
   ≅
   Regular Grammar
   ≅
   Left Linear Grammar (LLG)
   ≅
   Right Linear Grammar (RLG)
   ≅
   Regular Expression (RE)
   ≅
   Type-3 formal language
2. Context Free Language (CFL)
   ≅
   Push Down Automata (NPDA or PDA)
   ≅
   Context Free Grammar (CFG)
   ≅
   Type-2 formal language
3. Context Sensitive Grammar (CSG)
   ≅
   Linear Bound Automata (LBA)
   ≅
   Context Sensitive Language (CSL)
   ≅
   Type-1 formal language
4. Recursive Enumerable Grammar or Unrestricted Grammar (UG)
   ≅
   Turing Machine (TM)
   ≅
   Recursive Enumerable Language (REL)
   ≅
   Type-0 formal language

Class	Formal Languages	Grammars	Automata
Type-3	Regular Languages	Regular grammars	Finite Automata
Type-2	Context Free Languages	Context Free Grammars	Push down Automata
Type-1	Context Sensitive Languages	Context Sensitive Grammars	Linear Bound Automata
Type-0	Recursive Enumerable Languages	Recursive enumerable grammars (unrestricted grammars)	Turing Machines

1.6 Expressive Power of Automata

Expressive power of a machine is determined by the set of languages accepted by the particular type of machines.

FA < DPDA < PDA < LBA < TM.

FA is less powerful than any other machine and TM is more powerful than any other machine.

- Type 3 class ⊂ Type 2 class ⊂ Type 1 class ⊂ Type 0 class
- FA ≅ TM with read only tape ≅ TM with unidirectional tape ≅ TM with finite tape ≅ PDA with finite stack
- PDA ≅ FA with stack
- TM ≅ PDA with additional stack ≅ FA with two stacks

1.7 Applications of Automata

- Finite automata can be used in the following cases:
  (a) To design a lexical analysis of a compiler
  (b) To recognize the pattern using regular expressions
  (c) To design the combination and sequential circuits using Moore / Mealey machines.
  (d) Used in text editors
  (e) Used to implement spell checkers
- PDA can be used in the following cases:
  (a) To design the parsing phase of a compiler (syntax analysis)
  (b) To implement stack applications
  (c) To evaluate arithmetic expressions
  (d) To solve the Tower of Hanoi problem
- Linear Bounded Automata can be used in the following cases:
  (a) To construct syntactic parse trees for semantic analysis of the compiler.
  (b) To implement genetic programming.
- Turing Machine can be used in the following cases:
  (a) To solve any recursively enumerable problem
  (b) To understand complexity theory
  (c) To implement neural nets
  (d) To implement artificial Intelligence
  (e) To implement Robotic applications

Summary

- **Alphabet:** It is a set of finite number of symbols.
- **String:** It is a sequence of symbols over given alphabet.
- **Substring:** Any part of the given string is called a substring.
- **Language:** It is a set of strings over given alphabet.
- **Grammar:** It is a set of 4 tuples that contain set of rules to generate the strings of a language.
- **Automaton:** Automaton can accept the language and it may be used to produce the output.
- **Formal Languages:** Type0, Type1, Type2 and Type3 formal languages.
- **Automaton Types:** Finite automata, Push down automata, Linear bound automata and Turing Machines.
- **Grammars:** Regular grammars. CFGs, CSGs, REGs
- Language may be finite set or infinite set.
- Expressive power of a machine is the class or set of languages accepted by the particular type of machines.

$$FA < PDA < LBA < TM$$

- FA is less powerful than any other automata.
- TM is more powerful than any other automata.
- A grammar is regular grammar iff it is left linear grammar or right linear grammar.
- DCFL class is subset of CFL class.
- NFA has same power as of DFA.
- DPDA has less power than NPDA.
- DTM has same power as of NTM.
- Finite automaton does not have infinite memory for comparison.
- Every finite language is regular but converse need not be true.
- Language may be infinite but variables, terminals and production rules are finite in every grammar.
- Let w be a string of length n. Then number of possible substrings including ε is atmost $\frac{n(n+1)}{2}+1$.
- Number of prefixes for the given n-length string = $(n+1)$.
- Number of suffixes for the given n-length string = $(n+1)$.

Student's Assignments 1

Q.1 Finite automata has
(a) Finite Control (b) Read only head
(c) Both (a) and (b) (d) None of these

Q.2 Compared to NFA, DFA has
(a) Less power
(b) More power
(c) Deterministic transition
(d) None of these

Q.3 Finite Automata is used in which of the following?
(a) Pattern matching
(b) Sequential circuit design
(c) Compiler design
(d) All (a), (b) and (c)

Q.4 A finite automata is
(a) Acceptor that accepts a regular language
(b) Transducer that computes some simple functions
(c) Both (a) and (b)
(d) None of these

Q.5 The language accepted by finite automata is
(a) Context-free (b) Regular
(c) Non-regular (d) None of these

Q.6 In computer system, finite automata program can be stored using
(a) Table
(b) Two-dimensional array
(c) Graph
(d) All (a), (b) and (c)

Q.7 Which of the followings is true
(a) All NFAs are DFAs
(b) All DFAs are NFAs
(c) Both (a) and (b)
(d) NFA and DFA have different power

Q.8 Two-way finite automata has
(a) Two tapes
(b) Two heads
(c) Bi-directional head movement
(d) All the above

Q.9 The behavior of a NFA can be simulated by a DFA
(a) Always (b) Sometime
(c) Never (d) Depend on NFA

Q.10 A FA with deterministic transitions capability is known as
(a) NFA (b) DFA
(c) 2DFA (d) NFA with ε-moves

Answer Key:

1. (c)	**2.** (c)	**3.** (d)	**4.** (c)	**5.** (b)
6. (d)	**7.** (b)	**8.** (c)	**9.** (a)	**10.** (b)

Student's Assignments 1 Explanations

1. (c)
Finite automata has finite control, read only head, tape is unbounded but stores finite length string.
∴ Both (a) and (b) are correct.

2. (c)
NFA and DFA both accepts regular language.
NFA and DFA are equivalent.
But NFA is non-deterministic whereas DFA is deterministic.
∴ Option (c) is correct.

3. (d)
FA can be used in pattern matching, sequential circuit design and lexical analysis of compiler design, etc.
∴ Option (d) is correct.

4. (c)
A finite automata may be an acceptor or a transducer.
∴ Option (c) is correct.

5. (b)
FA accepts regular languages
∴ Option (b) is correct.

6. (d)
In computer, FA program can be stored using table, graph, array, etc.
∴ Option (d) is correct.

7. (b)
Every DFA is a special case of NFA, but every NFA need not be a DFA.
∴ Option (b) is correct.

8. (c)
Two-way finite automata has bi-directional head. The head can move in both directions: left or right. This machine is called as 2DFA or 2NFA based on nature of acceptance.
∴ Option (c) is correct.

9. (a)
NFA and DFA are equivalent.
Hence NFA always simulated by a DFA and NFA can be converted to a DFA
∴ Option (a) is correct.

10. (b)
FA with deterministic transitions capability is called DFA.
∴ Option (b) is correct.

Student's Assignments 2

Q.1 A FA with non-deterministic transitions capability is known as _____.
(a) NFA (b) DFA
(c) 2DFA (d) NFA with ε-moves

Q.2 A FA having more than one input tape has _____
(a) More power than one tape FA
(b) Equal power as one tape FA
(c) Less power than one tape FA
(d) None of these

Q.3 A FA having more than one reading head has _____.
(a) More power than one reading head FA
(b) Equal power as one reading head FA
(c) More storage than one reading head FA
(d) Both (b) and (c)

Q.4 Limitation of a FA is _____
(a) Writing on tape
(b) Finite memory
(c) pattern recognition
(d) Both (a) and (b)

Q.5 What is true about finite automata? It is
(a) Acceptor (b) Recognizer
(c) Pattern Matcher (d) All of these

Q.6 Which is false?
(a) In DFA, there is only one transition for every input symbol from any state.
(b) In NFA, there is zero or more transition for an input symbol from any state.
(c) All DFAs are NFAs, but not all NFA are DFA. So, NFA have more power as compare to DFA.
(d) The language accepted by NFA is equivalent to a DFA.

Q.7 A finite automata can
(a) Check the validity of input string
(b) Manipulate the input and calculate the output
(c) Both (a) and (b)
(d) None of these

Q.8 FSM can recognize
(a) Any language
(b) Only regular language
(c) Only context-free
(d) None of these

Q.9 An automaton is a _____ device and a grammar is a _____ device.
(a) generative, cognitive
(b) generative, acceptor
(c) acceptor, cognitive
(d) cognitive, generative

Q.10 Context-free language can be recognized by
(a) finite state automaton
(b) linear bounded automata
(c) push down automata
(d) both (b) and (c) above

Q.11 Can a DFA simulate NFA?
(a) no (b) yes
(c) some time (d) depends on NFA

Q.12 A language L is accepted by a finite automaton if and only if it is
(a) context-free
(b) context-sensitive
(c) recursive
(d) expressible by a right-linear grammar

Q.13 The basic limitation of FSM is that
(a) it can not remember arbitrary information
(b) it sometimes fails to recognize regular language
(c) it sometimes fails to recognize regular language
(d) all of these

Q.14 Regarding the power of recognizing the languages, which of the following statements is false?
(a) The NDFA are equivalent to DFA
(b) NPDA are equivalent to DPDA
(c) NDTMs are equivalent to DTMs
(d) Universal TM is equivalent to standard TM

Q.15 Which of the following is false?
(a) The languages accepted by FAs are regular languages
(b) Every DFA is an NFA
(c) There are some NFAs for which no DFA can be constructed
(d) If L is accepted by an NFA with ε-transitions then L is also accepted by an NFA without ε transition

Q.16 The languages accepted by DPDA are _____.
(a) CFLs (b) CSLs
(c) DCFLs (d) RELs

Q.17 The languages accepted by PDA are _____
(a) CFLs (b) CSLs
(c) DCFLs (d) RELs

Q.18 The languages accepted by NPDA are _____
(a) CFLs (b) CSLs
(c) DCFLs (d) RELs

Q.19 The languages accepted by LBA are _____
(a) CFLs (b) CSLs
(c) DCFLs (d) RELs

Q.20 The languages accepted by TMs are _____
(a) CFLs (b) CSLs
(c) DCFLs (d) RELs

Q.21 Find the correct relation of expressive power of machines
(a) FA < PDA < LBA < TM
(b) FA > PDA > LBA > TM
(c) TM < PDA < LBA < FA
(d) TM > PDA > LBA > FA

Q.22 A language L is accepted by a PDA if and only if it is
(a) Regular language
(b) CSL
(c) Expressible by CFG
(d) Expressible by Right-linear grammar

Q.23 A language L is accepted by a TM if and only if it is _____.
(a) Recursive language
(b) Enumerable in lexicographical order
(c) Expressible by unrestricted grammar
(d) None of these

Q.24 Find the correct chomsky hierarchy
(a) Type0 ⊆ Type1 ⊆ Type2 ⊆ Type3
(b) Type3 ⊆ Type1 ⊆ Type2 ⊆ Type0
(c) Type0 ⊆ Type2 ⊆ Type1 ⊆ Type3
(d) None of these

Q.25 Which one of the following is correct for the input alphabet Σ?
(a) $\Sigma^* = \Sigma$ (b) $\Sigma^* = \Sigma^+ - \{\varepsilon\}$
(c) $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$ (d) $\Sigma^* = \Sigma^+ \cap \{\varepsilon\}$

Answer Key:

1. (a)	**2.** (b)	**3.** (b)	**4.** (d)	**5.** (d)
6. (c)	**7.** (c)	**8.** (b)	**9.** (d)	**10.** (d)
11. (b)	**12.** (d)	**13.** (a)	**14.** (b)	**15.** (c)
16. (c)	**17.** (a)	**18.** (a)	**19.** (b)	**20.** (d)
21. (a)	**22.** (c)	**23.** (c)	**24.** (d)	**25.** (c)

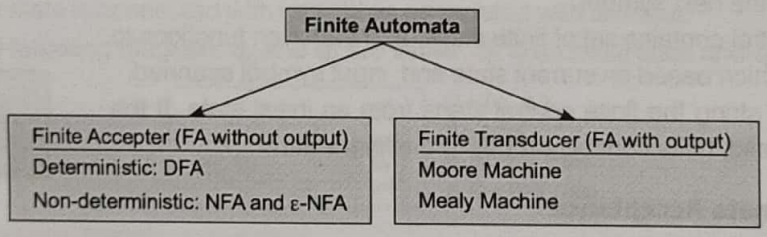
CHAPTER 02

Regular Languages & Finite Automata

2.1 Introduction

- Finite Automaton (FA) is also called as Finite State machine (FSM).
- Finite Automaton has no temporary storage hence it can remember finite information with the help of finite control which contain states and transitions.

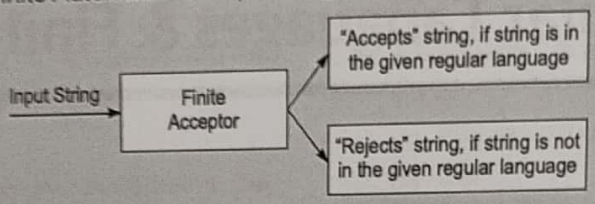
2.1.1 Types of Finite Automata



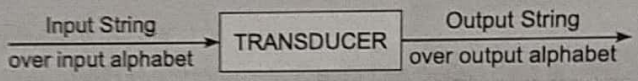
Finite automata is categorized into two types:

1. **Finite automata with output:** This is of two types Mealy machines and Moore machines.
   (a) In Mealy machine the output is associated with transition (state and input symbol)
   (b) In the Moore machine output is associated with only state.
2. **Finite automata without output:** This is of two types deterministic and non-deterministic.
   (a) **Deterministic Finite Automata (DFA):** It is deterministic finite automata. For every input symbol in DFA there is an exactly one transition from any state of finite automata. It accepts the string by halting at a final state and rejects the string by halting at a non-final state.
   (b) **Non-deterministic Finite Automata (NFA):** It is non deterministic finite automata. For every input symbol in NFA there is zero or more transitions from any state of finite automata. It accepts the string by halting at a final state and rejects the string either by halting at a non-final state or by halting in a dead configuration.
   (c) **ε-NFA:** It is a non deterministic finite automata which can include ε transitions. It has capability of changing the state without reading the input symbol.

- Acceptor is a machine which accepts or recognizes a language
- Transducer is a machine which produces an output
- Deterministic automaton is one in which each move is uniquely determined by reading an input.
- Non-deterministic automaton is one in which each move may have set of possible actions.
- Moore and Mealy machines are deterministic.
- Finite Automaton accepts a regular language as follows:

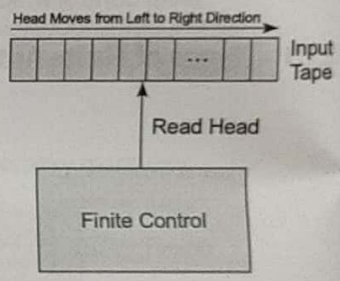


- Finite transducer produces an output for the given input as follows:



2.1.2 Model (Configuration) of Finite Automata

- Input tape contains a string over the given input alphabet.
- Read head is a pointer which reads one input symbol at a time and moves to the next symbol.
- Finite control contains set of finite states and transition functions to take an action based on current state and input symbol scanned.
- For every string the finite control starts from an initial state. If the string is valid, finite control reaches to the final state.



2.1.3 Finite Automata Acceptance

- Finite Automata accepts type-3 formal languages called as regular languages.
- Regular language is either finite or infinite language.
- A language is finite if it contains finite number of strings otherwise it is infinite language.
- Every finite language is regular hence it is possible to construct finite automata to accept the finite language.
- For infinite language the construction of finite automata may or may not possible. If the finite automata does not require infinite memory then it is possible to construct otherwise it is not possible to construct as it requires infinite number of states, which is not allowed in FA.

2.1.4 Regular Language

- All finite languages are regular
- If the language is infinite over unary alphabet and whose strings are forming an arithmetic progression then it is regular language, otherwise it is non-regular.
- If the language is infinite over two or more input symbols and which do not contain any dependency between symbols then the language is regular.

2.1.5 Representations of Regular Language

- Finite Automata (by construction)
- Regular Grammars (by writing productions)
- Regular Sets (by representation of a set)
- Regular Expression (by writing an expression)

2.1.6 Equivalence of Finite Automata

- DFA ≡ NFA ≡ ε-NFA
  All these three finite automata accept same class of languages called as regular languages
- Every DFA is also a special case of NFA
- NFA can be converted to DFA using subset construction
- ε-NFA can be converted to NFA and also to DFA

2.1.7 Representations of FA

FA = $(Q, \Sigma, \delta, q_0, F)$ is a 5-tuple notation

where, Q is set of finite states,

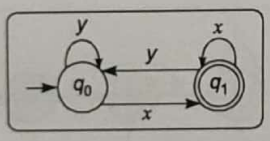
Σ is an input alphabet contains finite number of input symbols.

δ is transition function defined over transitions of FA for state and input.

q_0 is initial state or start state of FA, $q_0 \in Q$.

F is set of final states of FA.

1. State Transition Diagram:
   (a) Every state represented by a circle.
   (b) For an Initial state, a circle is associated with an arrow.
   (c) Final state represented by double circle.
   (d) Every state is connected with transitions associated with an input.
   (e) In the following diagram: q_0 and q_1 are states, q_0 is an initial state and q_1 is a final state.



$Q = \{q_0, q_1\}$
$\Sigma = \{x, y\}$
q_0 is initial state
q_1 is a final state

2. State transition table:
   (a) Table contains rows and columns
   (b) Each row corresponds to the state of DFA
   (c) Each column corresponds to the input symbol of DFA
   (d) Initial state is associated with an arrow
   (e) Final state is associated with a circle or star
   (f) In the following table: q_0 and q_1 are states, q_0 is an initial state and q_1 is a final state.

	x	y
$\rightarrow q_0$	q_1	q_0
$^*q_1$ or (q_1)	q_1	q_0

2.2 Deterministic Finite Automata (DFA)

- If for every input symbol of an alphabet there is an exactly one transition from every state of finite automata then such FA is called as DFA.

- DFA covers deterministic transitions for all valid and invalid strings in the given regular language.
- For every valid string, DFA reaches to the final state and for every invalid string it reaches to the non-final state.

2.2.1 Specifications of DFA

DFA = $(Q, \Sigma, \delta, q_0, F)$ is a 5-tuple notation

where Q is the set of finite states,

Σ is an input alphabet contain finite number of input symbols.

δ is a transition function defined over transitions of FA for state and input, $(\delta : Q \times \Sigma \to Q)$

q_0 is an initial state or start state of DFA, $(q_0 \in Q)$.

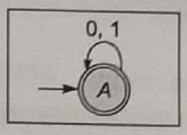
F is the set of final states of DFA $(F \subseteq Q)$.

2.2.2 Constructing Minimal DFA for the Regular Languages

Example-2.1 Construct a minimal DFA that accepts a language where it contain all binary strings.

Solution:

Language L = {ε, 0, 1, 00, 01, 10, 11...} = $\{w \mid w \in \{0, 1\}^*\}$

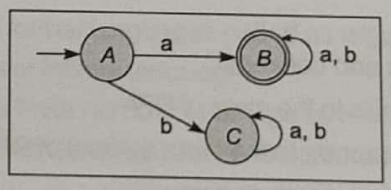


Initial state = A, Final states = {A}

Example-2.2 Language contain all strings of a's and b's over the alphabet $\Sigma = \{a, b\}$, where every string is starting with a.

Solution:

L = {a, aa, ab...}

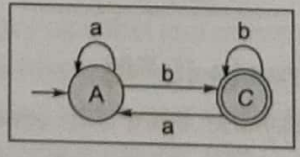


Initial state = A, Final states = {B}, reject state = C, and non-final states = {A, C}

Reject state is constructed to reject all invalid combinations in finite automata.

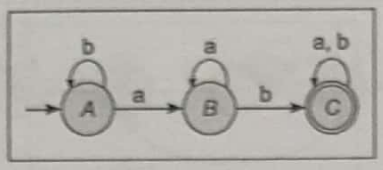
Example-2.3 Language contain all strings of a's and b's with an input alphabet of 'a' and 'b', where every string is ending with b.

Solution:



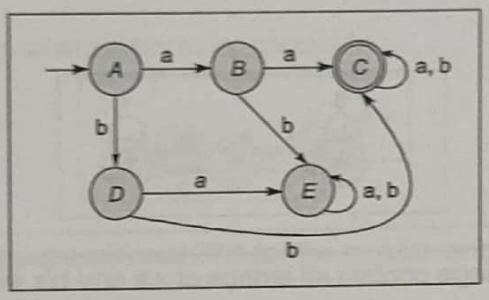
Example-2.4 Language contain all strings of a's and b's with an input alphabet of 'a' and 'b', where each string contain substring 'ab'.

Solution:



Example-2.5 Language contain all strings of a's and b's with an input alphabet of 'a' and 'b', where each string starts with aa or bb.

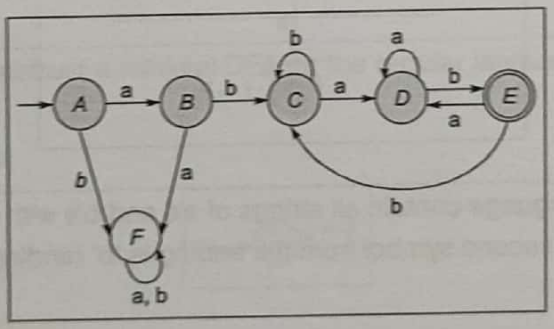
Solution:



Note: States F and C can be combined into a single state.

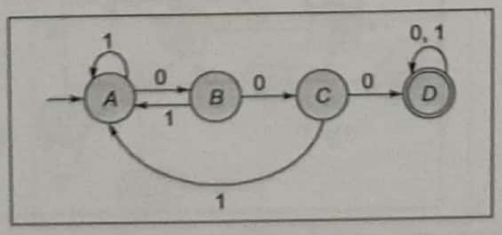
Example-2.6 Language contain all strings of a's and b's with an input alphabet of 'a' and 'b', where each string starts with ab and ends with ab. (Excluding the string ab)

Solution:



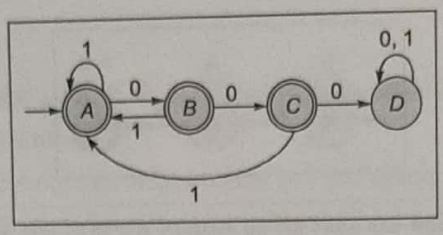
Example-2.7 Language contain all binary strings, where each string contains 000, as a substring.

Solution:



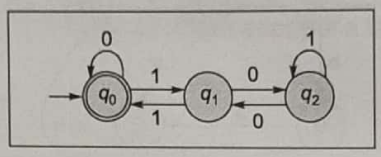
Example-2.8 Language contain all binary strings, where each string not contains 000.

Solution:



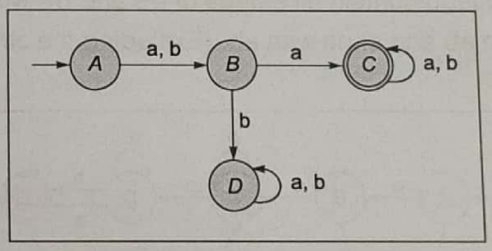
Example-2.9 Language contain all binary strings, where the decimal value of each string is divisible by 3.

Solution:



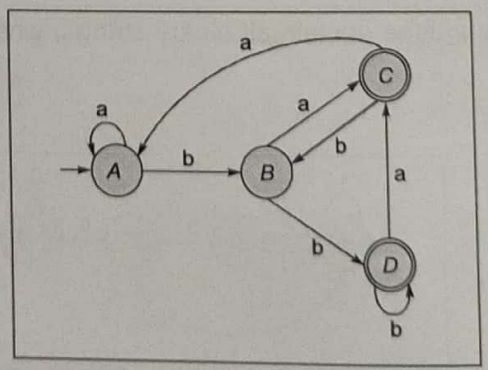
Example-2.10 Language contain all strings of a's and b's with an input alphabet of 'a' and 'b', where each string contain second symbol from the beginning is 'a'.

Solution:



Example-2.11 Language contain all strings of a's and b's with an input alphabet of 'a' and 'b', where each string contain second symbol from the ending is 'b' (ending with 'ba' or 'bb').

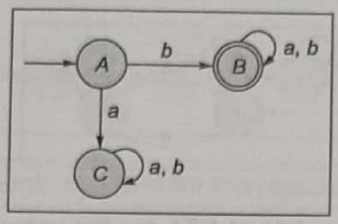
Solution:



Example-2.12 Construct a minimal DFA for the regular language $L = \{bw \mid w \in \{a, b\}^*\}$

Solution:

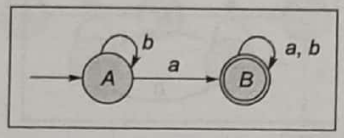
L is the set of all strings start with '*b*'.



Example-2.13 Construct a minimal DFA for the regular language $L = \{w_1 a\, w_2 \mid w_1, w_2 \in \{a, b\}^*\}$

Solution:

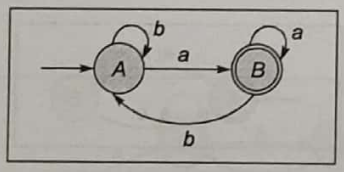
L is the set of all strings where every string contain '*a*'.



Example-2.14 Construct a minimal DFA for the regular language $L = \{w\, a \mid w \in \{a, b\}^*$

Solution:

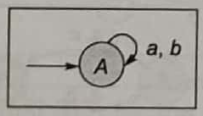
L is the set of all strings end with '*a*'.



Example-2.15 Construct a minimal DFA for the regular language $L = \{\}$ over $\Sigma = \{a, b\}$

Solution:

L is empty language.

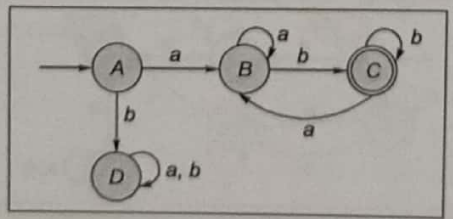


No final state in DFA.

Example-2.16 Construct a minimal DFA for the regular language $L = \{a\, w\, b \mid w \in \{a, b\}^*\}$

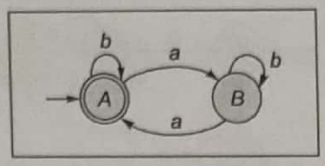
Solution:

Set of all strings start with *a* and end with *b*.



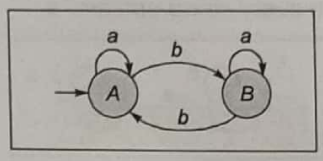
Example-2.17 Construct a minimal DFA for the regular language L = {$w \mid w \in \{a, b\}^*$, w contains an even number of a's}

Solution:



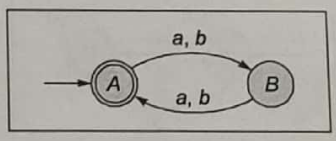
Example-2.18 Construct a minimal DFA for the regular language L = {$w \mid w \in \{a, b\}^*$, w contains an odd number of b's}

Solution:



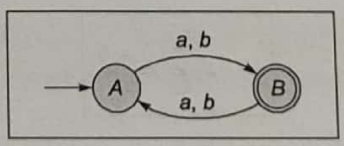
Example-2.19 Construct a minimal DFA for the regular language L = {$w \mid w \in \{a, b\}^*$, w has even length}

Solution:



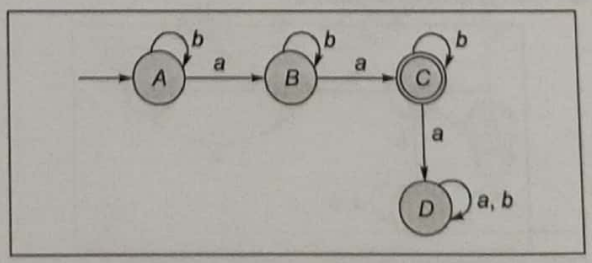
Example-2.20 Construct a minimal DFA for the regular language L = {$w \mid w \in \{a, b\}^*$, w has odd length}

Solution:



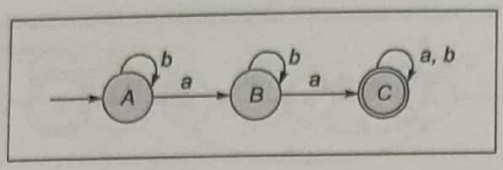
Example-2.21 Construct DFA for the regular language L = {$w \mid w$ contains exactly two a's, $w \in \{a, b\}^*$}.

Solution:



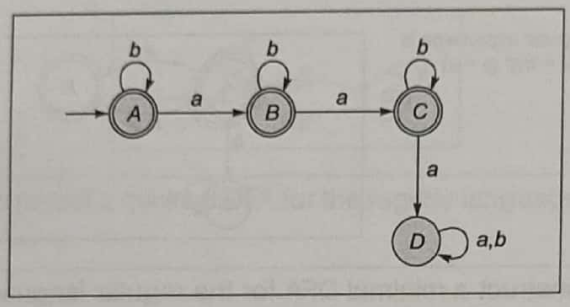
Example - 2.22 Construct a minimal DFA for the regular language L = {w | w contains atleast two a's, $w \in \{a, b\}^*$}

Solution:



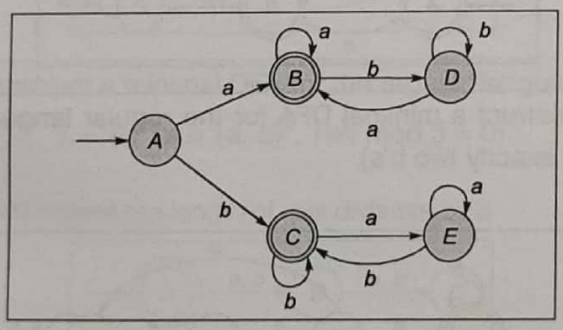
Example - 2.23 Construct a minimal DFA for the regular language L = {w | w contains atmost two a's, $w \in \{a, b\}^*$}

Solution:



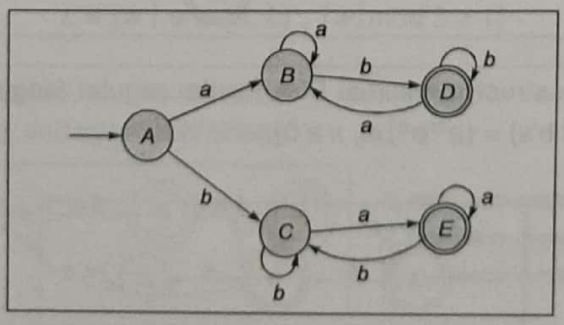
Example - 2.24 Construct a minimal DFA for the regular language L = {w | w starts and ends with same symbol, $w \in \{a, b\}^*$}

Solution:



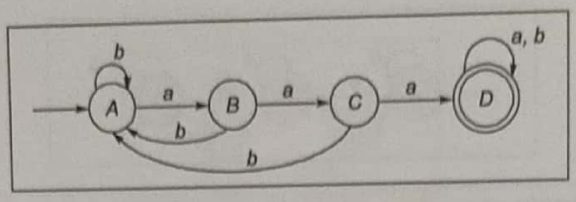
Example - 2.25 Construct a minimal DFA for the regular language L = {w | w start and end with different symbols, $w \in \{a, b\}^*$}

Solution:



Example - 2.26 Construct a minimal DFA for the regular language $L = \{w \mid w$ contain 3 consecutive a's, $w \in \{a, b\}^*\}$

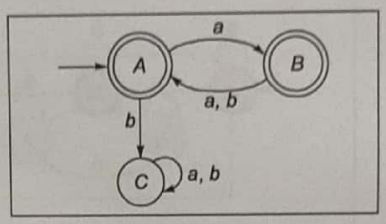
Solution:



Example - 2.27 Construct a minimal DFA for the regular language $L = \{w \mid w$ contain 'a' in every odd position, $w \in \{a, b\}^*\}$

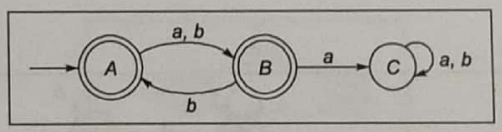
Solution:

Regular expression is
$(a(a + b))^* (\varepsilon + a)$



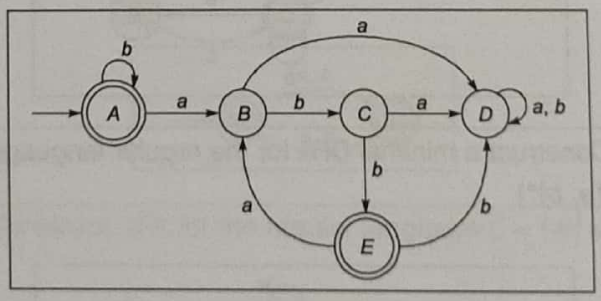
Example - 2.28 Construct a minimal DFA for the regular language $L = \{w \mid w$ contains 'b' in every even position, $w \in \{a, b\}^*\}$

Solution:



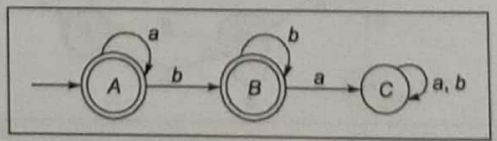
Example - 2.29 Construct a minimal DFA for the regular language $L = \{w \mid w \in \{a, b\}^*, w$ contains every 'a' followed by exactly two b's$\}$

Solution:



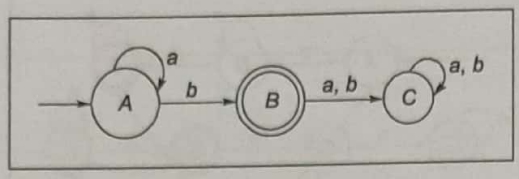
Example - 2.30 Construct a minimal DFA for the regular language $L = \{w \mid w \in \{a, b\}^*, w$ contains all a's followed by all b's$\} = \{a^m b^n \mid m, n \geq 0\}$.

Solution:



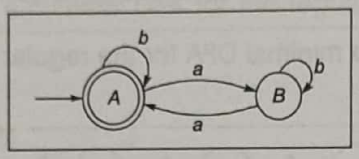
Example - 2.31 Construct a minimal DFA for the regular language $L = \{w \mid w$ contains all a's followed by single 'b', $w \in \{a, b\}^*\} = \{a^m b \mid m \geq 0\}$.

Solution:



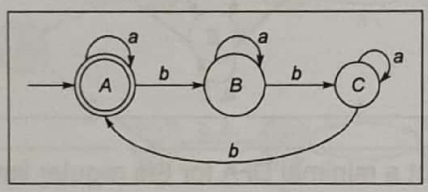
Example - 2.32 Construct a minimal DFA for the regular language $L = \{w \mid w \in \{a, b\}^*$, number of a's in w is divisible by 2$\}$

Solution:



Example - 2.33 Construct a minimal DFA for the regular language $L = \{w \mid w \in \{a, b\}^*$, number of b's in w is divisible by 3$\}$

Solution:

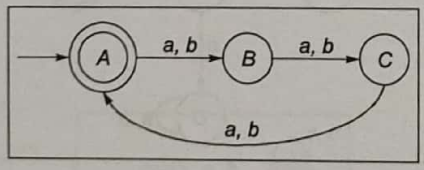


Example - 2.34 Construct a minimal DFA for the regular language

$$L = \{w \mid w \in \{a, b\}^*, |w| \bmod 3 = 0\}$$

Solution:

Here, $|w| \bmod 3 = 0$ implies the length of w is divisible by 3



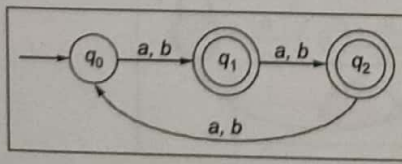
Example - 2.35 Construct a minimal DFA for the regular language

$$L = \{w \mid w \in \{a, b\}^*, |w| \bmod 3 \geq 1\}$$

Solution:

Mod 3 has remainders: 0, 1, 2 $\Rightarrow$ 3 states.

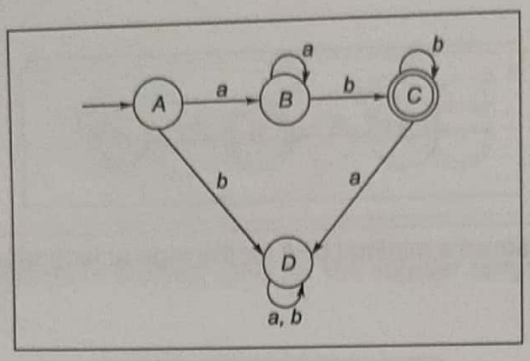
$|w| \bmod 3 \geq 1 \Rightarrow q_1$ and q_2 are final states.



q_0 represents remainder 0
q_1 represents remainder 1
q_2 represents remainder 2

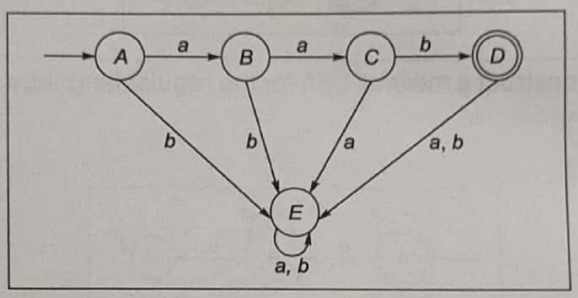
Example - 2.36 Construct a minimal DFA for the regular language $L = \{a^m b^n | m \geq 1, n \geq 1\}$

Solution:



Example - 2.37 Construct a minimal DFA for the regular language $L = \{a^m b^n | m = 2, n = 1\}$

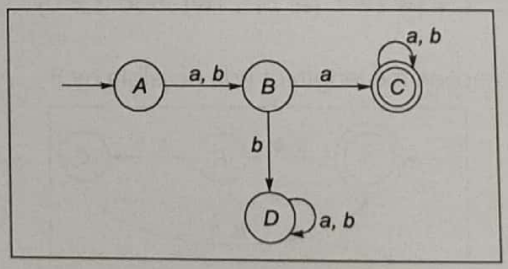
Solution:



Example - 2.38 Construct a minimal DFA for the regular language

$$L = \{x\ a\ w | x \in \{a, b\}, w \in \{a, b\}^*\}$$

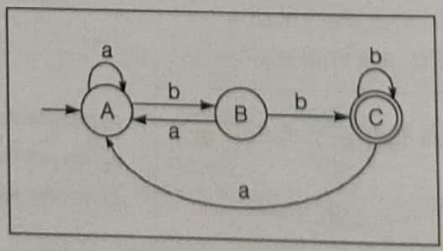
Solution:



Example - 2.39 Construct a minimal DFA for the regular language $L = \{w\,bb | w \in \{a, b\}^*\}$.

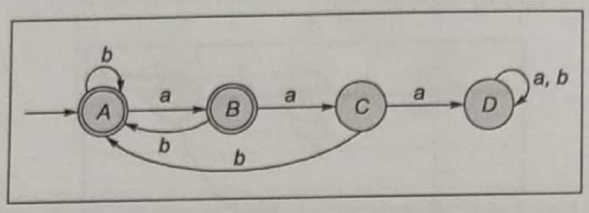
Solution:

Every string in L ends with 'bb'.



Example - 2.40 Construct a minimal DFA for the regular language $L = \{w \mid$ every occurrence of aa is followed by a 'b' in each w, $w \in \{a, b\}^*\}$

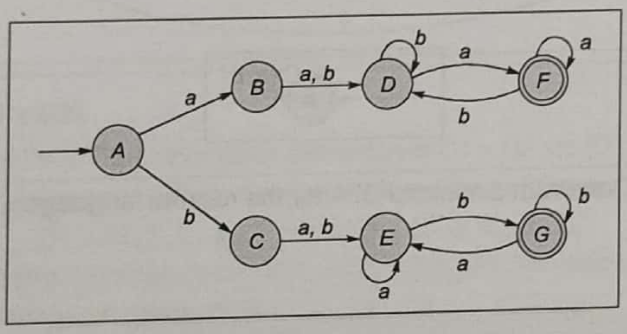
Solution:



Example - 2.41 Construct a minimal DFA for the regular language $L = \{wxw^R \mid x \in \{a, b\}^+$, $w \in \{a, b\}^+$, w^R is the reverse of $w\}$

Solution:

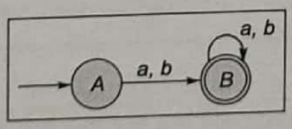
$L = (a\{a, b\}^+a) \cup (b\{a, b\}^+ b)$



Example - 2.42 Construct a minimal DFA for the regular language $L = \{w\,xw^R \mid x \in \{a, b\}^+$, $w \in \{a, b\}^*$, w^R is the reverse of w

Solution:

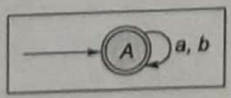
$L = (a + b)^+$



Example - 2.43 Construct a minimal DFA for the regular language $L = \{wxwy \mid x, y \in \{a, b\}^*$, $w \in \{a,b\}^*\}$

Solution:

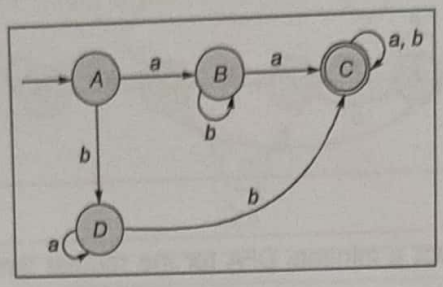
$L = (a + b)^*$



Example - 2.44 Construct a minimal DFA for the regular language $L = \{wxwy | x, y \in \{a, b\}^*, w \in \{a, b\}^+\}$

Solution:

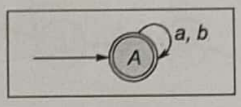
$L = a\{a, b\}^* a\{a, b\}^* \cup b\{a, b\}^* b\{a,b\}^*$



Example - 2.45 Construct a minimal DFA for the regular language $L = \{xwyw | x, y \in \{a, b\}^*, w \in \{a, b\}^*\}$

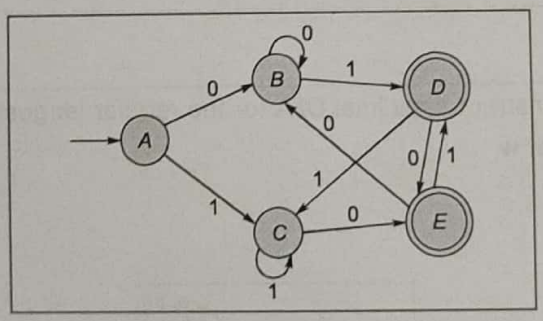
Solution:

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



Example - 2.46 Construct a minimal DFA for the regular language $L = \{w | w \in \{0, 1\}^*$ and last two bits are different$\}$

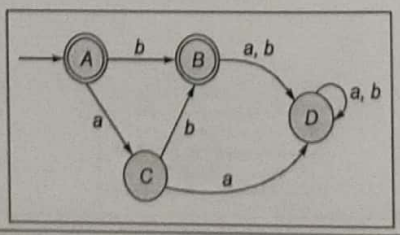
Solution:



Example - 2.47 Construct a minimal DFA for the regular language $L = \{a^m b^n | m \le n < 2\}$

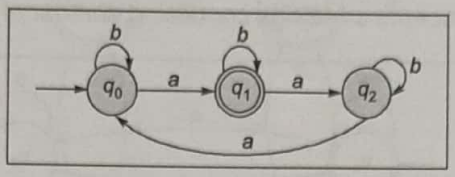
Solution:

$L = \{a^0b^0, a^0b^1, a^1b^1\} = \{\varepsilon, b, ab\}$



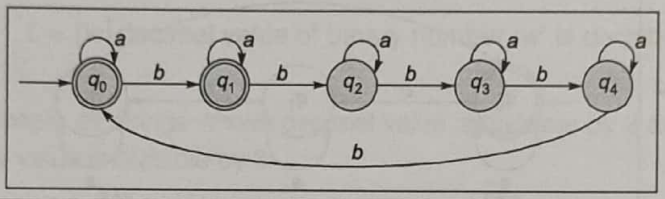
Example - 2.48 Construct a minimal DFA for the regular language $L = \{w \mid$ Number of a's in w is congruent to 1 modulo 3, $w \in \{a,b\}^*\} = \{w \mid n_a(w) \bmod 3 = 1, w \in \{a, b\}^*\}$.

Solution:



Example - 2.49 Construct a minimal DFA for the regular language $L = \{w \mid$ (Number of b's in w) mod $5 < 2$, $w \in \{a, b\}^*\}$

Solution:



2.2.3 Compound FA (D1 × D2)

Let D1 and D2 be two DFA. Then operations defined over D1 × D2 are (D1 ∪ D2), (D1 ∩ D2), (D1 – D2), and (D2 – D1). Construction of compound FA for the given D1 and D2 as follows:

$$D1 = (Q, \Sigma, \delta_1, q_0, F_1) \text{ and } D2 = (Q, \Sigma, \delta_2, q_0, F_2)$$

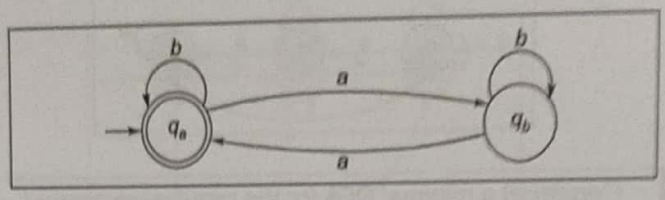
1. The number of states in compound FA (D1 × D2) is equal to mn, where m is the number of states of D1 and n is the number of states of D2.
2. Initial state of compound FA is combination of initial states of D1 and D2. Let q_0 be the start state of D1 and q_a be the start of D2. (q_0, q_a) is initial state of D1 × D2.
3. Let $Q1 = \{q_0, q_1, q_2\}$ is set of states of D1, and $Q2 = \{q_a, q_b\}$ is set of states of D2. Then $Q1 \times Q2 = \{(q_0, q_a), (q_0, q_b), (q_1, q_a), (q_1, q_b), (q_2, q_a), (q_2, q_b)\}$
4. Assume q_2 is the final state of D1 and q_b is the final state of D2. Final states depends on type of compound finite automata (D1 × D2)
   (a) (D1 ∪ D2) final states: Make the states of (Q1 × Q2) as final, if final state of D1 **or** final state of D2 contain in any of the states. (D1 ∪ D2) accepts L(D1) ∪ L(D2).
   Final sates of (D1 ∪ D2) = $\{(q_2, q_a), (q_2, q_b), (q_0, q_b), (q_1, q_b)\}$, where F1 = $\{q_2\}$ and F2 = $\{q_b\}$
   (b) (D1 ∩ D2) final states: Make the states of (Q1 × Q2) is final, if both final state of D1 and final state of D2 contain in any of the states. (D1 ∩ D2) accepts L(D1) ∩ L(D2).
   Final sates of (D1 ∩ D2) = $\{(q_2, q_b)\}$
   (c) (D1 – D2) final states: Make the states of (Q1 × Q2) as final, if final state of D1 and non-final state of D2 contain in any of the states. (D1 – D2) accepts L(D1) – L(D2).
   Final sates of (D1 – D2) = $\{(q_2, q_a)\}$ where q_2 is final state of D1 and q_a is non-final state of D2.
   (d) (D2 – D1) final states: Make the states of (Q1× Q2) as final, if final state of D2 and non-final state of D1 contain in any of the states. (D2 – D1) accepts L(D2) – L(D1).
   Final sates of (D2 – D1) = $\{(q_0, q_b), (q_1, q_b)\}$, where q_b is final state of D2, q_0 and q_1 are non-final states of D1.

Example - 2.50 Construct a minimal FA that accepts a language over $\Sigma = \{a, b\}$ where every string contain number of a's is divisible by 2 and number of b's is divisible by 3.

Solution:

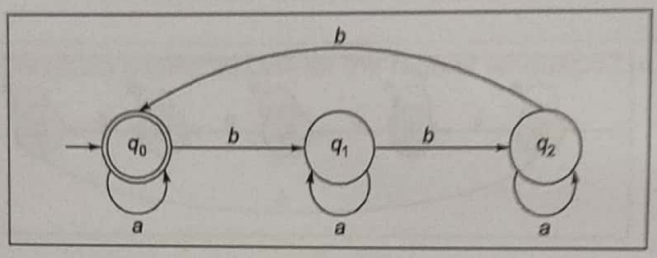
Let FA1 be a DFA that accepts all strings containing number of *a*'s is divisible by 2.

FA1



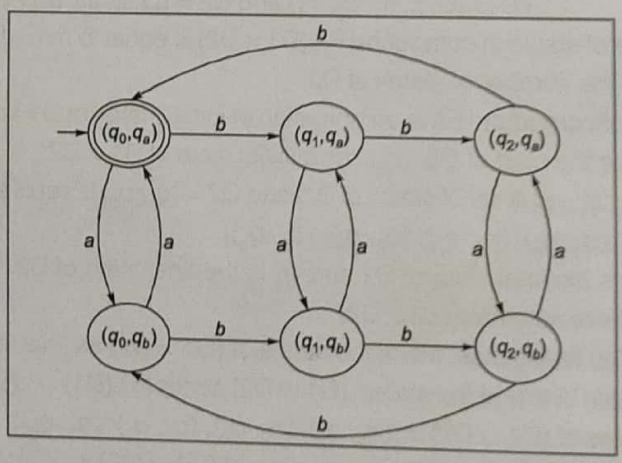
Let FA2 be a DFA that accepts all strings containing number of *b*'s is divisible by 3.

FA2



Let FA1 ∩ FA2 is a DFA that accepts all strings containing number of *a*'s is divisible by 2 and number of *b*'s is divisible by 3.

FA1 ∩ FA2

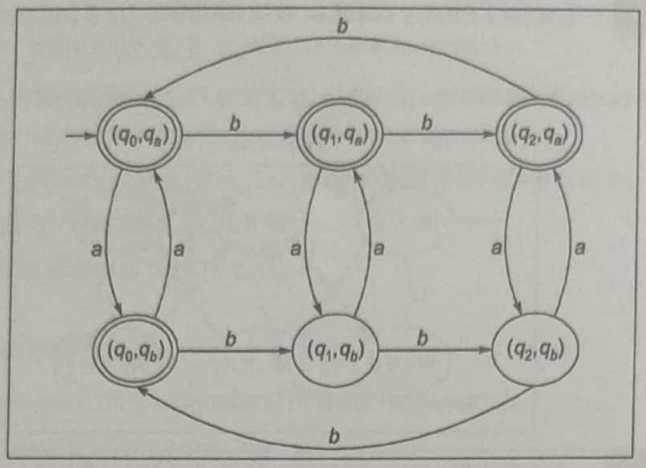


Example - 2.51 Construct a minimal FA that accepts a language over $\Sigma = \{a, b\}$ where every string contain number of a's is divisible by 2 or number of b's is divisible by 3.

Solution:

Let FA1 be a DFA that accepts all strings containing number of a's is divisible by 2.
Let FA2 be a DFA that accepts all strings containing number of b's is divisible by 3.
Let FA1 ∪ FA2 is a DFA that accepts all strings containing number of a's is divisible by 2 or number of b's is divisible by 3.

FA1 ∪ FA2

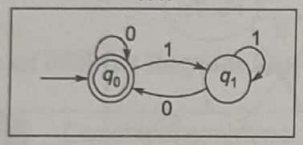


Example - 2.52 $L = \{w |$ decimal value of binary number 'w' is divisible by 2 or 3$\}$

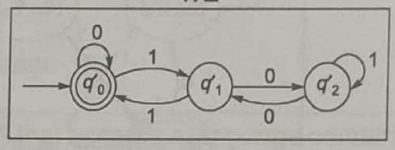
Solution:

Let FA1 be accepts all strings whose decimal value is divisible by 2 and FA2 accepts all strings whose decimal value is divisible by 3.

FA1

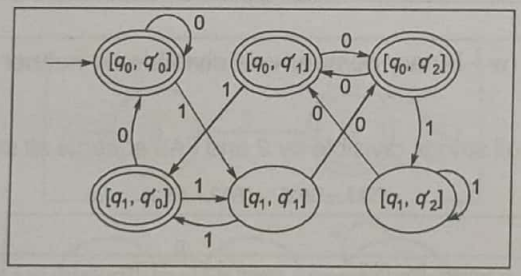


FA2



FA1 ∪ FA2 accepts all strings divisible by 2 or 3.

FA1 ∪ FA2

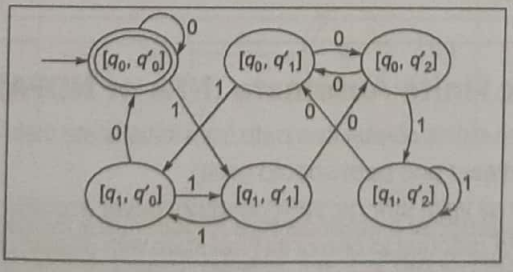


Example - 2.53 $L = \{w |$ Binary number 'w' is divisible by both 2 and 3$\}$

Solution:

Let FA1 be accepts all strings divisible by 2 and FA2 accepts all strings divisible by 3.

FA1 ∩ FA2

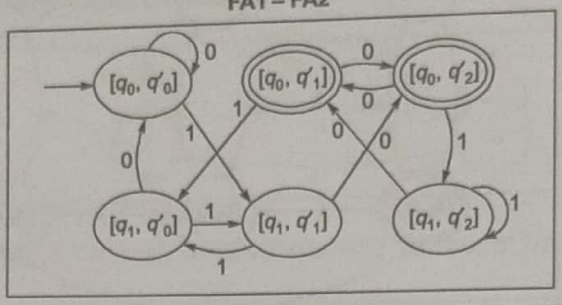


Example - 2.54 $L = \{w \mid$ Binary number w is divisible by 2 but not divisible by 3$\}$

Solution:

Let FA1 be accepts all strings divisible by 2 and FA2 accepts all strings divisible by 3.

FA1 – FA2

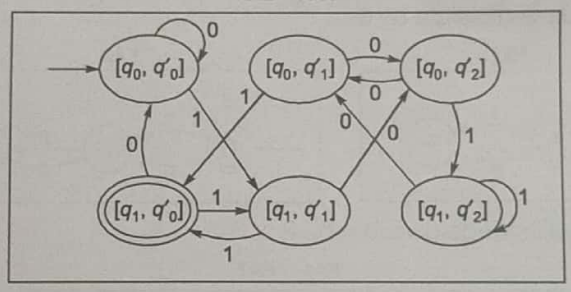


Example - 2.55 $L = \{w \mid$ Binary number w is divisible by 3 but not divisible by 2$\}$

Solution:

Let FA1 be accepts all strings divisible by 2 and FA2 accepts all strings divisible by 3.

FA2 – FA1

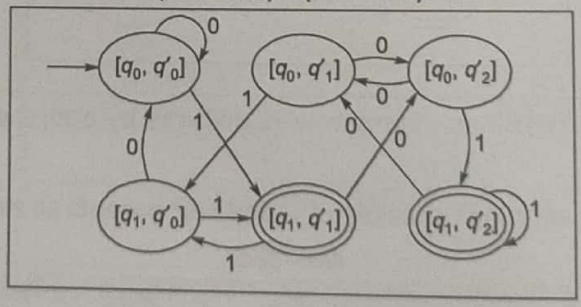


Example - 2.56 $L = \{w \mid$ Binary number w is divisible by neither 2 nor 3$\}$

Solution:

Let FA1 be accepts all strings divisible by 2 and FA2 accepts all strings divisible by 3.

(FA1 – FA2) ∪ (FA2 – FA1)



2.3 Non-deterministic Finite Automata (NFA or NDFA)

For every valid string there exists atleast one path from initial state that reaches to final state. (Every path may not reach to final state but atleast one path should exist).

NFA cover transitions for all valid strings. NFA need not cover transitions for invalid combinations.

For every valid string, NFA reaches to one of its final state with atleast one path. For invalid strings it may or may not contain path that reaches to non-final state.

2.3.1 Specification of NFA

NFA = $(Q, \Sigma, \delta, q_0, F)$ is a 5-tuple notation.

where Q is the set of finite states,

Σ is an input alphabet contain finite number of input symbols

δ is a transition function defined over transitions of NFA for state and input, ($\delta: Q \times \Sigma \to 2^Q$)

q_0 is a start state of NFA, ($q_0 \in Q$)

F is the set of final states of NFA ($F \subseteq Q$)

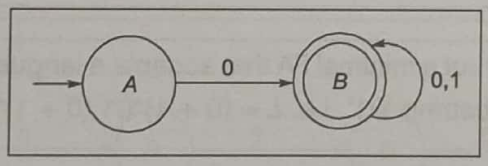
2.3.2 Comparison of DFA and NFA

- NFA and DFA accepts the type-3 languages (Regular languages).
- NFA and DFA are equivalent hence they have same power.
- Every DFA is a special case of NFA but every NFA need not be a DFA.
- $Q \times \Sigma \to Q$ is DFA transition function and $Q \times \Sigma \to 2^Q$ is NFA transition function.
- Construction of NFA is easier than DFA for many problems.
- In NFA, zero or more number of transitions from any state for each input symbol. But in DFA there is exactly one transition from every state for each input symbol.

2.3.3 Construction of NFA for Regular Language

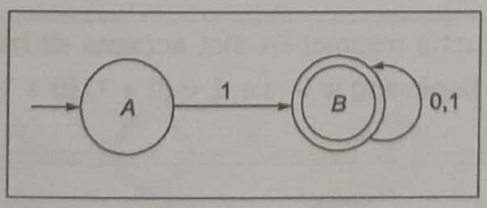
Example - 2.57 Construct a minimal FA that accepts a language containing all binary strings where each string is starting with 0.

Solution:



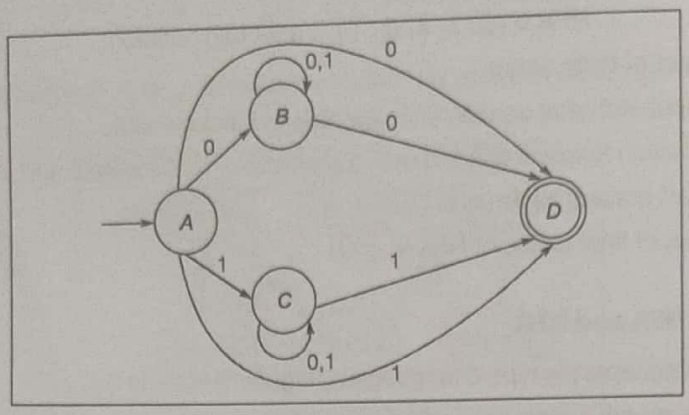
Example - 2.58 Construct a minimal FA that accepts a language containing all binary strings where each string starts with 1.

Solution:



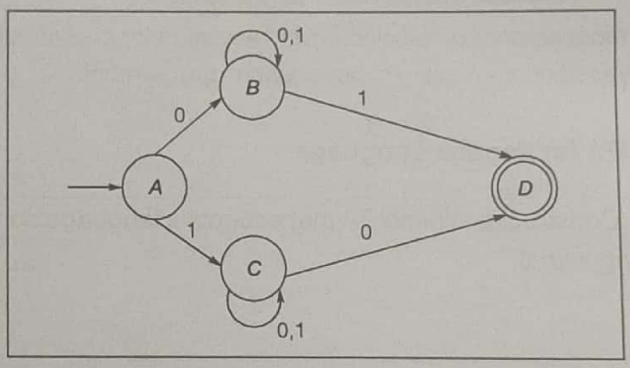
Example - 2.59 Construct a minimal FA that accepts a language containing all binary strings where each string start and end with the same symbol. i.e. $L = 0(0 + 1)^*0 + 1(0 + 1)^*1 + 0 + 1$.

Solution:



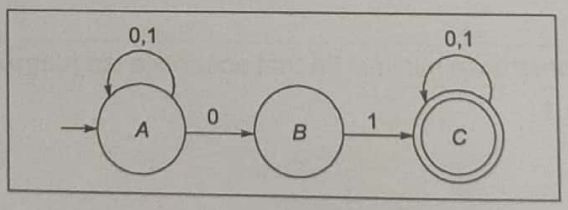
Example - 2.60 Construct a minimal FA that accepts a language containing all binary strings where each string starts and ends with different symbol. i.e. $L = 0(0 + 1)^*1 + 1(0 + 1)^*0$.

Solution:



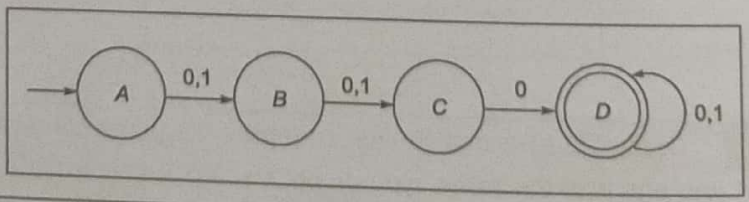
Example - 2.61 Construct a minimal FA that accepts a language containing all binary strings where each string contains a substring '01'. i.e. $L = (0 + 1)^*01\ (0 + 1)^*$.

Solution:



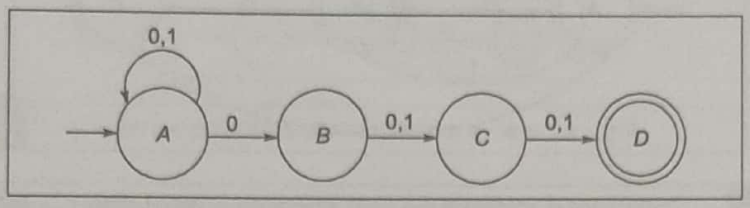
Example - 2.62 Construct a minimal FA that accepts all binary strings where each string contains a third symbol from the beginning is 0. i.e. $L = (0 + 1)\ (0 + 1)\ 0\ (0 + 1)^*$.

Solution:



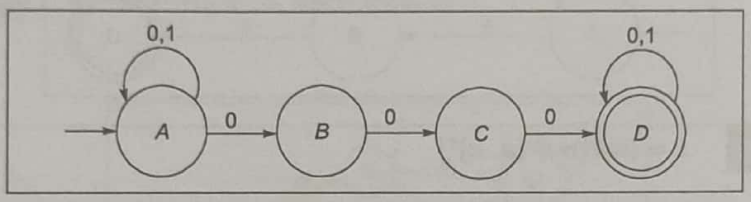
Example - 2.63 Construct a minimal FA that accepts all binary strings where each string contains a third symbol from the ending (right hand side) is 0. i.e. $L = (0 + 1)^*0\ (0 + 1)(0 + 1)$.

Solution:



Example - 2.64 Construct a minimal FA that accepts all binary strings where each string contain '000' as a substring. i.e. $L = (0 + 1)^*000\ (0 + 1)^*$.

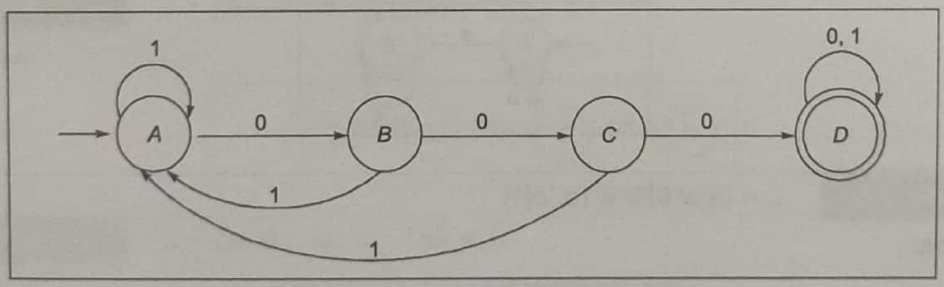
Solution:



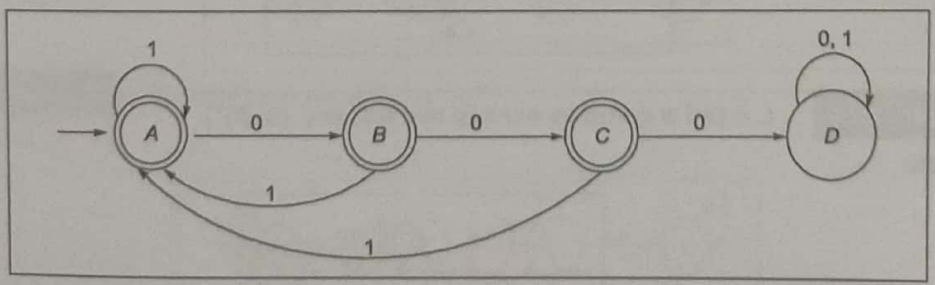
Example - 2.65 Construct a minimal FA that accepts all binary strings where no string contain '000' as a substring.

Solution:

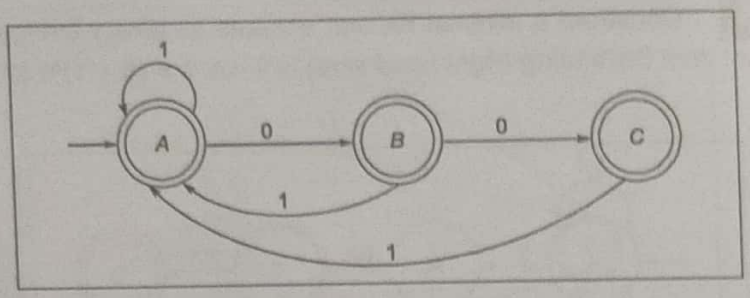
Since this language has a negative condition we will design a minimal DFA for its complement (containing '000' as substring), complement it and then remove the trap state if any, to get the required minimal NFA. The minimal DFA (for containing '000' as substring) is given below:



Its complement which will be a minimal DFA accepting the required language (not containing '000' as substring) is given below:

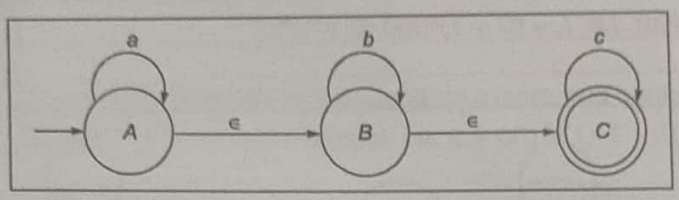


Now, removing the trap state D we get the required minimal NFA for the given language.



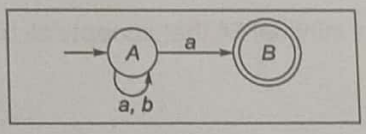
Example - 2.66 Construct a minimal FA that accepts a language $L = \{a^m b^n c^k | m, n, k \geq 0\}$.

Solution:



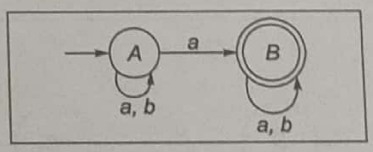
Example - 2.67 $L = \{wa | w \in \{a, b\}^*\}$

Solution:



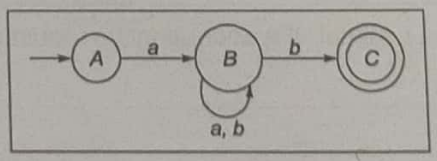
Example - 2.68 $L = \{w_1 a w_2 | w_1, w_2 \in \{a, b\}^*\}$

Solution:



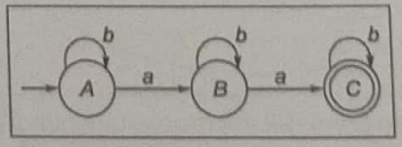
Example - 2.69 $L = \{a w b | w \in \{a, b\}^*\}$

Solution:



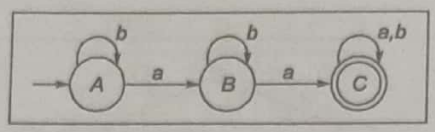
Example - 2.70 $L = \{w | w$ contains exactly two a's, $w \in \{a, b\}^*\}$

Solution:



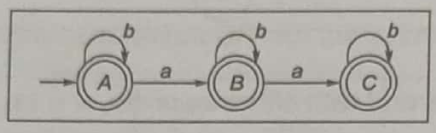
Example - 2.71 $L = \{w \mid w \text{ contains atleast two a's, } w \in \{a, b\}^*\}$

Solution:



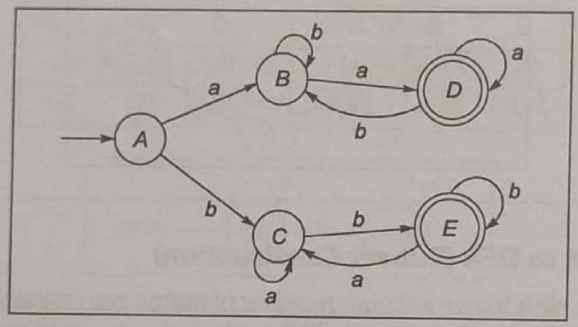
Example - 2.72 $L = \{w \mid w \text{ contains atmost two a's, } w \in \{a, b\}^*\}$

Solution:



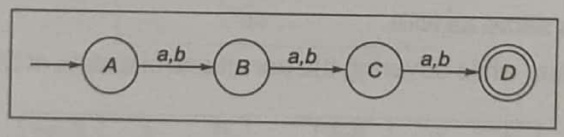
Example - 2.73 $L = \{x\, w\, x \mid x \in \{a, b\}, w \in \{a, b\}^*\}$

Solution:



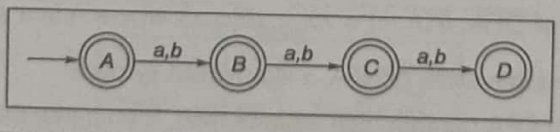
Example - 2.74 $L = \{w \mid w \in \{a, b\}^*, |w| = 3\}$

Solution:



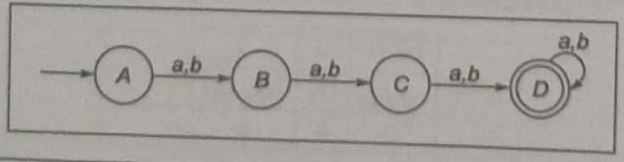
Example - 2.75 $L = \{w \mid w \in \{a, b\}^*, |w| \leq 3\}$

Solution:



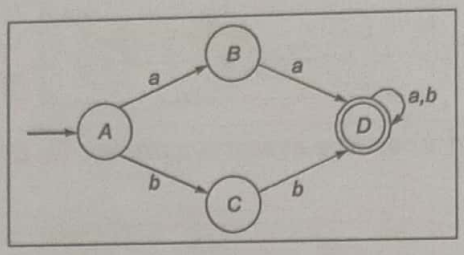
Example - 2.76 $L = \{w \mid w \in \{a, b\}^*, |w| \geq 3\}$

Solution:



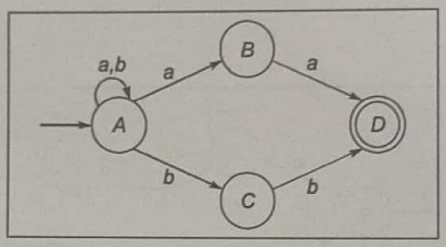
Example-2.77 $L = \{w | w$ start with either aa or bb, $w \in \{a, b\}^*\}$. i.e. $L = (aa + bb)(a + b)^*$.

Solution:



Example-2.78 $L = \{w | w$ end with either aa or bb, $w \in \{a, b\}^*\}$. i.e. $L = (a + b)^*(aa + bb)$.

Solution:



2.3.4 Conversion from NFA to DFA (Subset Construction)

If n states are present in NFA then minimum number of states are present in DFA is 1 and maximum is 2^n states.

$$\text{NFA} = (Q, \Sigma, \delta, q_0, F) \text{ and DFA} = (2^Q, \Sigma, \delta', q_0, F')$$

Every NFA can be converted to DFA as following:

1. Initial state of DFA is same as NFA
2. If there is any new state while covering the transitions of a state of DFA then cover the transitions for the new state also.
3. Repeat the step-2 until there are no new states to cover the transitions.
   Let δ' be a transition function of DFA and δ is transition function of NFA.
   Transition function of DFA: $\delta'(A, x) = \cup_{\forall q \in A} \delta(q, x)$, where A is a set of states of NFA and A is the state of DFA, q is a single state of NFA, x is an input symbol.
4. If any of the states of DFA contain final state of NFA, then make that state as final.
   If n states are there in some NFA then the equivalent DFA has atmost 2^n states.

Example-2.79 Construct an equivalent DFA for the following NFA.

NFA	0	1
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
$^*q_1$	ϕ	ϕ

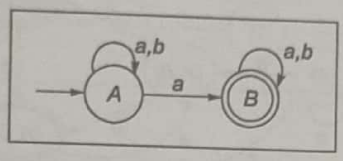
Solution:

DFA initial state q_0 is same as initial state of NFA.

DFA	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
$*\{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_1\}$

While covering the transitions of q_0, the new state is created as $\{q_0, q_1\}$.
Final state of DFA is $\{q_0, q_1\}$ which contains a final state q_1 of NFA.

Example - 2.80 Construct an equivalent DFA for the following NFA.

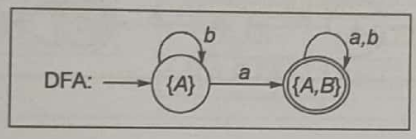


Solution:

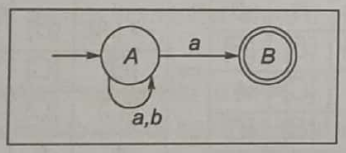
NFA	a	b
$\rightarrow A$	$\{A,B\}$	$\{A\}$
$*B$	$\{B\}$	$\{B\}$

$\Rightarrow$

DFA	a	b
$\rightarrow \{A\}$	$\{A,B\}$	$\{A\}$
$*\{A,B\}$	$\{A,B\}$	$\{A,B\}$



Example - 2.81 Construct an equivalent DFA for the following NFA.

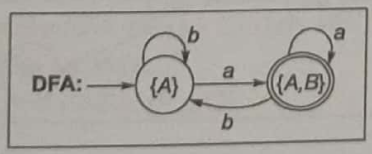


Solution:

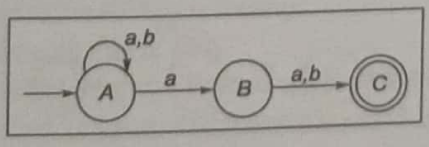
NFA	a	b
$\rightarrow A$	$\{A,B\}$	$\{A\}$
$*B$	ϕ	ϕ

$\Rightarrow$

DFA	a	b
$\rightarrow \{A\}$	$\{A,B\}$	$\{A\}$
$*\{A,B\}$	$\{A,B\}$	$\{A\}$



Example - 2.82 Construct an equivalent DFA for the following NFA.

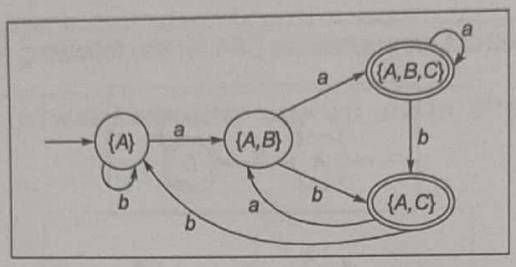


Solution:

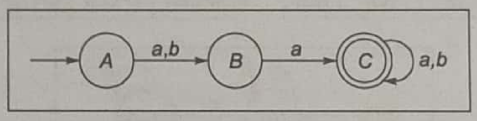
NFA	a	b
→ A	{A,B}	{A}
B	{C}	{C}
*C	ϕ	ϕ

⇒

DFA	a	b
→ {A}	{A,B}	{A}
{A,B}	{A,B,C}	{A,C}
*{A,B,C}	{A,B,C}	{A,C}
*{A,C}	{A,B}	{A}



Example - 2.83 Construct an equivalent DFA for the following NFA.

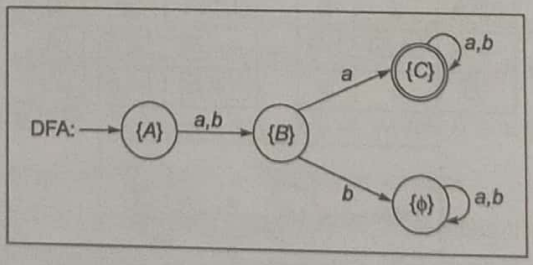


Solution:

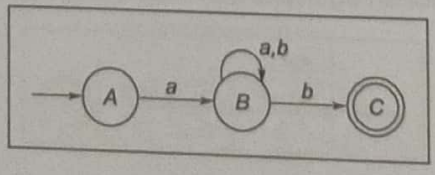
NFA	a	b
→ A	{B}	{B}
B	{C}	ϕ
*C	{C}	{C}

⇒

DFA	a	b
→ {A}	{B}	{B}
{B}	{C}	ϕ
*{C}	{C}	{C}
ϕ	ϕ	ϕ



Example - 2.84 Construct an equivalent DFA for the following NFA.



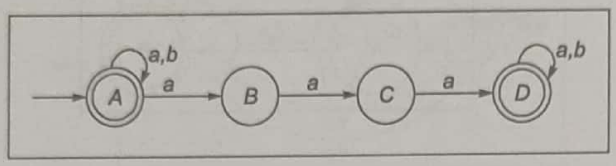
Solution:

NFA	a	b
→ A	{B}	ϕ
B	{B}	{B,C}
*C	ϕ	ϕ

⇒

DFA	a	b
→ {A}	{B}	ϕ
{B}	{B}	{B,C}
*{B,C}	{B}	{B,C}
ϕ	ϕ	ϕ

Example - 2.85 Construct an equivalent DFA for the following NFA.



Solution:

NFA	a	b
→ A	{A,B}	{A}
B	{C}	ϕ
C	{D}	ϕ
*D	{D}	{D}

⇒

DFA	a	b
→ {A}	{A,B}	{A}
{A,B}	{A,B,C}	{A}
{A,B,C}	{A,B,C,D}	{A}
*{A,B,C,D}	{A,B,C,D}	{A,D}
*{A,D}	{A,B,D}	{A,D}
*{A,B,D}	{A,B,C,D}	{A,D}

2.4 Epsilon NFA (ε-NFA)

For every valid string there exists a path from initial state that reaches to final state. ε-NFA is same as NFA but it can include epsilon transitions.

ε-NFA cover transitions for all valid strings similar to NFA.

For every valid string, ε-NFA reaches to one of its Final state with atleast one path. For invalid strings it may or may not contain a path that reaches to the non-final state.

2.4.1 Specification of ε-NFA

ε-NFA = (Q, Σ, δ, q_0, F) is a 5-tuple notation.

where, Q is the set of finite states,
Σ is an input alphabet contain finite number of input symbols
δ is a transition function of ε-NFA, $(\delta : Q \times \Sigma \cup \{\varepsilon\} \to 2^Q)$
q_0 is a start state of ε-NFA, $(q_0 \in Q)$
F is the set of final states of ε-NFA, $(F \subseteq Q)$

2.4.2 Conversion from ε-NFA to NFA

ε-NFA = (Q, Σ, δ, q_0, F) and NFA = (Q, Σ, δ′, q_0, F′)

Every ε-NFA can be converted to the NFA as following:

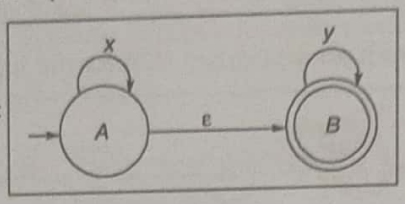
1. Initial state of ε-NFA is same as that of NFA
2. Transition function of ε-NFA: δ′(q, a) = ε-closure (δ(ε-closure (q), a)).
   where δ′ represents transition function of NFA and δ is ε-NFA transition function.

ε-closure (q) = the set of all the states which are reachable from state q by reading only ε.
(q is always reachable from q, so ε-closure (q) includes q)

3. The number of states in ε-NFA is same as NFA.
4. In NFA make all states as final where ε-closure of that state contain a final state of ε-NFA.

Example - 2.86 Construct an equivalent NFA for the following ε-NFA.

ε-NFA :



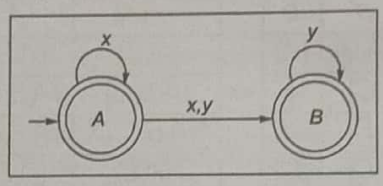
Solution:

ε-closure(A) = {A, B}, ε-closure(B) = {B}

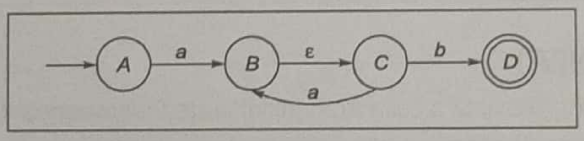
Initial state A of NFA is same as initial state of ε-NFA.

Final states in NFA are both A and B, their ε-closures contain final state B of ε-NFA.

NFA :

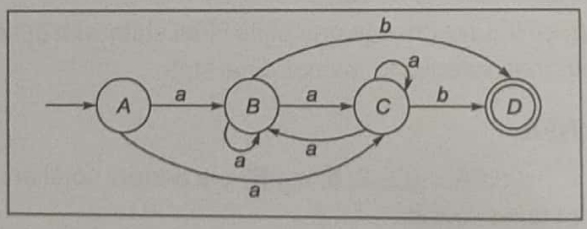


Example - 2.87 Construct an equivalent NFA for the following ε-NFA.

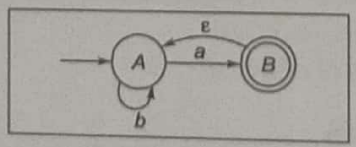


Solution:

NFA :

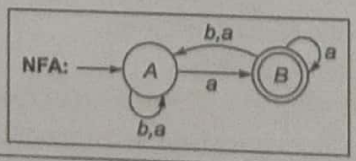


Example - 2.88 Construct an equivalent NFA for the following ε-NFA.

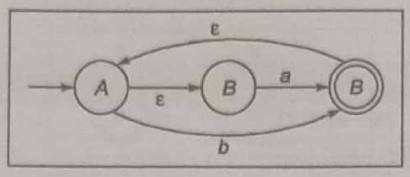


Solution:

NFA:

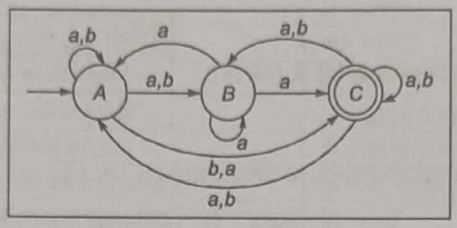


Example - 2.89 Construct an equivalent NFA for the following ε-NFA.



Solution:

NFA :



2.4.3 Conversion from ε-NFA to DFA

ε-NFA = $(Q, \Sigma, \delta, q_0, F)$ and DFA = $(2^Q, \Sigma, \delta', \varepsilon\text{-closure}(q_0), F')$

Every ε-NFA can be converted to DFA as following:

(a) Initial state of DFA is "ε-closure of initial state of ε-NFA"
Let q_0 is an initial state of ε-NFA. Then ε-closure (q_0) is initial state of DFA.

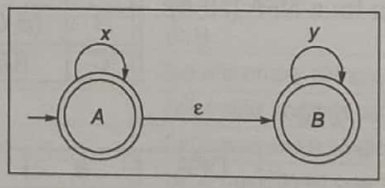
(b) Transition function of DFA: **$\delta'(q, a) = \varepsilon\text{-closure}(\delta(q, a))$.**
where δ' and δ are transition functions of DFA and ε-NFA respectively.

(c) If any state of DFA contain final state of ε-NFA, make that state as a final state in DFA.

(d) ε-NFA to DFA conversion also possible with the help of NFA. Convert first ε-NFA to NFA, then convert NFA to DFA.

Example - 2.90 Construct an equivalent DFA for the following ε-NFA.

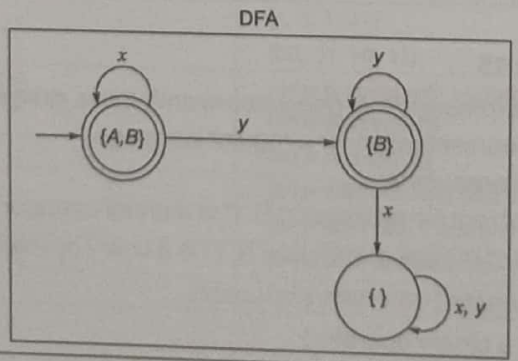
ε-NFA :



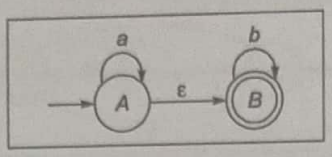
Solution:

ε-closure(A) = {A, B} and ε-closure(B) = {B}
Initial state of DFA is ε-closure(A) = {A, B}.

DFA



Example - 2.91 Construct an equivalent DFA for the following ε-NFA.



Solution:

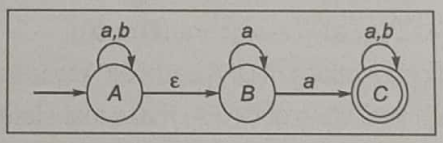
Indirect transition table for ε-NFA (NFA):

NFA	a	b
→ *A	{A, B}	{B}
*B	ϕ	{B}

Initial state of DFA is ε-closure(A), ε-closure(A) = {A, B}

DFA	a	b
→ *{A,B}	{A, B}	{B}
*{B}	ϕ	{B}
ϕ	ϕ	ϕ

Example - 2.92 Construct an equivalent DFA for the following ε-NFA.



Solution:

Indirect transition table for ε-NFA (NFA):

NFA	a	b
→ A	{A, B, C}	{A,B}
B	{B, C}	ϕ
*C	{C}	{C}

ε-closure(A) = {A, B}

DFA	a	b
→ {A,B}	{A, B,C}	{A,B}
*{A,B,C}	{A, B,C}	{A,B,C}

2.5 Regular Expressions

Regular expression is an expression that generates exactly those strings which are in a regular language. It is a declarative way of representation for a regular language.

Operators of Regular Expressions:

- R* is *kleene closure* of regular expression R. (* is a unary operator)
- R+ is *positive closure* of regular expression R. (+ is a unary operator)
- · is *concatenation* operator. (· is a binary operator)
- + is *or* operator. (+ is a binary operator)

2.5.1 Equivalence between Regular Languages and Regular Expressions

	Regular language Over $\Sigma = \{a, b\}$	Regular Expression
1	L = {All strings of a's and b's}	$(a + b)^* = \Sigma^*$
2	L = {All strings start with a}	$a(a + b)^*$
3	L = { }	ϕ
4	L = {All strings ends with b}	$(a + b)^*b$
5	L = {All strings start with a and end with b}	$a(a + b)^*b$
6	L = {All strings with atmost two length}	$(a + b + \varepsilon)\ (a + b + \varepsilon)$
7	L = {All strings with atleast two length}	$(a + b)\ (a + b)\ (a + b)^* = (a + b)(a + b)^+$
8	L = {All strings with exactly two length}	$(a + b)\ (a + b)$
9	L = {All strings containing 'ab' as a substring}	$(a + b)^*\ ab(a + b)^*$
10	L = {In all strings, every 'a' must followed by 'b'}	$(b + ab)^*$
11	L = {All strings contain second symbol from LHS is 'a'}	$(a + b)\ a\ (a + b)^*$
12	L = {All strings contain second symbol from RHS is 'a'}	$(a + b)^*\ a\ (a + b)$
13	L = {All strings of even length}	$((a + b)\ (a + b))^*$
14	L = {All strings of odd length}	$(a + b)\ ((a + b)\ (a + b))^*$
15	L = {Every string contain all a's followed by b's}	$a^*\ b^*$

2.5.2 Equivalence between Regular Expressions and Regular Sets

	Regular Expression	Regular Set Over $\Sigma = \{0, 1\}$
1	ϕ	{ }
2	ε	{ε}
3	0	{0}
4	0 + 1	{0,1}
5	$\varepsilon + \Sigma^* 0 = \varepsilon + (0+1)^* 0$	{w| w is empty string or ends with 0}
6	$0\Sigma^*0 + 1\Sigma^* 1 + 0 + 1$	{w| w start and end with same symbol}
7	0^*10^*	{w| w contains a single one}
8	$\Sigma^*1\ \Sigma^*$	{w| w has atleast one 1} or {w| w has substring '1'}
9	$\Sigma^*00\ \Sigma^*$	{w| w contains '00' as a substring}
10	$(1 + 01)^*$	{w| every 0 in w is followed by atleast 1}
11	$(\Sigma\ \Sigma)^*$	{w| w is even length string}
12	$(0 + \varepsilon)\ (1 + \varepsilon)$	{ε, 0, 1, 01}
13	$(0 + 1)\ (0 + 1)$	{00, 01, 10, 11}
14	$(0 + 1 + \varepsilon)\ (0 + 1 + \varepsilon)$	{w| w is string with atmost 2 length}
15	$0(0 + 1)^*$	{w| w start with 0}
16	$(0 + 1)^*0$	{w| w end with 0}
17	$0(0 + 1)^*0 + 0$	{w| w start and end with 0}
18	$1(0 + 1)^*0$	{w| w start with 1 and end with 0}
19	$1(0 + 1)^*0 + 0(0 + 1)^*1$	{w| w start and end with different symbols}
20	$1(0 + 1)^*00$	{w| w start with 1 and end with 00}

2.5.3 Properties of Regular Expressions using Operators

(a) Union operator satisfies commutative property and associative property.
- $a + b = b + a$ (commutative)
- $a + (b + c) = (a + b) + c$ (associative)

(b) The concatenation operator satisfies associative property but not commutative property
- $a.b \neq b.a$ (not commutative)
- $a.(b.c) \equiv (a.b).c$ (associative)

(c) Both left and right distributive properties of concatenation over union are holds.
- $a.(b + c) \equiv (a.b) + (a.c)$ (Left distribution of. over +)
- $(a + b).c \equiv (a.c) + (b.c)$ (Right distribution of. over +)

(d) Both left and right distributive properties of union over concatenation are not holds.
- $a + (b.c) \neq (a + b).(a + c)$
- $(a.b) + c \neq (a + c).(b + c)$

(e) Union operator satisfies idempotent property but the concatenation operator does not holds.
- $a + a = a$ (idempotent)
- $a.a \neq a$ (not idempotent)

(f) Identity property:
- $R + \phi = \phi + R = R$ (ϕ is identity element with respect to union operation)
- $\varepsilon.R = R.\varepsilon = R$ (ε is identity element with respect to concatenation)

(g) Annihilator property: $R \circ X = X$
- $R + \Sigma^* = \Sigma^*$ (Σ^* is annihilator with respect to union operator)
- $R.\phi = \phi$ (ϕ is annihilator with respect to concatenation operator)

2.5.4 Equivalences of Languages (or Regular Expressions)

Let r_1, r_2 and r_3 are regular expressions.

(a) $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
(b) $L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$
(c) $L(r^*) = (L(r))^*$
(d) $r_1 \cdot r_2 \neq r_2 \cdot r_1$
(e) $r_1(r_2 r_3) = (r_1 r_2) r_3$
(f) $r_1(r_2 + r_3) = r_1 r_2 + r_1 r_3$
(g) $r_1 + r_2 r_3 \neq (r_1 + r_2)(r_1 + r_3)$
(h) $\phi + r = r$
(i) $\phi.r = r.\phi = r^*.\phi = \phi.\phi = \phi^+ = \phi$
(j) $\varepsilon r = r\varepsilon = r$
(k) $\varepsilon^* = \varepsilon = \varepsilon^+$
(l) $\phi^* = \varepsilon$
(m) $r + r = r$
(n) $r \cdot r \neq r$
(o) $r^*.r^* = r^*$
(p) $r^* \cdot r^+ = r^+$
(q) $(r^*)^* = (r^*)^+ = (r^+)^* = r^*$
(r) $(r)^* = r^*$
(s) $(\varepsilon + rr^*) = r^* = (\varepsilon + r^+)$
(t) $p(qp)^* = (pq)^* p$
(u) $(p + q)^* = (p^* q^*)^* = (p^* + q^*)^*$
(v) $(p + q)^* p^* q^* = (p + q)^*$

Remember

Consider the following regular expressions are over the symbols *a* and *b*.
- Regular expression R that generate all strings where each string start and end with same symbol

  $R = a + b + a\,(a + b)^*\,a + b\,(a + b)^*\,b$
- Regular expression R that generate all strings where each string start with 'a' and not having two consecutive b's. $R = (a + ab)^+$

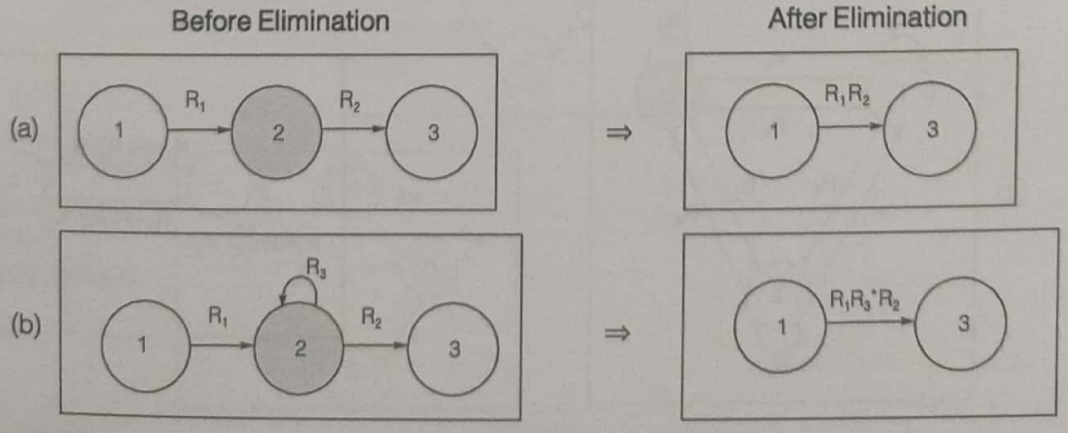
- Regular expression R that generate all strings where length of the string is exactly '3'
  $R = (a + b)(a + b)(a + b)$
- Regular expression R that generate all strings where the length of the string is atleast '3'
  $R = (a + b)(a + b)(a + b)^+$
- Regular expression R that generate all strings where the length of the string is atmost '3'
  $R = (a + b + \varepsilon)(a + b + \varepsilon)(a + b + \varepsilon)$
- Regular expression R that generate all strings where the length of the string is even
  $R = ((a + b)(a + b))^*$
- Regular expression R that generate all strings where the length of the string is divisible by 3.
  $R = ((a + b)(a + b)(a + b))^*$
- Regular expression R that generate all strings where the length of the string is odd
  $R = ((a+b)(a + b))^*.(a + b)$
- Regular expression R that generate all strings where each string contain exactly one 'b'. $R = a^* b a^*$
- Regular expression R that generate all strings where each string contain exactly two b's. $R = a^* ba^* ba^*$
- Regular expression R that generate all strings where each string contain atmost two b's.
  $R = a^* (b + \varepsilon) a^* (b + \varepsilon) a^*$
- Regular expression R that generate all strings
  $R = (a + b)^* = (a + b + \varepsilon)^+ = (a^*b^*)^+ = (b^*a^*)^+ = (a^* + b)^+ = (a + b^*)^+$

2.6 Equivalence between Finite Automata and Regular Expressions

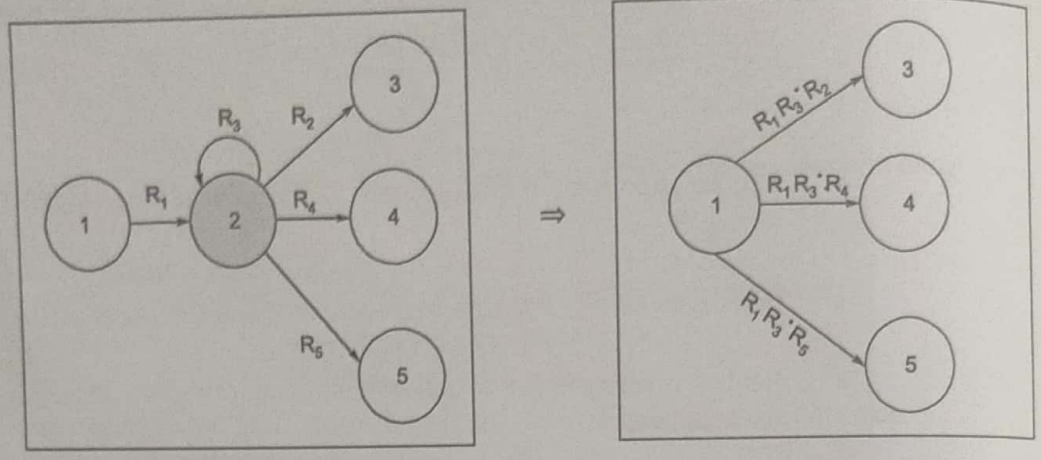
2.6.1 Conversion from Finite Automata to Regular Expressions

(A) State Elimination Method

1. More than one initial state is invalid. So simplify the transition graph such that it contain exactly one initial state and exactly one final state.
2. Simplify the transition graph such that it contain different states for initial and final.
3. Eliminate all the states except initial state and final state by forming regular expressions.
4. **Elimination of state 2:**



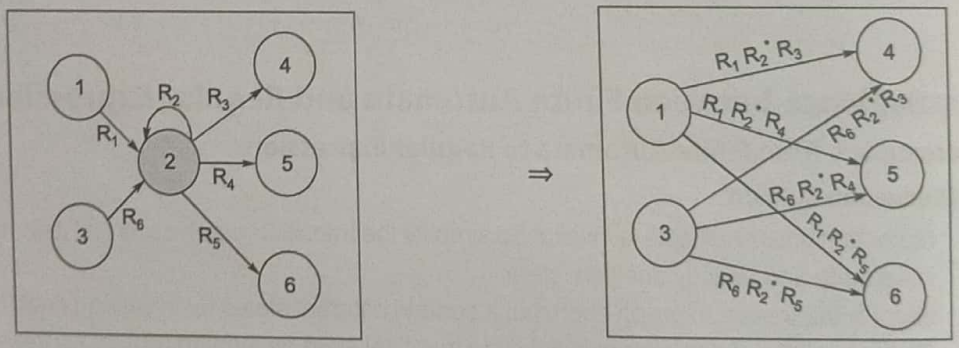
(c)



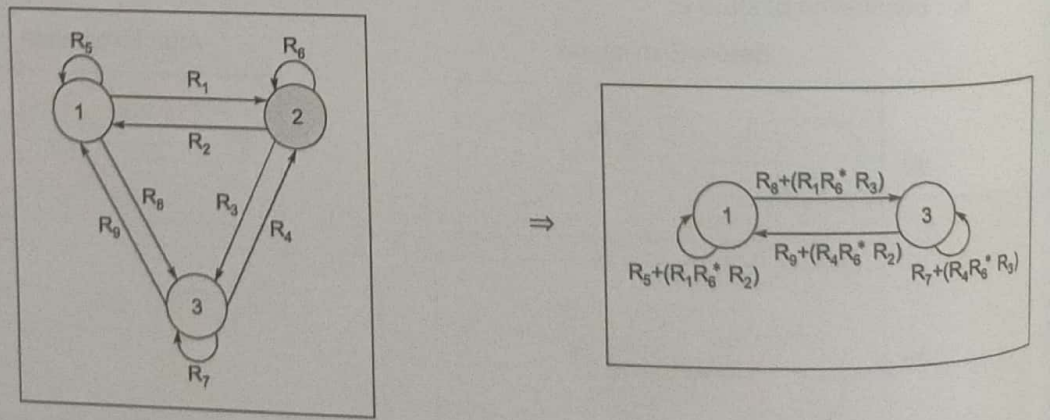
(d)

(e)

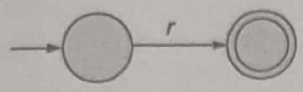
(f)



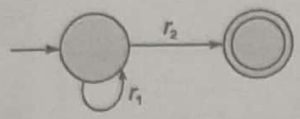
(g)



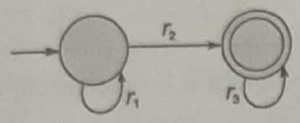
5. Continue the state elimination until transition graph is converted to any one of the following with 2 states.

(a) 

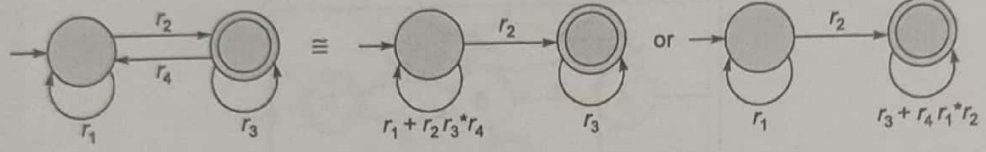
$R = r$

(b) 

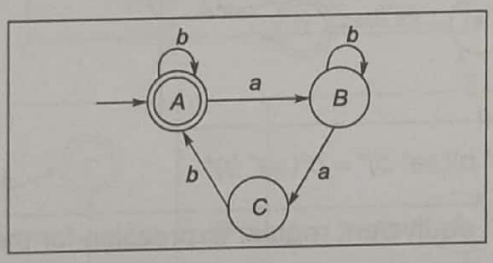
$R = r_1^* r_2$

(c) 

$R = r_1^* r_2 r_3^*$

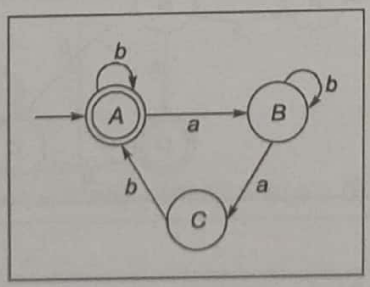
(d) 

Example - 2.93 Find an equivalent regular expression for the following finite automata.

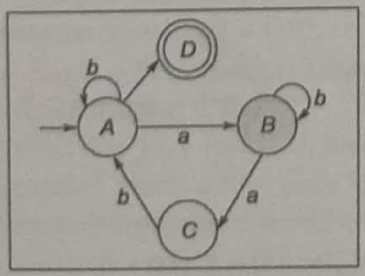


Solution:

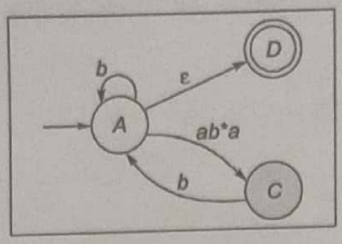
Keep initial and final states separately



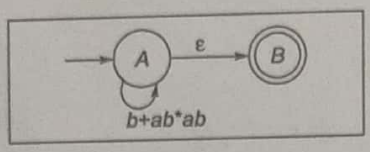
Delete state C:



Delete state B:

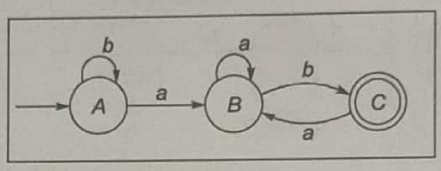


Delete state C:



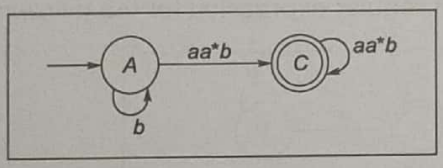
Regular expression = $(b + ab^* ab)^* \varepsilon = (b + ab^* ab)^*$

Example - 2.94 Find an equivalent regular expression for the following finite automata.



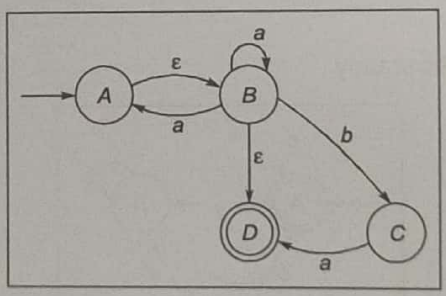
Solution:

Delete state B:

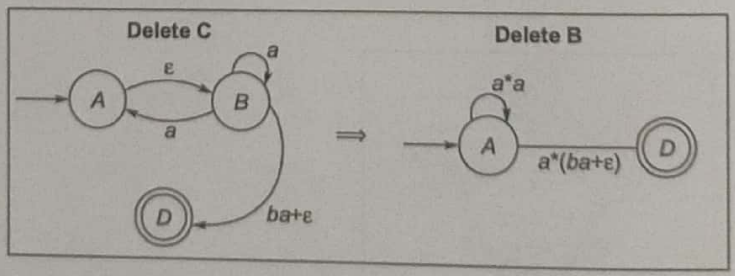


Regular expression = $b^*(aa^* b)(aa^* b)^* = b^*(aa^* b)^+$

Example - 2.95 Find an equivalent regular expression for the following finite automata.

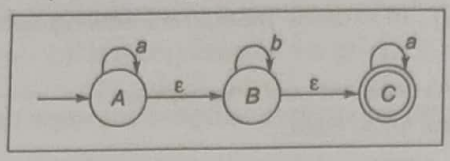


Solution:



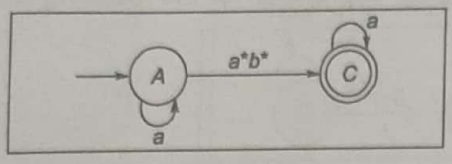
Regular expression = $(a^*a)^*a^* (ba + \varepsilon) = (a^+)^*a^* (ba + \varepsilon) = a^* (ba + \varepsilon)$

Example - 2.96 Find an equivalent regular expression for the following finite automata.



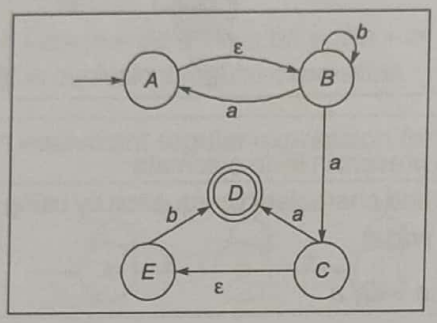
Solution:

Delete B:

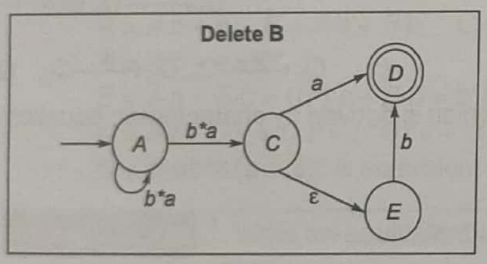


Regular expression = $a^*a^*b^*a^* = a^*b^*a^*$

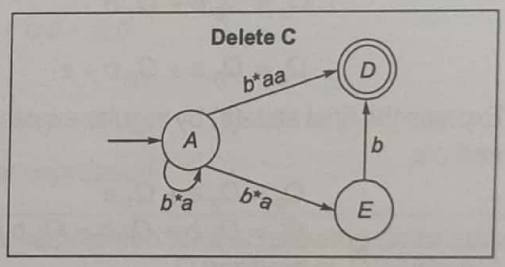
Example - 2.97 Find an equivalent regular expression for the following finite automata.



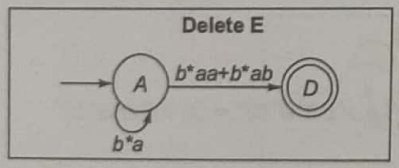
Solution:



⇒



⇒



Regular expression = $(b^*a)^*\,(b^*aa + b^*ab) = (b^*a)^*b^*a\,(a + b) = (b^*a)^+\,(a + b)$

(B) Arden's Method

If P and Q are two regular expressions over an alphabet Σ and P does not contain ε then the equation R = Q + RP has a unique solution given by R = QP*.

Note: The equation R = Q + PR, although it does not occur while converting FA to regular expression, can still be solved and the solution is R = P*Q.

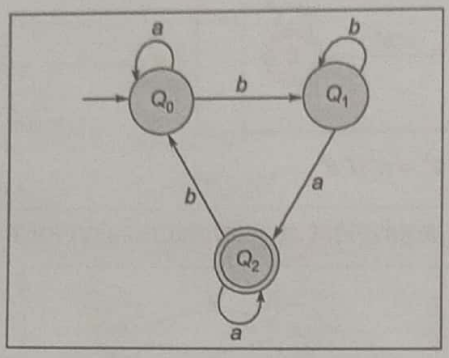
Example: The equation R = 10 + R(00+1), has a solution which is R = 10 (00+1)* and the equation R = 10 + (00+1) R, has a solution which is R = (00+1)*10.

Note: If P contains ε then the equations R = Q + RP as well as R = Q + PR, both have infinite number of solutions.

We use Arden's Theorem to find the regular expression recognized by transition systems. NFA and DFA are transition system. While applying the Arden's Theorems following assumptions are made regarding the transition system:

- The transition system (FA) does not contain ε-moves.
- The transition has only one initial state.

Example - 2.98 Write an equivalent regular expression for the following finite automata.



Solution:

(a) See that no ε-transition is present in finite automata

(b) For each and every state find characteristic equation by using direct transitions only

(c) For initial state equation, add ε

$$Q_2 = Q_2.a + Q_1.a \qquad (\because \; Q_2 \xleftarrow{a} Q_2, Q_2 \xleftarrow{a} Q_1)$$

$$Q_1 = Q_0.b + Q_1.b \qquad (\because \; Q_1 \xleftarrow{b} Q_1, Q_1 \xleftarrow{b} Q_0)$$

$$Q_0 = Q_0.a + Q_2.b + \varepsilon \qquad (\because \; Q_0 \xleftarrow{a} Q_0, Q_0 \xleftarrow{b} Q_2, \; Q_2 \leftarrow)$$

(d) Express the final state(s) by regular expression which is derived by characteristic equation of a's and b's.

$$Q_2 = Q_2.a + Q_1.a$$

$$Q_1 = Q_0.b + Q_1.b = Q_0\, b.b^* \qquad (\because \; R = Q + RP = Q.P^*)$$

Substitute Q_1 in equation Q_2

$$Q_2 = Q_2.a + (Q_0\, bb^*a) = Q_0\, b.b^*aa^*$$

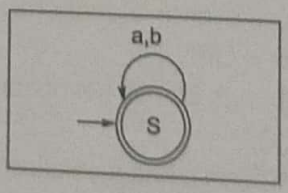
Substitute Q_2 in equation Q_0

$$Q_0 = Q_0.a + (Q_0.bb^*a)a^*\, b + \varepsilon = Q_0\, (a + bb^*a^*b)^* = (a + bb^*a^*b)^*$$

Substitute Q_0 in equation Q_2

$Q_2 = (a + bb^*\, a^*\, b)^*\, bb^*\, aa^*$ is equivalent regular expression for the given FA.

Example - 2.99 Write an equivalent regular expression for the following finite automata.



Solution:

S = Sa + Sb + ε (ε is included because S is initial state)

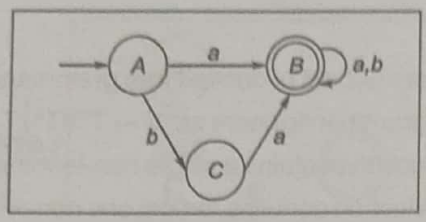
S = ε + (a + b)S

$S = \varepsilon(a + b)^*$ (Using Arden's equation $R = Q + RP$ and solution $R = QP^*$)

$S = (a + b)^*$ (S is final expression because S is final state)

So, the regular expression $(a + b)^*$ is recognized by the given DFA.

Example-2.100 Write an equivalent regular expression for the following finite automata.



Solution:

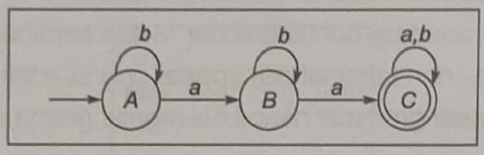
$A = \varepsilon$

$C = A \cdot b \quad [\because A = \varepsilon] \quad \Rightarrow \quad C = b$

$B = A \cdot a + Ba + Bb + Ca = a + Ba + Bb + ba = B(a + b) + ba + a$

$B = (ba + a)(a + b)^*$ is equivalent regular expression.

Example-2.101 Write an equivalent regular expression for the following finite automata.



Solution:

$A = A \cdot b + \varepsilon, \quad B = A \cdot a + B \cdot b, \quad C = B \cdot a + c \cdot a + c \cdot b$

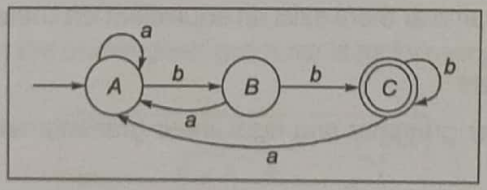
$A = A \cdot b + \varepsilon = b^*$

$B = A \cdot a + B \cdot b = (b^*) \cdot a + B \cdot b = b^*ab^*$

$C = Ba + C \cdot a + C \cdot b = b^*ab^*a + c(a + b)$

$C = b^*ab^*a\,(a + b)^*$ is equivalent regular expression.

Example-2.102 Write an equivalent regular expression for the following finite automata.



Solution:

$A = \varepsilon + A \cdot a + B \cdot a + C \cdot a, \quad B = A.b, \quad C = B \cdot b + C \cdot b$

$\Rightarrow \quad C = B \cdot b + C \cdot b = Abb + cb$

$\Rightarrow \quad A = \varepsilon + Aa + Ba + Ca = \varepsilon + Aa + Aba + Ca = \varepsilon + Aa + Aba + Abba + Cba$

$= (\varepsilon + Cba) + A(a + ba + bba)$

$\Rightarrow \quad C = Abb + Cb = (\varepsilon + Cba)(a + ba + bba)^*bb + Cb$

$= (a + ba + bba)^*bb + Cba(a + ba + bba)^*bb + Cb$

$= (a + ba + bba)^*bb + C(ba(a + ba + bba)^*bb + b)$

$= [(a + ba + bba)^*bb]\,(ba(a + ba + bba)^*bb + b)^*$

2.7 Regular Grammar

All productions of regular grammar are either left linear productions or right linear productions. Regular grammar is equivalent to left linear grammar and right linear grammar. A grammar is regular grammar iff it is left linear grammar or right linear grammar.

2.7.1 Linear Grammar

- Class of linear grammars are subset of context free grammars and super set of regular grammars.
- Each production of linear grammar appears as: $V \rightarrow T^*VT^* | T^*$
- Left hand side of the production contains a single non-terminal.
- Right hand side of the production contains atmost one non-terminal
- $S \rightarrow aSb|aS|Sb|a|b$ is linear grammar.

2.7.2 Left Linear Grammar (LLG)

- Class of left Linear grammars are subset of linear grammars.
- Each production of Left Linear grammar appears as: $V \rightarrow VT^* | T^*$ where V is a variable and T is terminal.
- Left hand side of the production contains a non-terminal.
- Right hand side of the production contains either "*a non-terminal followed by terminal sequence*" or "*terminal sequence*". i.e., non-terminal can appear only as a left most symbol.
- $S \rightarrow Sa|Sb|a|b$ is left linear grammar hence it is regular grammar.

2.7.3 Right Linear Grammar (RLG)

- Class of right linear grammars are subset of linear grammars.
- Each production or right linear grammar appears as: $V \rightarrow T^*V | T^*$
- Left hand side of the production contains a non-terminal.
- Right hand side of the production contains either "*terminal sequence followed by a non-terminal*" or "*terminal sequence*". i.e., non-terminal can appear only as a right most symbol.
- $S \rightarrow aS|bS|a|b$ is right linear grammar.
- For every right linear grammar there exist an equivalent left linear grammar and vice-versa.

2.7.4 Equivalence of Grammars

Regular grammar, left linear grammar and right linear grammar are always possible to generate the strings of given regular language.

- Regular grammars ≡ Left linear grammars ≡ Right linear grammars.
- All these three grammars generate same class of language called as regular languages.

2.7.5 Conversion from RLG to Finite Automata

Steps to convert RLG to finite automata

1. Starting state is named with starting symbol of given grammar
2. Final state is named with ε
3. Let G = (V, T, P, S) be the given RLG. Then we can construct an equivalent ε-NFA as M = (Q, T, δ, [S], {[ε]})

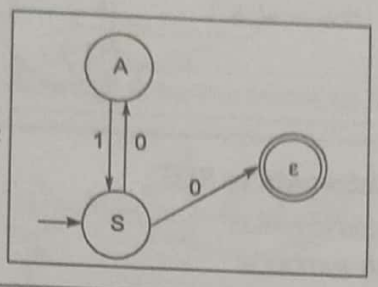
Example - 2.103 Construct an equivalent finite automata for the following RLG.

$$S \rightarrow 01S|0$$

Solution:

NFA can be constructed for the above given grammar is as following:

NFA :

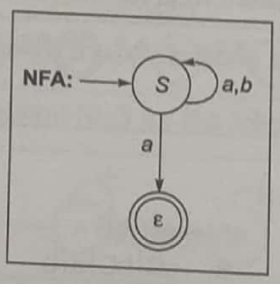


Example - 2.104 Construct an equivalent finite automata for the following RLG.

$$S \rightarrow aS|bS|a$$

Solution:

NFA can be constructed for the above given grammar is as following:

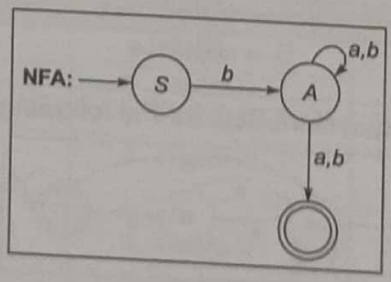


Example - 2.105 Construct an equivalent finite automata for the following RLG.

$$S \rightarrow bA$$
$$A \rightarrow aA|bA|a|b$$

Solution:

NFA can be constructed for the above given grammar is as following:

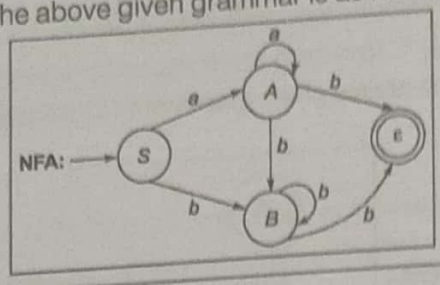


Example - 2.106 Construct an equivalent finite automata for the following RLG.

$$S \rightarrow aA|aB$$
$$A \rightarrow aA|bB|b$$
$$B \rightarrow bB|b$$

Solution:

NFA can be constructed for the above given grammar is as following:

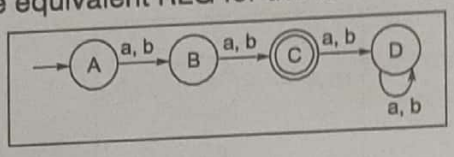


2.7.6 Conversion from Finite Automata to RLG

Steps to convert finite automata to a RLG:

1. Initial state acts as a start symbol.
2. If $\delta(A, a) = B$, then
   - $A \rightarrow aB$ is a production if B is not final state.
   - $A \rightarrow aB \mid a$ is a production if B is final state.
3. If initial and final states are same then make them separate.

Example-2.107 Find the equivalent RLG for the following finite automata.



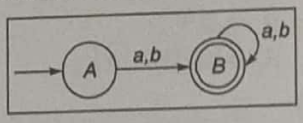
Solution:

The equivalent right linear grammar is:

$A \rightarrow aB \mid bB$

$B \rightarrow aC \mid bC \mid a \mid b$

Example-2.108 Find the equivalent RLG for the following finite automata.



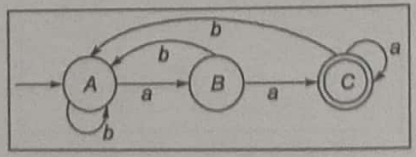
Solution:

The equivalent right linear grammar is:

$A \rightarrow aB \mid bB$

$B \rightarrow aB \mid bB \mid \in$

Example-2.109 Find the equivalent RLG for the following finite automata.



Solution:

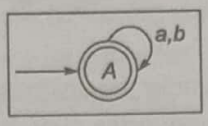
The equivalent right linear grammar is:

$A \rightarrow aB \mid bA$

$B \rightarrow bA \mid aC$

$C \rightarrow aC \mid bA \mid \in$

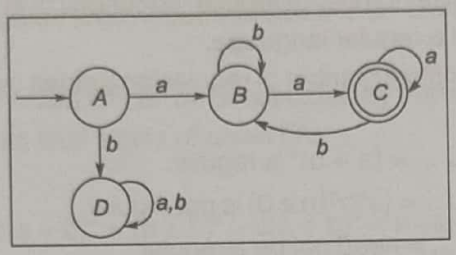
Example - 2.110 Find the equivalent RLG for the following finite automata.



Solution:

$A \rightarrow aA|bA|\epsilon$

Example - 2.111 Find the equivalent RLG for the following finite automata.

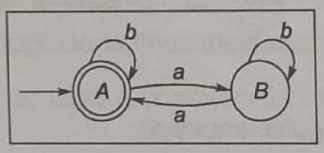


Solution:

$A \rightarrow aB$
$B \rightarrow bB|aC$
$C \rightarrow aC|bB|\epsilon$

D is useless symbol, so no need to write the productions of D.

Example - 2.112 Find the equivalent RLG for the following finite automata.



Solution:

$A \rightarrow bA|aB|\epsilon$
$B \rightarrow aA|bB$

2.7.7 Equivalence between Regular Expressions and Regular Grammars

S.No.	Regular Expression	Left Linear Grammar	Right Linear Grammar
1	a^*	$S \rightarrow Sa\|\epsilon$	$S \rightarrow aS\|\epsilon$
2	$(a+b)^*$	$S \rightarrow Sa\|Sb\|\epsilon$	$S \rightarrow aS\|bS\|\epsilon$
3	$(a+b)^+$	$S \rightarrow Sa\|Sb\|a\|b$	$S \rightarrow aS\|bS\|a\|b$
4	$a.(a+b)^*$	$S \rightarrow Sa\|Sb\|a$	$S \rightarrow aA$ $A \rightarrow aA\|bA\|\epsilon$
5	$(a+b)^*b$	$S \rightarrow Ab$ $A \rightarrow Aa\|Ab\|\epsilon$	$S \rightarrow aS\|bS\|b$
6	a^*b^*	$S \rightarrow Aa\|Bb\|\epsilon$ $A \rightarrow Aa\|\epsilon$ $B \rightarrow Bb\|A\|\epsilon$	$S \rightarrow aA\|bB\|\epsilon$ $A \rightarrow aA\|B\|\epsilon$ $B \rightarrow bB\|\epsilon$
7	$(ab)^*$	$S \rightarrow Sab\|\epsilon$	$S \rightarrow abS\|\epsilon$
8	$a(a+b)^*b$	$S \rightarrow Ab$ $A \rightarrow Aa\|Ab\|a$	$S \rightarrow aA$ $A \rightarrow aA\|bA\|\epsilon$

2.8 Closure Properties of Regular Languages

2.8.1 Union of Two Regular Languages

The set of regular languages is closed under the union operation.

- If L1 and L2 are any two regular languages over the same alphabet, then (L1 ∪ L2) is a regular language.

  Example: L1 = a(a + b)* and L2 = b(a + b)*.

  L1 ∪ L2 = a(a + b)* + b(a+ b)* = (a + b) (a + b)* = (a + b)+ is a regular language.
- **Finite Union:** Union of finite number of regular languages over the same alphabet is also a regular.

  L1 ∪ L2 ∪... Ln is always a regular language.
- **Infinite Union:** Union of infinite number of regular languages over the same alphabet is need not be a regular language.

  (*i*) (a + b)* ∪ L1 ∪ L2 ∪ ... = (a + b)* is regular.

  (*ii*) $a^0b^0 \cup a^1b^1 \cup a^2b^2 \cup ... = \{a^nb^n | n \geq 0\}$ is not regular.

  Therefore, L1 ∪ L2 ∪ L3 ... = need not be a regular.

2.8.2 Intersection of Two Regular Languages

The set of regular languages is closed under the intersection operation.

- If L1 and L2 are any two regular languages over the same alphabet, then L1 ∩ L2 is a regular language.

  Example: L1 = (a + b)* and L2 = b(a + b)*.

  L1 ∩ L2 = (a + b)* ∩ b(a + b)* = b(a + b)* is a regular language.
- **Finite Intersection:** Intersection of finite number of regular languages over the same alphabet is a regular language.

  L1 ∩ L2 ∩ ... Ln is always a regular language.
- **Infinite Intersection:** Intersection of infinite number of regular languages over the same alphabet is need not be a regular language.

  (*i*) { } ∩ L1 ∩ L2 ∩... = { } = ϕ is regular.

  (*ii*) $(\Sigma^* - a^0b^0) \cap (\Sigma^* - a^1b^1) \cap (\Sigma^* - a^2b^2) ... = (\Sigma^* - a^nb^n)$ is not a regular language.

  Therefore, L1 ∩ L2 ∩ L3... = need not be a regular.

2.8.3 Difference of Two Regular Languages

The set of regular languages is closed under the difference operation.

- If L1 and L2 are any two regular languages over the same alphabet, then L1 – L2 is a regular language.

  Example: L1 = (a + b)+ and L2 = b(a + b)*.

  L1 – L2 = (a + b)+ – b(a + b)* = a(a + b)* is a regular language.
- **Finite Difference:** Difference of infinite number of regular languages over the same alphabet is a regular language.

  L1 – L2 –... Ln is always a regular language.
- **Infinite Difference:** Difference of infinite number of regular languages over the same alphabet is need not be a regular language.

(*i*) { } – L1 – L2 –... = { } is regular

(*ii*) $\Sigma^* - (a^0b^0) - (a^1b^1) - (a^2b^2) - \ldots = (\Sigma^* - a^nb^n)$ is not regular.

Therefore, L1 – L2 – L3 ... = need not be a regular.

- Union, Intersection and Difference operations for given finite number of regular languages can be proved by constructing *compound FA*.

4 Complement of a Regular Language

The set of regular languages is closed under the complement operation.

- If L is any regular language over input alphabet, then Complement(L) is a regular language. Complement(L) = $\Sigma^* - L$.
- Construction of complemented FA can be constructed by making final states as non-final states and making non-final states as final states of given FA.

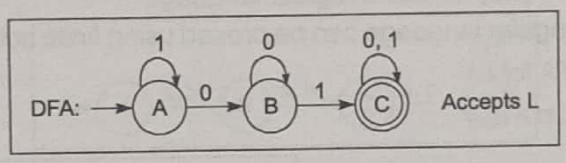
  Example: L = b(a + b)*.

  Complement(L) = $\Sigma^* - b(a + b)^* = (a + b)^* - b(a + b)^* = \varepsilon + a(a + b)^*$ is a regular language.
- **Complementation of DFA:**

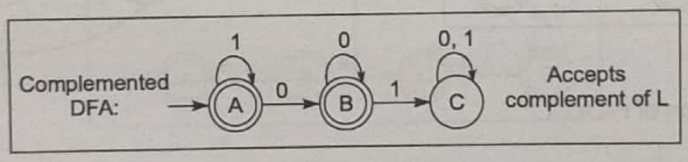
  (*i*) Construction of DFA for a Language L.

  Let L be a set of all binary strings which contain 01.

  L = (0+1)* 01 (0+1)*



  (*ii*) Construction of DFA that accepts a complement of L: Interchange final states and non-final states of given DFA.



8.5 Concatenation of Two Regular Languages

The set of regular languages is closed under the concatenation operation.

- If L1 and L2 are any two regular languages over the same alphabet, then L1.L2 is a regular language.

  Example: L1 = a(a + b)* and L2 = (a + b)*

  L1 · L2 = a(a + b)*· (a + b)* is a regular language.
- **Finite Concatenation:** Concatenation of finite number of regular languages over the same alphabet is a regular.

  L1 · L2 ... Ln is a regular language.
- **Infinite Concatenation:** Concatenation of infinite number of regular languages over the same alphabet is need not be a regular language.

  (*i*) { } · L1 · L2· ... = Empty language is a regular.

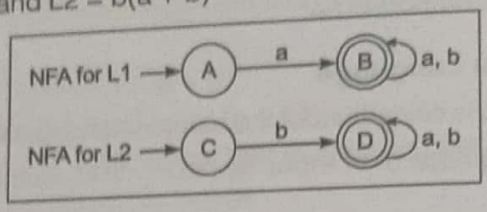
  (*ii*) {a} · {aa} · {aaa}· ... = $\{a^{\infty}\}$ is non-regular

Therefore, L1 · L2 · L3... = need not be a regular.

- Concatenation of two regular languages

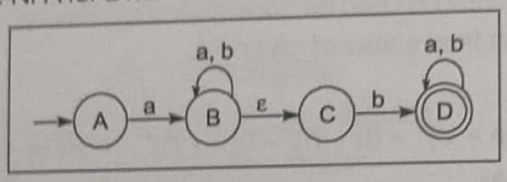
(*i*) Construction of NFA for L1 and L2:

Let L1 = a(a + b)* and L2 = b(a + b)*



(*ii*) Construction of ε-NFA for L1.L2

ε-NFA for L1·L2:



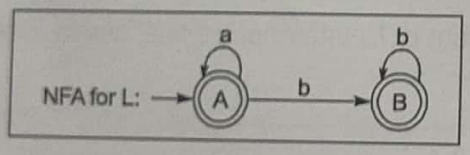
2.8.6 Kleene Closure of a Regular Language

The set of regular languages is closed under the kleene star operation.

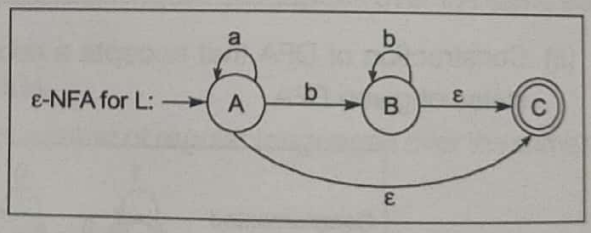
- Kleene closure of a regular language is a regular language.
  Let L = {ab}. Then L* = (ab)* is also a regular language.
- Kleene closure of a regular language can be proved using finite automata construction.

(*i*) Construction of FA for L:

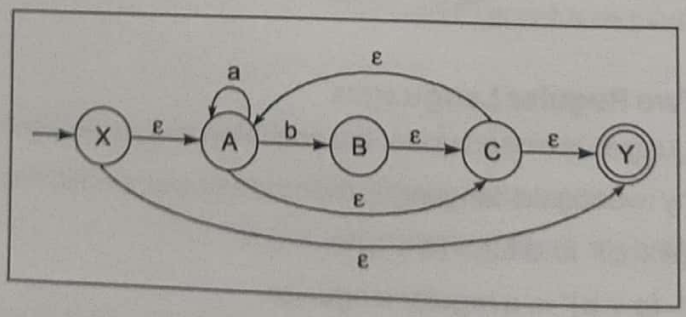
Let L = $\{a^m b^n | m, n \geq 0\}$



or



(*ii*) Construction of FA for L*:



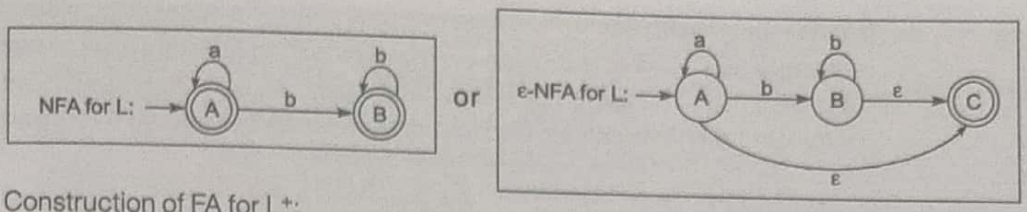
2.8.7 Positive Closure of a Regular Language

The set of regular languages is closed under the positive closure operation.

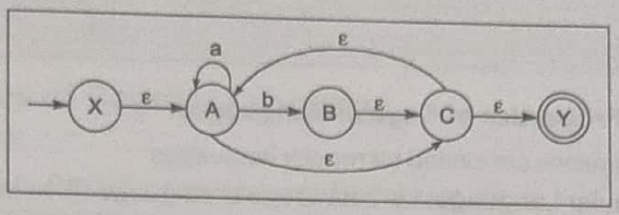
- Positive closure of a regular language is a regular language.
  Let L = {ab}. Then L+ = (ab)+ is also a regular language.
- Kleene closure of a regular language can be proved using finite automata construction.

(*i*) Construction of FA for L:

Let L = $\{a^m b^n | m, n \geq 0\}$



(*ii*) Construction of FA for L^+:



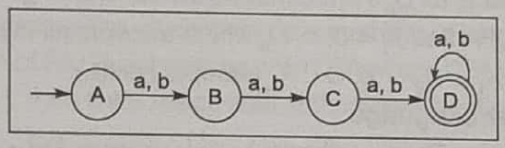
2.8.8 Prefix of a Regular Language

The set of regular languages is closed under the prefix operation.

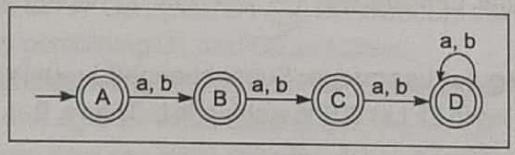
- Prefix(L) = {u | for some v, uv is in L, u, v ∈ Σ^*} = L/Σ^*
- Let L = {abcd}. Then Prefix(L) = {ε, a, ab, abc, abcd} is a regular.
- Prefix of a regular language using finite automata construction.

(*i*) Construction of FA for L:

Let L = set of all binary strings of atleast three length.



(*ii*) Construction of FA for Prefix(L) using FA of L: Make all the states as final states which can reach to the final state in the above FA to accept all the prefixes of given L.



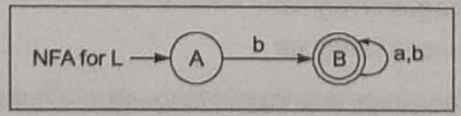
2.8.9 Reversal of a Regular Language

The set of regular languages is closed under the reversal operation.

- Let L = set of all strings which starts with 'b' over an input alphabet Σ = {a, b}
  Reversal(L) = (a + b)* b is also a regular.
- Reversal of a regular language using construction of finite automata

(*i*) Construction of FA for L

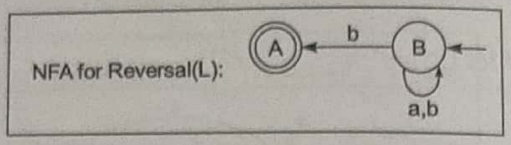
Let L = b (a + b)*



(*ii*) Construction of finite automata for reversal (L) using FA of L.

(a) Reversal of finite automata can be constructed by interchanging final and initial states, and by reversing the transition directions from the given finite automata.

(b) If finite automata contains more than one final state, it is possible to make one final state by creating a new state and then give ε-transitions from all final states to newly created final state and then change all previous final states to non-final states. Once epsilon NFA constructed then interchange final and initial states as well as reverse the transition directions.



2.8.10 Other Closure Properties of Regular Languages

The following operations are closed for regular languages:

- **Half of a Regular Language:** Let L be a regular language. Then Half(L) = $\{u \mid uv \in L \text{ and } |u| = |v|\}$. Construction of Half(L) for a given regular language L is as follows:
  (*i*) Construct DFA D that accepts L and states are $q_1, q_2 \ldots q_n$.
  (*ii*) There exist a string $w (w = uv)$ of length x which is acceptable in D.
  (*iii*) Let D_1 is same as the D but initial state is q_1, D_2 is same as D but initial state is q_2, and so on. If DFA D from state q on input a, it reaches to q', then D_i from state q on any input, it reaches to q'. i.e., $\delta(q, a) = q'$ in D $\Rightarrow \delta(q, \text{any}) = q'$.
  (*iv*) Let D_{11} is same as D but includes only q_1 as final state; D_{22} is same as D and includes only q_2 as final state, similarly all D_{ii}'s are constructed considering only q_i as final state.
  (*v*) For all i states of DFA find $X_i = D_i \cap D_{ii}$ which accepts all the half strings that reaches to q_i.
  (*vi*) Union of all X_i's = $X_1 \cup X_2 \cup X_3 \cup \ldots X_n$ accepts Half(L).
- **One third of a Regular Language:**
  Let L is a regular language. Then onethird (L) = $\{u \mid uvw \in L, |u| = |v| = |w|\}$
- **Quotient of two Regular Languages:** $\frac{L_1}{L_2} = \{x \mid xy \in L_1 \text{ for some } y \in L_2\}$.
- **Subsequence of a Regular Language:** Subsequence(L) = $\{w \mid w$ is obtained by removing symbols from anywhere of a string in L$\}$. Let 001 is a string in L. Then ε, 0, 1, 01, 001 are subsequences of the string 001.
- **Sub-word of a Regular Language:** Sub-word(L) = $\{v \mid$ for some u and for some w, uvw is in L$\}$
- **Suffix(L):** Suffix(L) = $\{v \mid$ for some u, uv is in L$\}$
- Homomorphism of a regular language is a regular
- Inverse Homomorphism of a regular language is a regular
- Substitution of a regular language is a regular
- **Shuffle:** Shuffle(L_1, L_2) = $\{x_1 y_1 x_2 y_2 \ldots x_k y_k \mid x_1 x_2 \ldots x_k$ is in L_1 and $y_1 y_2 \ldots y_k$ is in $L_2\}$
- Symmetric difference of two regular languages is regular
- NOR of two regular languages is regular
- COR of two regular languages is regular
- Square root of a regular language is a regular language. $\sqrt{L} = \{w \mid ww$ in a regular language L$\}$
- Max(L) is a Regular Language
- Min(L) = $\{w \mid w \in L$ and there is no proper prefix of w is in $L\}$
- Finite subset of any language L is always regular (finite language)

The following operations are not closed for regular languages:

(a) **Subset operation:** The subset of a regular language is need not be a regular language. $a(a+b)^* \subseteq (a+b)^*$ and $a^n b^n \subseteq (a+b)^*$.
(b) Infinite union of regular languages need not be a regular.
(c) Infinite intersection of regular languages need not be a regular.
(d) Infinite difference of regular languages need not be a regular.
(e) Infinite concatenation of regular languages need not be a regular.

2.9 Decision Properties of Regular Language

2.9.1 Emptiness (Is L Empty?)

- **Property:** Checking whether the given language is empty or not.
- **Proof by Construction of FA:** If there are no final states in the minimized FA, then the language accepted by FA is an empty.

2.9.2 Finiteness (Is L finite?)

- **Property:** Checking whether the given language is finite or not.
- **Proof by Construction of FA:** If there are no cycles and self loops in the minimized FA (excluding reject state), then the language accepted by FA is finite.

2.9.3 Membership (Is w ∈ L?)

- **Property:** Checking whether the given string is a member of language or not.
- **Proof by Construction of FA:** Run the string w on DFA where DFA accepts the given regular language L. If w is a member of L then it will reach final state otherwise it reaches non-final state.

2.9.4 Equivalence of two Regular Languages (Is L1 ≅ L2?)

- **Property:** Checking whether the given two regular languages are equal or not.
- **Proof by Construction of FA:** Construct D1 and D2 to accept the languages L1 and L2 respectively. Construct a new FA by combining D1 and D2 as follows:
  (a) Create a new initial state as (q_0, q_a) by combining both initial states of D1 and D2, where q_0 is an initial state of D1 and q_a is an initial state of D2.
  (b) Cover the transitions from the initial state for all input symbols using D1 and D2.
  (c) If any new state found while covering the transitions then cover all the transitions for the new state.
  (d) Repeat step (c) until there is no new state created.
  (e) If there exist a pair (p, q) as a state in the combined FA and one of its state is final and other is non-final (called as distinguishable pair), then the given two languages accepted by those D1 and D2 are not equal, hence given two languages are also not equal.
  (f) If there exist no distinguishable pair in the combined FA then those given two languages are equal because both D1 and D2 accepts the same language.

2.9.5 Subset (Is L1 ⊆ L2?)

- **Property:** Checking whether the given regular language L1 is subset of given regular language L2 or not.
- **Proof by Construction of FA:** Construct D1 and D2 to accept L1 and L2 respectively.
  If D1 ∩ Complement (D2) is accepts an empty language then L1 ⊆ L2, otherwise L1 is not a subset of L2. i.e., (L1 ⊆ L2) if and only if (L1 ∩ Complement (L2) = ϕ).

2.10 Moore and Mealy Machines

- Moore and Mealy machines are output generators.
- There is no final state concept in Moore and Mealy machines, because they are not used for recognizing languages.
- Moore and Mealy machines are deterministic finite state machines.

2.10.1 Moore Machine

Definition: If output symbol is associated with each state of finite state machine then such automata is called as Moore machine.

- It cover the transitions for all strings over the given alphabet.
- For every input string it generates an equivalent output string.
- If input string length is n, then length of the output string generated by Moore machine is $(n + 1)$.
- Moore machine generate the output associated with initial state by reading no input symbol.

Specification of Moore Machine

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a 6-tuple notation.

Where, Q is the set of finite states,

Σ is an input alphabet contain finite number of input symbols.

Δ is an output alphabet which contain finite number of output symbols.

δ is a transition function. $\delta : Q \times \Sigma \rightarrow Q$

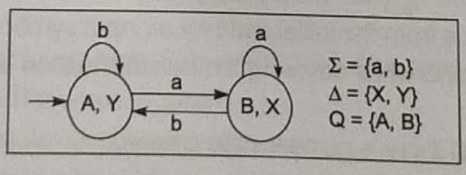
λ is a output function: $\lambda : Q \rightarrow \Delta$ (output associated with state)

q_0 is an initial state, $q_0 \in Q$

2.10.2 Construction of Moore Machine

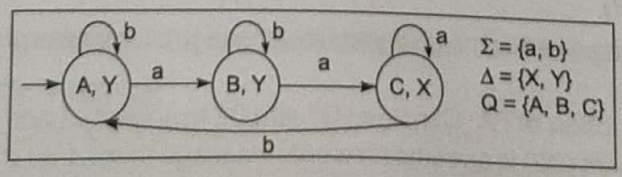
Example-2.113 Construct a Moore machine that generates an output X for every occurrence of 'a' in the input string. Otherwise it generates output Y. Consider input language is over the symbols of 'a' and 'b'.

Solution:



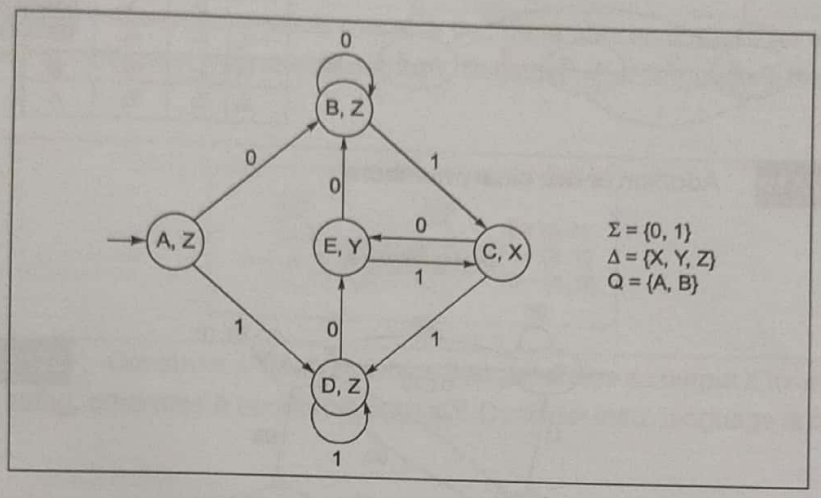
Example-2.114 Construct a Moore machine that generates an output X for every occurrence of 'aa' in the input string, otherwise it generates output Y. Consider input language is over the symbols of 'a' and 'b'.

Solution:



Example - 2.115 Construct a Moore machine that produces an output X for each occurrence of '01' in the input string, produces output Y for each occurrence of 10, otherwise it generates output Z. Consider input language is binary language.

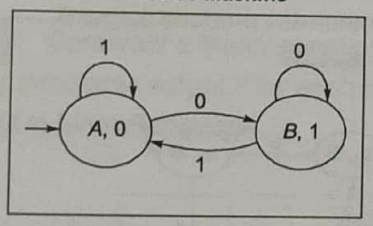
Solution:



Example - 2.116 One's complement of a binary number

Solution:

Moore Machine



or

	0	1	o/p
→A	B	A	0
B	B	A	1

Example - 2.117 2's complement of a binary number.

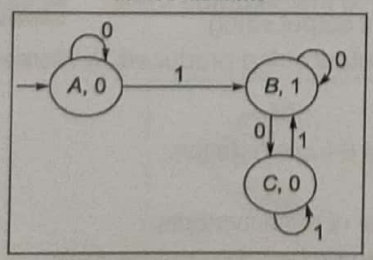
Solution:

Following machine reads an input from right to left and produces output from right to left.

Actual I/P: 0100 ⇒ O/P: 1100

Read I/P: 0, 0, 1, 0 ⇒ Produced O/P: 0,0,1,1

Moore Machine



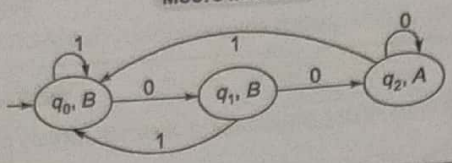
or

State	Input 0	Input 1	o/p
→A	A	B	0
B	B	C	1
C	B	C	0

Example-2.118 Produce output A for every occurrence of 00 in the input binary string, otherwise produce B.

Solution:

Moore Machine



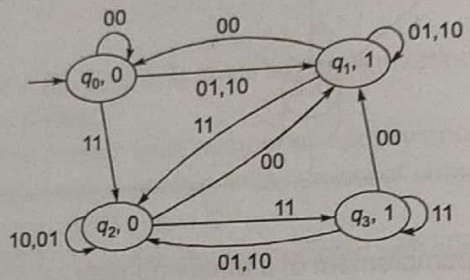
or

	0	1	o/p
$\rightarrow q_0$	q_1	q_0	B
q_1	q_2	q_0	B
q_2	q_2	q_0	A

Example-2.119 Addition of two binary numbers

Solution:

Moore Machine

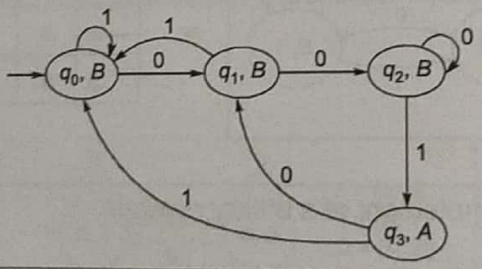


Example-2.120 Count the number of occurrences of 001 in the input binary string.

Solution:

Produce output A for every occurrence of 001 otherwise produce output B.

Moore Machine



2.10.3 Mealy Machine

Definition: If output symbol is associated with transition (state and input) of finite state machine then such automata is called as Mealy machine.

- It covers the transitions for all strings over given alphabet.
- For every input string it generates an equivalent output string.
- If length of input string is n then length of the output string produced by Moore machine is n.

Specification of Mealy Machine

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a 6-tuple notation.

Where, Q is the set of finite states,

Σ is an input alphabet contain finite number of input symbols.

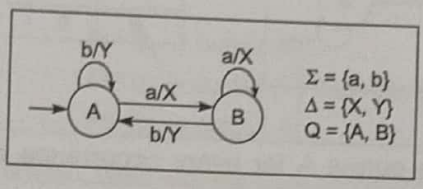
Δ is an output alphabet which contain finite number of output symbols.

δ is a transition function. $\delta : Q \times \Sigma \to Q$
λ is a output function: $\lambda : Q \times \Sigma \to \Delta$ (output associated with state and input)
q_0 is an initial state, $q_0 \in Q$

2.10.4 Construction of Mealy Machine

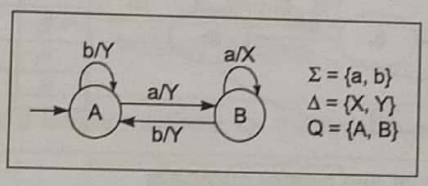
Example - 2.121 Construct a Mealy machine that generates an output X for every occurrence of 'a' in the input string, otherwise it generates output Y. Consider input language is over the symbols of 'a' and 'b'.

Solution:



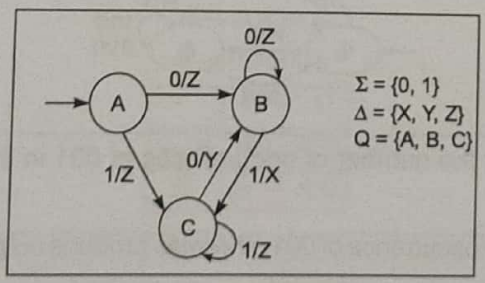
Example - 2.122 Construct a Mealy machine that generates an output X for every occurrence of 'aa' in the input string, otherwise it generates output Y. Consider input language is over the symbols of 'a' and 'b'.

Solution:



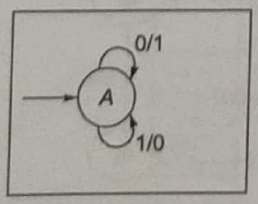
Example - 2.123 Construct a Mealy machine that produces an output X for each occurrence of '01' in the input string, produces output Y for each occurrence of 10, otherwise it generates output Z. Consider input language is binary language.

Solution:



Example - 2.124 One's complement of a binary number

Solution:



or

	0	1
$\to A$	A, 1	A, 0

Example-2.125 2's complement of a binary number.

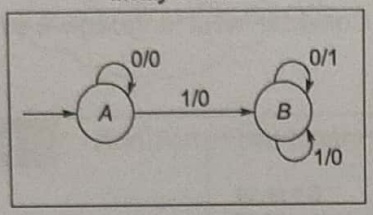
Solution:

Following machine reads an input from right to left and produces output from right to left.

Actual I/P: 0100 ⇒ O/P: 1100

Read I/P: 0, 0, 1, 0 ⇒ Produced O/P: 0,0,1,1

Mealy Machine



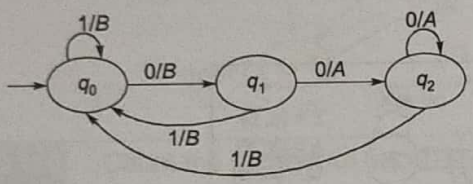
or

State	0		1	
	NS	o/p	NS	o/p
→A	A	0	B	0
B	B	1	B	0

Example-2.126 Produce output A for every occurrence of 00 in an input binary string, otherwise produce B.

Solution:

Mealy Machine



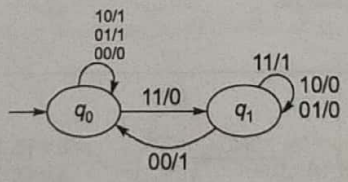
or

	0	1
→ q_0	q_1, B	q_0, B
q_1	q_2, A	q_0, B
q_2	q_2, A	q_0, B

Example-2.127 Addition of two binary numbers

Solution:

Mealy Machine

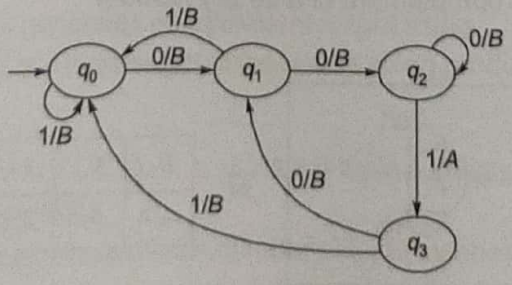


Example-2.128 Count the number of occurrences of 001 in the input binary string.

Solution:

Produce output A for every occurrence of 001 otherwise produce output B.

Mealy Machine



2.10.5 Conversion from Moore Machine to Mealy Machine

Example - 2.129 For the following Moore machine construct an equivalent Mealy machine

	0	1	output
$\rightarrow q_0$	q_1	q_0	B
q_1	q_2	q_0	B
q_2	q_2	q_0	A

Solution:

For $q_0 \rightarrow$ output B
For $q_1 \rightarrow$ output B } from the given Moore machine
For $q_2 \rightarrow$ output A

For all transitions the output is associated with the state and input.

	0	1
$\rightarrow q_0$	q_1, B	q_0, B
q_1	q_2, A	q_0, B
q_2	q_2, A	q_0, B

Mealy Machine

Example - 2.130 Construct Mealy machine for the following Moore machine.

	0	1	output
$\rightarrow A$	B	A	0
B	B	A	1

Solution:

$A \rightarrow$ output 0
$B \rightarrow$ output 1

	0	1
$\rightarrow A$	B,1	A,0
B	B,1	A,0

2.10.6 Conversion from Mealy Machine to Moore Machine

Example - 2.131 Construct an equivalent Moore machine for the following Mealy machine

	0	1
$\rightarrow q_0$	q_1, B	q_0, B
q_1	q_2, A	q_0, B
q_2	q_2, A	q_0, B

Solution:

B produced, for the state q_0
B produced, for the state q_1 } from Mealey machine transitions
A produced, for the state q_2

Therefore three states are needed in Moore machine.

	0	1	output
$\rightarrow q_0$	q_1	q_0	B
q_1	q_2	q_0	B
q_2	q_2	q_0	A

Moore Machine

Example - 2.132 Construct an equivalent Moore machine for the following Mealy machine

	0	1
$\rightarrow A$	A, 0	B, 0
B	B, 1	B, 0

Solution:

Output 0 produced for the state A

Output 0 or 1 produced for the state $B \nearrow^{B_0}_{B_0}$, so create B_0 and B_1 states in Moore machine.

Therefore three states needed to construct equivalent Moore machine

	0	1	Output
$\rightarrow A$	A	B_0	0
B_0	B_1	B_0	0
B_1	B_1	B_0	1

Moore Machine

Here, B_0 is created to produce output 0. B_1 is to produce the output 1.

NOTE: Every Moore machine can be converted to Mealy machine and every Mealy machine can be converted to Moore machine. Moore machine and Mealy machine are equivalent.

2.11 Pumping Lemma

- Pumping Lemma can be used to prove that certain sets are not regular sets
- Pumping Lemma is Proof By Contradiction technique which uses pigeon hole principle.
- It gives a method for pumping (generating) many substring from a given string. In other words, we can say it provides means to break a given long input string into several substrings.
- It gives necessary condition(s) to prove a set of strings is not regular.
- **Theorem:** $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA having n states. M recognizes the language L. A string $w \in L$ such that $|w| \geq n$ and $w = xyz$, where $y \neq \varepsilon$, then $xy^iz \in L$ for all $i \geq 0$.
- Proving Non-Regular Languages using Pumping Lemma:
  (i) Assume that language L is regular

(*ii*) There exist a pumping constant '*n*' such that for every string '*w*' in L of length '*n*' or more the following holds.

(*iii*) Let *w* has three parts *x*, *y* and *z* such that $w = xyz$, where $w \in L$ and $|w| \geq n$.

(*iv*) If $\forall i \geq 0$, $xy^i z \in L$, where $|y| > 0$ and $|xy| \leq n$ then L is regular.

(*v*) Find a suitable integer '*i*' such that $xy^i z \notin L$ and this contradicts our assumption hence L is not regular.

- The following languages are non-regular.
  $\{a^m b^m | i \geq 0\}$, $\{a^p | p \text{ is a prime}\}$, $\{a^m b^n | m > n\}$, $\{a^m b^n | m < n\}$, $\{a^m b^n | m! = n\}$

  $\{w | w \in (a + b)^*\}$ and *w* has equal number of a's and b's}, $\{ww | w \in (a + b)^*\}$, $\{a^{n^2}\}$, $\{a^{2^n}\}$ and $\{a^n b^n c^n\}$.

2.12 Minimization of Finite Automata

Finite automata need to be minimized to reduce number of states.

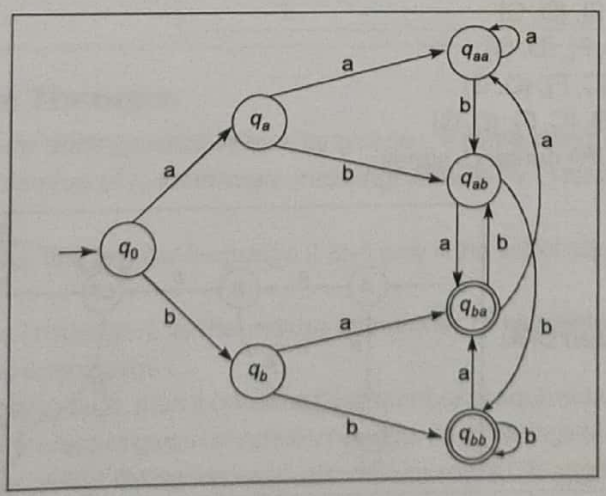
The advantage of the minimization saves the lot of space while the implementation of the problems.

2.12.1 Partition Method (State Equivalence Method)

Algorithm to minimize the DFA using partition method:

1. Separate the final states and non final states by making two separate sets one set contain all final states and other set contain all non final states.
2. Identify the **distinguishable** states and separate them if they are in same set.
   Distinguishable states: The state *p* and state *q* are distinguishable, "if $\delta(p, x)$ is final and $\delta(q, x)$ is non final" or "if $\delta(p, x)$ is non final and $\delta(q, x)$ is final". i.e., one state is final and other is non-final.
3. Distinguishable states can be found **directly or indirectly** from the sets.
   (a) "If $\delta(p, x)$ is final and $\delta(q, x)$ is non final" or "If $\delta(p, x)$ is non final and $\delta(q, x)$ is final" , then *p* and *q* are distinguishable and hence separated from the same set.
   (b) The state *p* and state *q* are distinguishable (Direct or Indirect)
   Let $\delta(p, x) = r$ and $\delta(q, x) = s$. If *r* and *s* are distinguishable (*s* and *r* in different sets) then (p, q) is also distinguishable. So states *p* and *q* are separated from the set.
4. Repeat step 3 until there is no distinguishable pair found.

Example - 2.133 Minimize the following DFA using partition method



Solution:

$\{q_0, q_a, q_b, q_{aa}, q_{ab}\}$ is non final states set, $\{q_{ba}, q_{bb}\}$ is final set

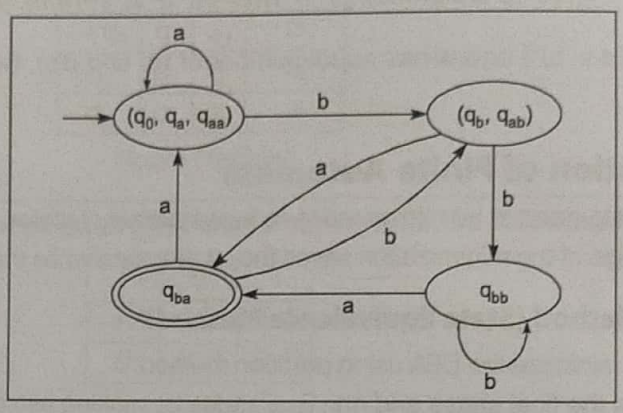
Step 1: $\{q_0, q_a, q_b, q_{aa}, q_{ab}\}\{q_{ba}, q_{bb}\}$ (Partitions after step 1)

Step 2:$\{q_0, q_a, q_{aa}\}\{q_b, q_{ab}\}\{q_{ba}\}\{q_{bb}\}$ (Partitions after step 2)

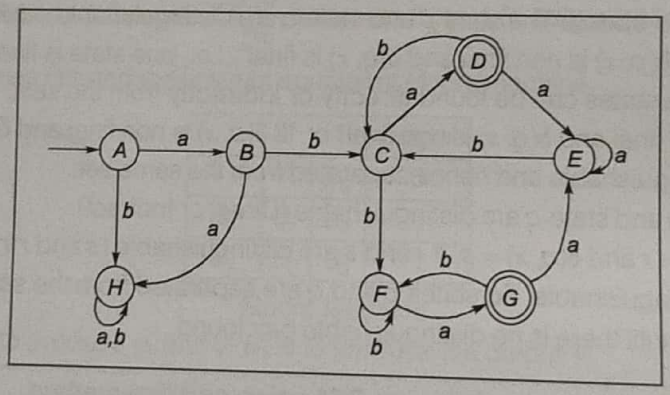
Step 3:$\{q_0, q_a, q_{aa}\}\{q_b, q_{ab}\}\{q_{ba}\}\{q_{bb}\}$(Partitions after step 3, same as step 2)

The minimized DFA contain four states.

Minimized DFA:



Example - 2.134 Minimize the following DFA using partition method



Solution:

Step1: {A, B, H, C, F, E}, {D, G}

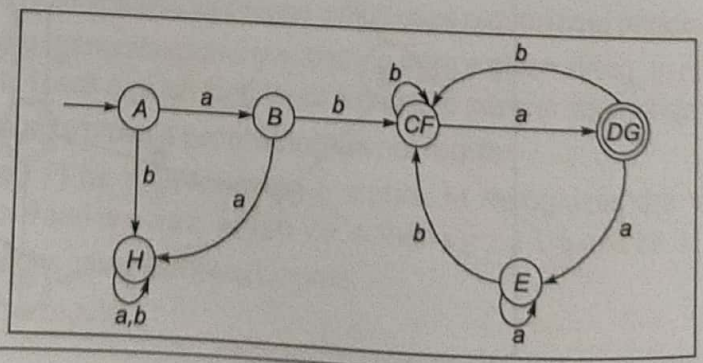
Step2: {A, B, H, E},{C, F}, {D, G}

Step3: {A, H}, {B, E}, {C, F}, {D, G}

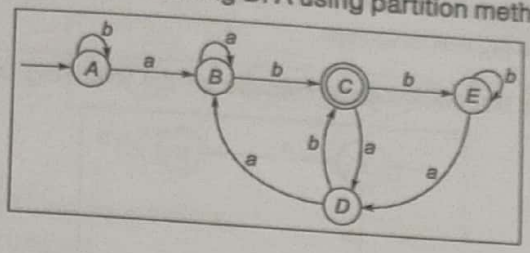
Step4: {A}, {H}, {B}, {E}, {C, F}, {D, G}

Therefore minimized DFA contain 5 states.

Minimized DFA:



Example - 2.135 Minimize the following DFA using partition method



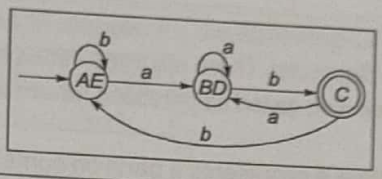
Solution:

Step1: {A, B, E, D}, {C}

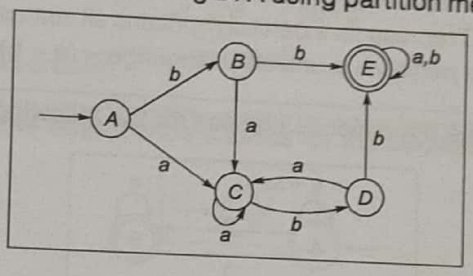
Step2: {A, E}, {B, D}, {C}

Minimized DFA contain 3 states.

Minimized DFA:



Example - 2.136 Minimize the following DFA using partition method



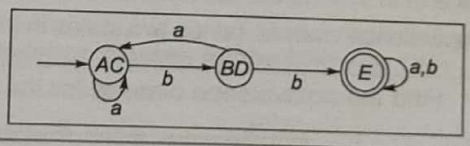
Solution:

Step1: {A, B, C, D}, {E}

Step2: {A, C}, {B, D}, {E}

Minimized DFA contain 3 states.

Minimized DFA:



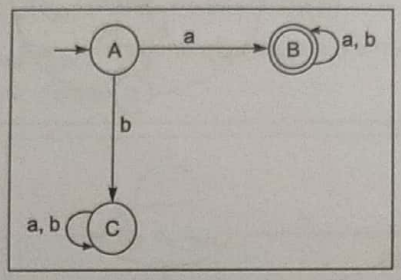
2.13 Myhill – Nerode Theorem

String u and string v are **distinguishable** by a language L if some string w exists such that exactly one of the strings uw and vw is a member of L. Otherwise (indistinguishable by L) for every string w, uw and vw are members of L.

- A given language L is a regular language if and only if the set of equivalence classes of L is finite (finite index).
- **Index**: Index of a language L is the maximum number of elements in any set that has pairwise distinguishable by a language L.
- Let L be a regular language, then it contains finite number of equivalence classes. Union of all Myhill-Nerode equivalence classes gives universal language. All the strings of universal language partitioned into a finite number of Myhill-Nerode equivalence classes, if L is regular.

- The number of Myhill-Nerode equivalence classes for a regular language ≡ Number of states in the minimal DFA recognizing that language.

Example - 2.137 Find the equivalence classes for the following DFA



Solution:

Given DFA is already minimized. There are three states hence three equivalence classes.

Equivalence classes are also called as partitions, where each partition contains set of strings accepted by that state.

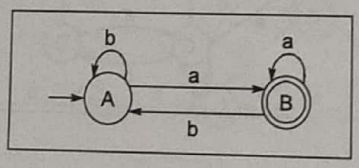
[A] = State A represents a partition contains {ε}

[B] = State B represents a partition contains all strings of the form a(a + b)*

[C] = State C represents a partition contains all strings of the form b(a + b)*

Union of all the above three partitions is universal language = $(a + b)^* = \Sigma^* =$ [A] ∪ [B] ∪ [C]

Example - 2.138 Find the equivalence classes for the following DFA



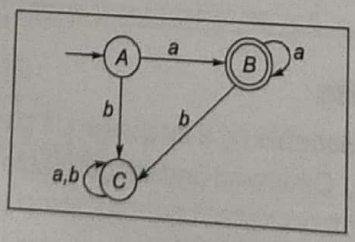
Solution:

A = {empty string or all strings that end in b} = {ε} ∪ {b, ab, bb,...} = ε + (a + b)*b

B = {All strings that end in a} = {a, ba, aa, aaa, bba,...} = (a+b)*a

So there are two equivalence classes. (since two states in minimized DFA)

Example - 2.139 Find the equivalence classes for the following DFA



Solution:

$[A] = \varepsilon$

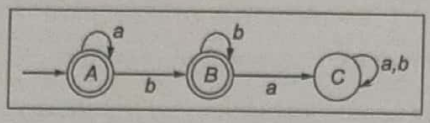
$[B] = a^+$

$[C] = (b + ab)(a + b)^*$

} Three equivalence classes (three partitions)

$[A] \cup [B] \cup [C] = \varepsilon + a^+ + (b + ab)(a + b)^* = (a + b)^* = \Sigma^*$

Example-2.140 Find the equivalence classes for the following DFA



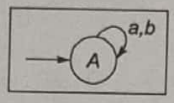
Solution:

$[A] = a^*$
$[B] = a^*b^+$
$[C] = a^*b^+a(a+b)^*$

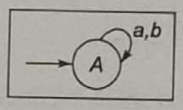
} Three equivalence classes

$[A] \cup [B] \cup [C] = a^* + a^*b^+ + a^*b^+ a(a+b)^* = (a+b)^*$

Example-2.141 Find the equivalence classes for the following DFA



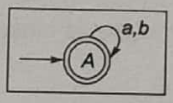
Solution:



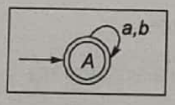
$[A] = \Sigma^*$

Only one equivalence class present.

Example-2.142 Find the equivalence classes for the following DFA



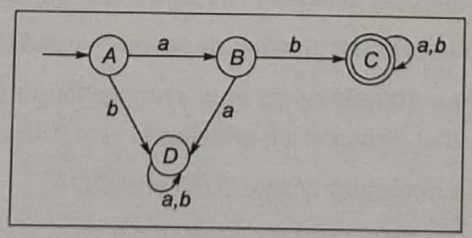
Solution:



$[A] = \Sigma^*$

Only one equivalence class present.

Example-2.143 Find the equivalence classes for the following DFA



Solution:

$[A] = \varepsilon$
$[B] = a$
$[C] = ab(a+b)^*$
$[D] = b(a+b)^* + aa(a+b)^*$
4 equivalence classes.
$[A] \cup [B] \cup [C] \cup [D] = \Sigma^*$

Summary

- **DFA:** Deterministic finite automata that accepts a regular language.
- **NFA:** Non-deterministic finite automata that accepts a regular language.
- **ε-NFA:** NFA with epsilon transitions that accepts a regular language.
- **Equivalence:** DFA ≡ NFA ≡ ε-NFA
- **Regular Expression:** It is a representation uses the operators such as ·, +, *, ⁺ to generate the regular language.
- **State Elimination Method:** All states are eliminated except an initial state and a final state to write equivalent regular expression.
- **Kleene Method:** $R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^{*} R_{kj}^{k-1}$
- **Arden's Method:** Equation R = Q+RP has a solution R = QP*
- **Left Linear Grammar:** $V \rightarrow VT^* | T^*$
- **Right Linear Grammar:** $V \rightarrow T^*V | T^*$
- **Regular Grammar:** It is equivalent to left linear grammar and right linear grammar.
- **Closure properties of regular Languages:** ∪, ∩, −, prefix, reversal, complement, concatenation, etc. are closed.
- **Subset Operation:** Finite subset is closed but infinite subset is not closed for regular languages.
- **Decision Properties:** Emptiness, Finiteness, Membership, equivalence etc. are decidable properties for regular languages.
- Moore and Mealy machines are equivalent
- Pumping lemma and Myhill-nerode theorem can be used to prove the non-regular languages.
- **Moore Machine:** Output function depends only on state.
- **Mealy Machines:** Output function depends on state and input.
- **Two-way finite automata:** It accepts a regular language where it contain a head which can move in either direction left and right.
- **Two-way finite automata types:** 2DFA and 2NFA.
- The set of regular languages is closed under the union operation.
- Finite union is closed for regular languages but infinite union is not closed.
- The set of regular languages is closed under the intersection operation.
- Regular languages are closed under finite intersection but not under infinite intersection.

Student's Assignments 1

Q.1 Consider a regular expression $R = (a + \varepsilon)(bb^*a)^*$. What is the language generated by R over $\Sigma = \{a, b\}$

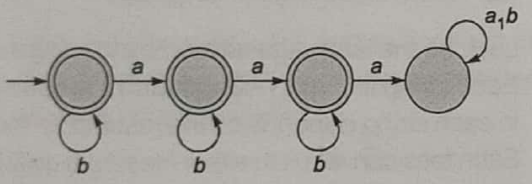
(a) Set of all strings that do not contain aa.
(b) Set of all strings that do not contain two or more consecutive a's.
(c) Set of all strings that do not end with b and do not contain two or more consecutive a's.
(d) None of these

Q.2 Let $L = \{pq \mid p, q \in \{0, 1\}^*$ and dec(p) × dec(q) = 10 (ten)}, where dec(x) is decimal equivalent of binary number x. Find the language L by assuming that leading zero's are allowed for the strings of L.

[01 = 001 = 0001]

(a) Regular
(b) CFL but not regular
(c) Recursive but not CFL
(d) None of these

Q.3 Consider the following DFA:



Identify the language accepted by the DFA?

(a) All strings not contain aaa
(b) All strings not starting with aaa.
(c) All strings with no more than two a's
(d) None of these

Q.4 Let L_1 and L_2 be languages over Σ. If L_1 is finite language and $L_1 \cup L_2$ is regular then L_2 is _____?

(a) Regular language and finite
(b) Regular language and infinite
(c) Need not be regular
(d) None of these

Q.5 Find the number of states in minimized DFA for the regular expression $a^+b^+ + b^+a^+$

(a) 4 (b) 5
(c) 6 (d) None of these

Q.6 Consider the following configuration of a DFA.
DFA = $(Q, \Sigma, \delta, q_0, A)$ where $Q = \{q_0, q_1\}$, q_0 is initial state, A is set of final states.
Given, $A = \{q_1\}$ and $\delta(q_i, x) = q_{1-i}$, $x \in \Sigma$, $i = 0, 1$.
Identify the language accepted by the above DFA?

(a) Set of all strings over a given alphabet Σ
(b) Set of all strings of odd length over Σ
(c) Set of all strings of even length over Σ
(d) None of these

Q.7 Identify the non regular language from the following.

(a) $\{x \mid x \in (0+1)^*, |x|$ is odd, First symbol of x is 1$\}$
(b) $\{x \mid x \in (0+1)^*, |x|$ is odd, Middle symbol of x is 1$\}$
(c) $\{x \mid x \in (0+1)^*, |x|$ is odd, Last symbol of x is 1$\}$
(d) None of these

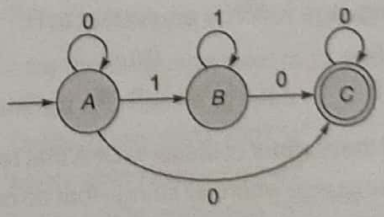
Q.8 Consider a regular language R over $\Sigma = \{a, b\}$ which is equivalent to the regular expression $(ab + b)^*$. How many equivalence classes of Σ^* are present for the language R?

(a) 2 (b) 3
(c) 4 (d) 5

Q.9 Let $L = \{w x w^R \mid w \in (a + b)^*, x \in (a + b)\}$. The complement of language L is _____.

(a) Regular
(b) Finite language
(c) Non regular language
(d) None of these

Q.10 What is the language accepted by the following NFA?



(a) $(0 + 1)^*0$ (b) $0^*1^*0^*$
(c) $0^*1^*0^*0$ (d) $(0 + 1)^+$

Q.11 Find the number of states in minimized DFA that accepts a language over $\Sigma = \{a, b\}$ where each string has exactly three a's and atleast two b's.

(a) 12 (b) 13
(c) 14 (d) 15

Q.12 Let $L_1 = a^*b^*$ and $L_2 = \{ab\}$. L_3 = Prefix $(L_1^* \cap L_2)$, where prefix $(L) = \{u \mid uv \in L$ for any $v\}$. Find the number of strings in L_3?

(a) 3 (b) 4
(c) 5 (d) 6

Q.13 How many number of states are there in the minimized DFA that accepts the following language L.

$$L = \{a^{3(n+1)} \mid n \geq 0\}$$

(a) 3 (b) 4
(c) 5 (d) 6

Q.14 Find the number of states in DFA which accepts a language such that each block of 4 consecutive symbols of every string contain at least two a's for $\Sigma = \{a, b\}$. If string length is less than 4 then it must be accepted.

(a) 10 (b) 11
(c) 12 (d) 13

Common Data Questions (15 and 16):

Consider the following regular expression R.

$$R = a^*b^* + b^*a^*$$

Q.15 How many final states exist in the minimized DFA that accepts a language equivalent to R.

(a) 3 (b) 4
(c) 5 (d) 6

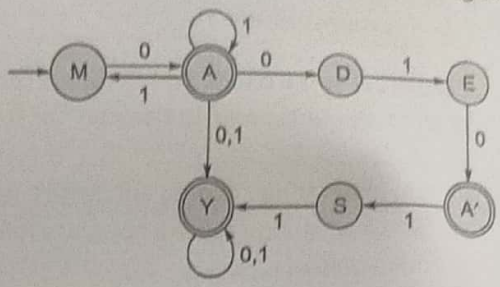
Q.16 How many equivalence classes of Σ^* to represent a language which is equivalent to R.

(a) 3 (b) 4
(c) 5 (d) 6

Q.17 Find the number of states in DFA that recognizes the language where all strings that do not contain the substring bab, for $\Sigma = \{a, b\}$

(a) 3 (b) 4
(c) 5 (d) 6

Q.18 Find the language accepted by the following NFA.



(a) All strings of 0's and 1's
(b) All strings starting with '1'
(c) All strings starting with '0'
(d) None of these

Q.19 In MADE EASY, "Reddy" found a regular language named as L1. "Kumar" is a friend of Reddy who found a new language L2. "Singh" is teaching TOC to Reddy and Kumar, He has collected both L1 and L2 to compute the new language L3. If L3 = L1 – L2 then Singh found that L3 is _____.

(a) Regular language
(b) Non regular language
(c) Finite language
(d) Need not be regular language

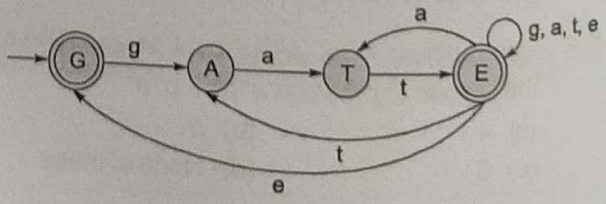
Q.20 Let L be the language formed by tossing a coin. Each string in L has *n*-length and the sequence in each string depends on the result of '*n*' tosses. Each toss can result in either Head (H) or Tail (T). Find the number of strings in L over the input alphabet $\Sigma = \{T, H\}$

(a) n (b) n^2
(c) 2^n (d) n^n

Q.21 How many states are required to construct a DFA that accepts a binary language 'L', where every string in 'L' contains a substring '0' or '1'.

(a) 1 (b) 2
(c) 3 (d) 4

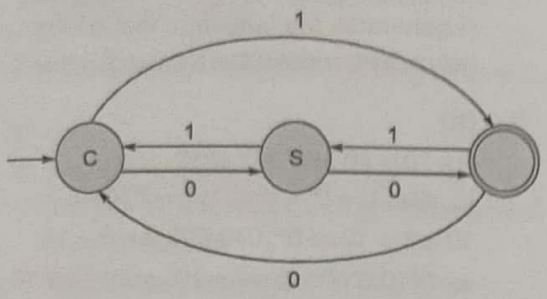
Q.22 Consider the following NFA for $\Sigma = \{g, a, t, e\}$



Find the number of states in equivalent DFA.

(a) 4 (b) 5
(c) 6 (d) 8

Q.23 Consider the following finite automata F1 that accepts a language 'L'



Let F2 be a finite automata which is obtained by the reversal of F1. Then which of the following is correct?

(a) $L(F1) \neq L(F2)$ (b) $L(F1) = L(F2)$
(c) $L(F1) \subseteq L(F2)$ (d) $L(F2) \subseteq L(F1)$

Q.24 Let $L_1 = (0 + 1)^*$, $L_2 = (01^*0 + 1^*)$ and $L_3 = (01^+ 0 + 1^+ + \varepsilon)$. If $L = L_1 \cap L_2 \cap L_3$ then find number of strings in L which do not contain 11.

(a) 2 (b) 3
(c) 4 (d) 5

Q.25 Let $L = \{w | w \in (a + b)^*$ and w does not end in $ba\}$. Find the number of states needed for minimal DFA to accept L

(a) 2 (b) 3
(c) 4 (d) 5

Q.26 Let L = {madeeasy} over Σ = {m,a,d,e,s,y}, L1 = prefix(L) and L2 = $L1/\Sigma^*$. Find the number of strings in L2. Assume L1 and L2 do not include empty string.

(a) 6 (b) 7
(c) 8 (d) 9

Q.27 Let L1 be the language accepted by DFA D1 and L2 be the language accepted by DFA D2. Similarly L3 and L4 languages are accepted by NFA N1 and N2 respectively.

1. D2 is obtained by swapping the accepting and non-accepting states of D1
2. N2 is obtained by swapping the accepting and non-accepting states of N1

Which of the following statement is incorrect?

(a) L2 is complement of L1
(b) Class of languages recognized by NFA's is not closed under complement
(c) Class of languages recognized by DFA's is closed under complement
(d) None of these

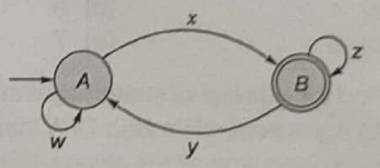
Q.28 Assume every L_i is regular language for $i \geq 1$. If infinite union is defined as : $X = L_1 \cup L_2 \cup L_3 \ldots = \bigcup_{i \geq 1} L_i$ then X is _____.

(a) Regular
(b) Finite language
(c) Non regular language
(b) Need not be regular language

Q.29 Let L_1 be a regular language and $L_2 = L_1^*$. Then L_2 is _____.

(a) Always finite language
(b) Always infinite language and regular
(c) Need not be infinite language
(d) None of these.

Q.30 Consider the following state diagram:



Assume $w, x, y, z \in \{0, 1\}$. If above state diagram is possible to make a DFA with deterministic inputs w, x, y, z then find number of possible DFA's. [Assume all given states are fixed]

(a) 4 (b) 5
(c) 6 (d) 7

Q.31 Let L be the language over {a,b} and it contain all strings with following rules.

1. Every string starts with "a"
2. Every string contains "ab"
3. Every string ends with "b"

Find the number of states required in the minimized DFA that accepts L.

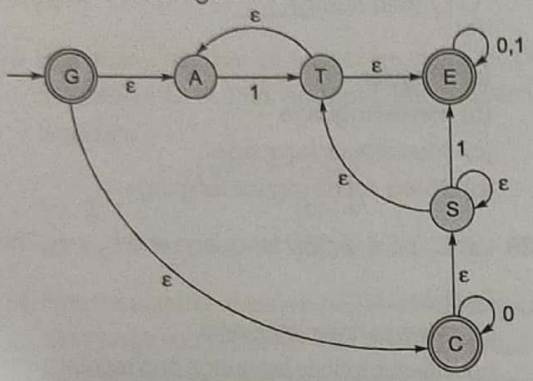
(a) 4 (b) 5
(c) 6 (d) 7

Q.32 Let L be a language and $L = \{w \mid w$ has exactly one 'b', $w \in \{a, b\}^*, \mid w \mid$ is even$\}$. Find the minimum number of states required for DFA that accepts L _____.

(a) 4 (b) 5
(c) 6 (d) 7

Common Data (Q.33 and Q.34)

Consider the following ε-NFA:



Q.33 Find the cardinality of epsilon closure of state 'S' [ε-closure (q) is the set which contain all the states those are reachable from state q without reading any input symbol]

(a) 4 (b) 5
(c) 6 (d) 7

Q.34 Find the number of states required to construct an equivalent minimized DFA that accepts the same language as the given ε-NFA.

(a) 1 (b) 2
(c) 3 (d) None of these

Answer Key:

1. (c)	**2.** (a)	**3.** (c)	**4.** (d)	**5.** (c)
6. (b)	**7.** (b)	**8.** (b)	**9.** (c)	**10.** (c)
11. (b)	**12.** (a)	**13.** (b)	**14.** (c)	**15.** (c)
16. (d)	**17.** (b)	**18.** (c)	**19.** (d)	**20.** (c)
21. (b)	**22.** (b)	**23.** (b)	**24.** (b)	**25.** (b)
26. (c)	**27.** (b)	**28.** (d)	**29.** (c)	**30.** (a)
31. (a)	**32.** (b)	**33.** (a)	**34.** (a)	

Student's Assignments 1 Explanations

1. (c)

$R = (a + \varepsilon)(bb^*a)^*$

R generates the language that do not contain two or more consecutive a's and do not end with b.

2. (a)

$1 \times 10 = 10 \Rightarrow 0^*1.0^*1010$

$[\because \text{dec}(1) = 0^*1, \text{dec}(10) = 0^*1010]$

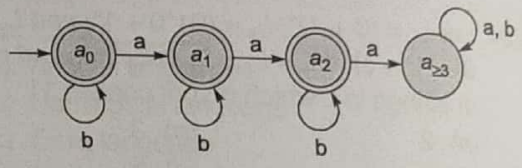
$10 \times 1 = 10 \Rightarrow 0^*1010.0^*1, 2 \times 5 = 10$

$\Rightarrow 0^*10.0^*101, 5 \times 2 = 10 \Rightarrow 0^*101.0^*10$

$\Rightarrow$ R.E. $= 0^*10^*1010 + 0^*1010\,0^*1 + 0^*100^*101 + 0^*1010^*10$

$\therefore$ L is regular language

3. (c)



All strings with atmost two a's are accepted by DFA.

4. (d)

Example 1

$L_1 = \{ba\}, L_2 = \{a^*b^*\}$

$L_1 \cup L_2 = ba + a^*b^*$

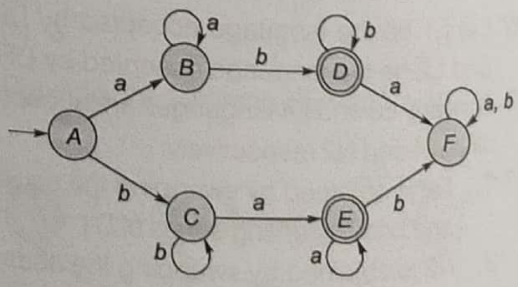
Example 2

$L_1 = \{ba\}, L_2 = \{a\}$

$L_1 \cup L_2 = ba + a$

$\therefore$ L_2 is regular but it may be finite or infinite.

5. (c)

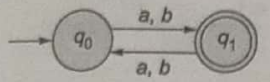


$\therefore$ 6 states are required to accept $a^+b^+ + b^+a^+$

6. (b)

Let $\Sigma = \{a, b\}$

Given, $\boxed{\delta(q_i, x) = q_{1-i},\ x \in \Sigma, i = 0, 1}$



is equivalent DFA.

∴ DFA accepts all strings of odd lengths over Σ.

7. (b)

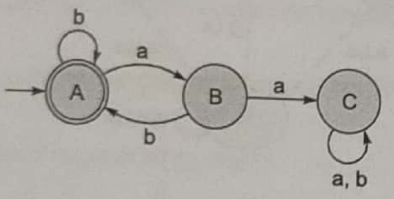
(a) Regular language: $1\,[(0 + 1)(0 + 1)]^*$

(b) Non regular language (Finding middle symbol is not possible)

(c) Regular language: $[(0 + 1)(0 + 1)]^* 1$

8. (b)

Minimized DFA for the language R:



$[A] = (ab + b)^*$

$[B] = (ab + b)^* a$

$[C] = (ab + b)^*\, aa\, (a + b)^*$

∴ Three equivalence classes are present.

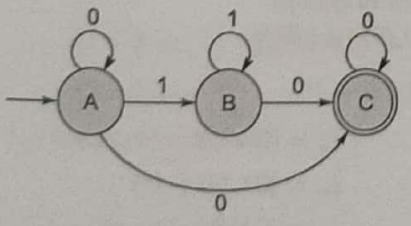
$([A] \cup [B] \cup [C] = \Sigma^*)$

9. (c)

$\bar{L}$ has every even length string and it contain all odd length strings which are not in the form of $w x w^R$.

L is non regular. Hence $\bar{L}$ is also non regular.

10. (c)



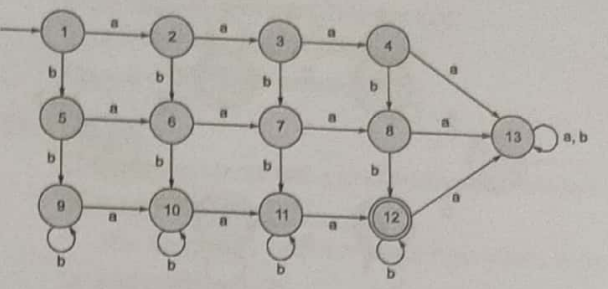
R.E. $= 0^*(11^* 0 + 0)\, 0^*$

$= 0^*((11^* + \varepsilon)\, 0)\, 0^* = 0^*\, 1^*\, 0\, 0^*$

$= 0^*\, 1^*\, 0^*\, 0$

So, option (c) is correct.

11. (b)



Number of states = 13 states.

12. (a)

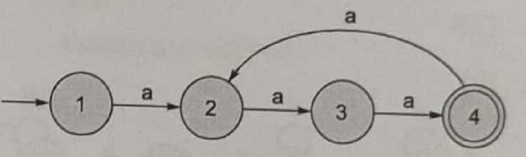
$L_1 = a^*b^* \Rightarrow L_1^* = (a^* b^*)^* = (a + b)^*$

$L_2 = \{ab\}$

$L_1^* \cap L_2 = (a + b)^* \cap \{ab\} = \{ab\}$

$L_3 = \text{Prefix}\,(L_1^* \cap L_2) = \{\varepsilon, a, ab\}$

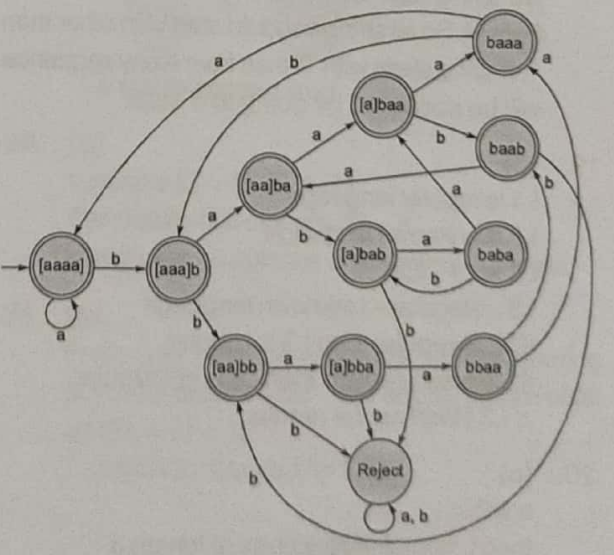
13. (b)



$L = a^{3(n+1)} = a^{3n+3}$

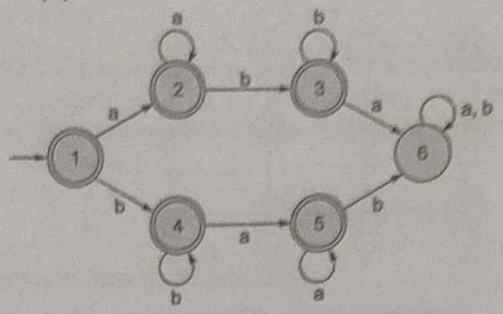
∴ 4 states are present in minimized DFA.

14. (c)



∴ Total 12 states.

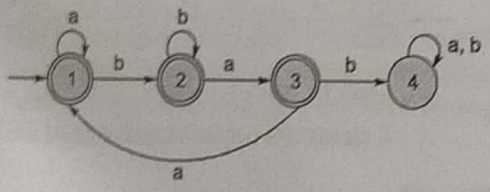
15. (c)



16. (d)

[1] = {ε}
[2] = {aa*}
[3] = {aa* bb*}
[4] = {bb*}
[5] = {bb* aa*}
[6] = {(aa*bb*a + bb*aa*b) (a + b)*}
∴ 6 equivalence classes = 6 states in minimized DFA

17. (b)



18. (c)

No string can start with 1, as there is no path from M. So all strings should start with other than 1. If string starts with '0' then from A any sequence will be accepted by going to Y state.

19. (d)

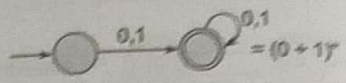
L1 is regular language
L2 is unknown language
L3 = L1 – L2
L3 = Regular – Unknown language
If L2 is regular then L3 is regular
If L2 is non-regular then L3 is not regular
∴ L3 Need not be regular

20. (c)

n = 3
then L contains all strings of length 3
L = {HHH, HHT, HTH, HTT, THH, THT, TTH, TTT}
∴ |L| = 2^3 = 8 strings

21. (b)

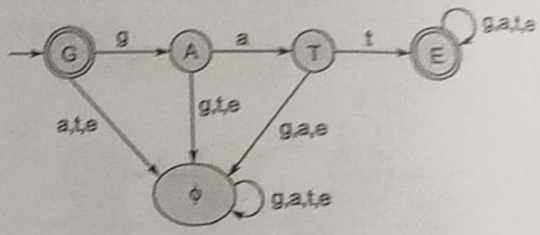
Contains a substring '0' or '1'
≡ starts with 0 or 1



∴ It needs 2 states

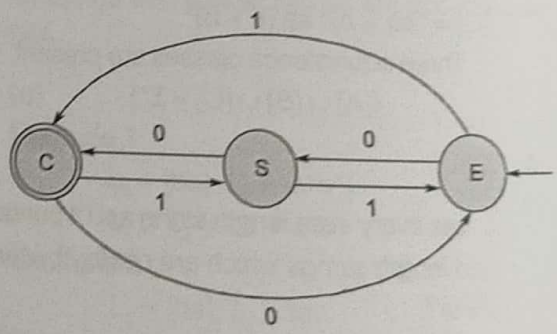
22. (b)

The given NFA accepts a language where each string starts with 'gat' [including Null string]
∴ Number of states required in DFA = 4 + 1
= 5 states



23. (b)

F2 = Reversal of F1 :

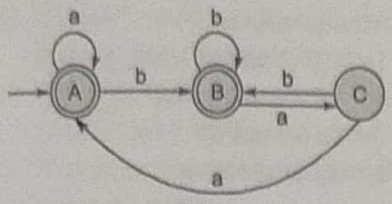


This is same as F1 except the state names.
Reversal of F1 is equivalent to F1, Both accept same language.
So, L(F2) = L(F1)

24. (b)

$L_1 = (0 + 1)^*$
$L_2 = (01^*0 + 1^*)$
$L_3 = (01^+0 + 1^+ + \varepsilon)$
$L_1 \cap L_2 \cap L_3 = L_2 \cap L_3 = L_3 = 01^+0 + 1^+ + \varepsilon$
Number of strings which do not contain 11 ⇒ ε, 1, 010 = 3 strings

25. (b)



3 States

26. (c)

L = {madeeasy} ⇒ L1 = {m, ma, mad, made, madee, madeea, madeeas, madeeasy}

L2 = L1/Σ* = {m, ma, mad, made, madee, madeea, madeeas, madeeasy} = L1

∴ Total 8 strings

[∵ Prefix (L)/Σ* = Prefix (L)]

27. (b)

Class of languages recognized by NFA's

≅

Class of languages recognized by DFA's

≅

Class of regular languages

⇓

Closed under complement.

So option (b) is not correct statement.

[*Note:* Given data is applicable to option (a) only]

28. (d)

$$X = L_1 \cup L_2 \cup L_3 \ldots$$

Case1: $\Sigma^* \cup L_2 \cup L_3 \ldots = \Sigma^*$. X is regular [in one case]

Case 2 : $\{\varepsilon\} \cup \{ab\} \cup \{a^2b^2\} \cup \ldots\ldots = \{a^n b^n\} \Rightarrow$ X is non-regular

(a), (b), (c) options can not be correct. We can conclude that X need not be regular.

29. (c)

$L_1 = \phi \Rightarrow L_1^* = \{\varepsilon\}$ is finite

$L_1 = \{a\} \Rightarrow L_1^* = \{a^*\}$ is infinite

∴ L_2 need not be infinite

30. (a)

state	A (w)	A (x)
Input : 0	1	
Input : 1	0	

state	B (y)	B (z)
Input : 0	1	
Input : 1	0	

From state A, there are two choices

From state B, there are two choices

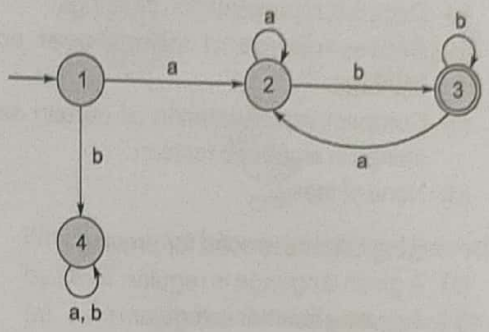
∴ 2 × 2 = 4 possible selections

Hence 4 DFA's are possible

31. (a)

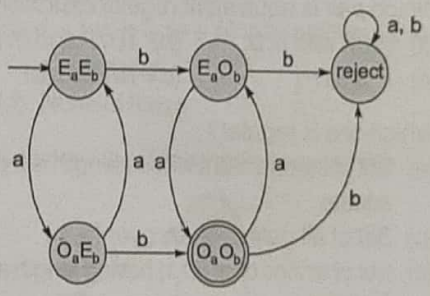
Every string starts with a and ends with b always contains substring ab.

∴ The language contain all strings starts with 'a' and ends with 'b'



∴ 4 states are required

32. (b)



∴ 5 states are required.

33. (a)

ε-closure (S)= {S, T, A, E}

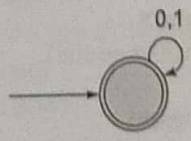
Cardinality of ε-closure (S) = 4

[Cardinality is number of elements in the set]

34. (a)

From G to E can be reachable without reading any input and then can read any sequence of 0's and1's = (0 + 1)*

∴ 1 state is required for DFA



Student's Assignments 2

Q.1 Regular languages are recognized by
(a) Finite Automaton
(b) Pushdown Automaton
(c) Turing Machines
(d) All of the above

Q.2 Regular expressions are
(a) Compact representation of strings
(b) Representation of strings over some alphabet
(c) Compact representation of certain set of strings in algebraic fashion
(d) None of these

Q.3 Pumping Lemma is used for proving
(a) A given language is regular
(b) A given grammar is regular
(c) A given language is not regular
(d) All of the above

Q.4 Which pair is equivalent regular expression?
(a) $(ab)^*$ and a^*b^* (b) $r(rr)^*$ and $(rr)^*r$
(c) r^+ and r^*r (d) (b) and (c)

Q.5 Which one is regular?
(a) Set of strings of a's whose length is a perfect square
(b) Set of all palindromes over {a, b}
(c) Set of strings over {0, 1} having length a prime number
(d) Set of strings over {0} having odd number of zeros

Q.6 The logic of Pumping lemma is a good example of
(a) The Pigeon-hole principle
(b) Divide and conquer method
(c) Iteration
(d) Recursion

Q.7 Set of regular languages over certain alphabet is not closed under
(a) Complement
(b) Iteration (Kleene star)
(c) Union
(d) None of these

Q.8 Which of the following pairs of regular expression are not equivalent?
(a) $(a^* + b^*)^*$ and $(a + b)^*$
(b) $(a^* + b)^*$ and $(a + b)^*$
(c) $(ab)^*a$ and $a(ba)^*$
(d) None of the above

Q.9 Choose the correct statement. A class of languages that is closed under
(a) Intersection and complementation has not to be closed under union
(b) Union and complementation has to be closed under intersection
(c) Union and intersection has to be closed under complementation
(d) Both (b) and (c)

Q.10 Let $\Sigma = \{a, b\}$, $L = \{a^n b^n : n \geq 1\}$ and $R = \Sigma^*$, then the languages $R \cup L$ and R are
(a) Regular, regular
(b) Regular, not regular
(c) Not regular, regular
(d) Not regular, not regular

Q.11 P, Q and R are three languages. P and R are regular languages and R = PQ, then
(a) Q is non regular
(b) Q is regular
(c) Q is finite language
(d) Q need not be regular

Q.12 Myhill-Nerode theorem can find which of the following
(a) Proves regular languages
(b) Proves non regular languages
(c) Index of a language
(d) All of the above

Q.13 Myhill-Nerode theorem is used for
(a) Finding equivalence classes
(b) Minimization of FA
(c) Proving regular and not regular
(d) All of the above

Q.14 The Pumping Lemma is used for
(a) Proving regular languages only
(b) Proving every language
(c) Proving non regular languages
(d) Proving finite languages

Q.15 Choose the correct statement
(a) For every regular language, there is right linear and left linear grammar
(b) For every regular language, there is right linear or left linear grammar but not both
(c) Transpose of a regular language is regular
(d) Both (a) and (c)

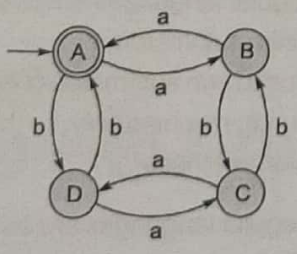
Q.16 Consider the following grammar G:
$S \rightarrow bS \mid aA \mid b$
$A \rightarrow bA \mid aB$
$B \rightarrow bB \mid aS \mid a$

Let $N_a(w)$ and $N_b(w)$ denote the number of a's and b's in a string w respectively. The language $L(G) \subseteq (a + b)^*$ generated by G is
(a) $\{w \mid N_a(w) > 3N_b(w)\}$
(b) $\{w \mid N_b(w) > 3N_a(w)\}$
(c) $\{w \mid N_a(w) = 3k, k \in \{0, 1, 2, 3, \ldots\}\}$
(d) $\{w \mid N_b(w) = 3k, k \in \{0, 1, 2, 3, \ldots\}\}$

Q.17 A language containing all the words over {a, b} having even number of a's and b's is
(a) (aa + bb + (ab + ba)(aa + bb)* (ab + ba))*
(b) (ab + ba + (ab + ba)(aa + bb)* (ab + ba))*
(c) (aa + bb + ab + ba)*
(d) None of these

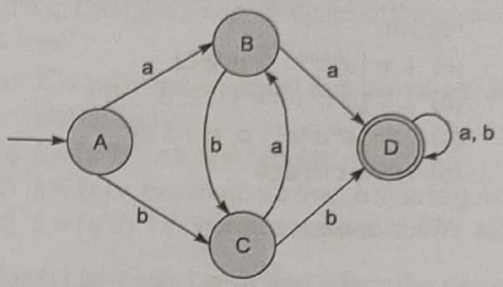
Q.18 Which one of the following regular expressions pair, do not define the same language?
(a) (a* + b)* and (a + b)*
(b) (ab)*a and a(ba)*
(c) (a* + b*)* and (a + b)*
(d) None of these

Q.19 Consider the following deterministic finite state automata



The language accepted by the above automata is
(a) The words over {a, b} having even number of a's and odd number of b's.
(b) The words over {a, b} having odd number of a's and odd number of b's.
(c) The words over {a, b} having even number of a's and even number of b's.
(d) The words over {a, b} having odd number of a's and even number of b's.

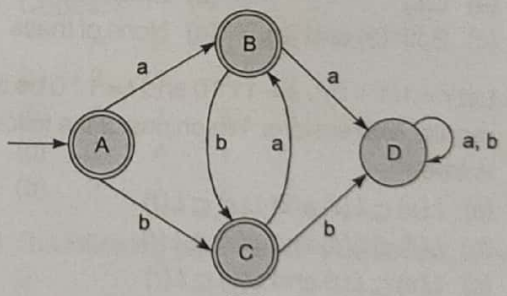
Q.20 Consider the following DFA



DFA

Which one of the following languages is accepted by DFA?
(a) $L = \{w : w \in (a + b)^*$ has either aa or bb as a substring$\}$
(b) $L = \{w : w \in (a + b)^*$ has both aa and bb as substring$\}$
(c) $L = \{w : w \in (a + b)^*$ has neither aa nor bb as a substring$\}$
(d) None of these

Q.21 Consider the following finite automata



DFA

Which one of the following languages is accepted by DFA?
(a) $L = \{w : w \in (a + b)^*$ has either aa or bb as a substring$\}$
(b) $L = \{w : w \in (a + b)^*$ has both aa and bb as substring$\}$
(c) $L = \{w : w \in (a + b)^*$ has neither aa nor bb as a substring$\}$
(d) None of these

Q.22 Which one of the following languages is regular?
(a) $L = \{a^n b^m : n > m \geq 0\}$
(b) $L = \{a^n b^n : 0 < n \leq k$, where k is fixed natural number$\}$
(c) $L = \{a^n b^n c^m : m, n \geq 0\}$
(d) None of these

Q.23 Which one of the following languages is not regular?
(a) $L = \{a^n b^m : n, m \geq 0\}$
(b) $L = \{a^n b^n : n \in \{1, 2, 3, ...\}\}$
(c) $L = \{a^p b^q c^r : p, q, r \geq 0\}$
(d) None of these

Q.24 Which one is incorrect?
(a) $L = \bigcup_{i=1}^{n} L_i$ is regular language, where L_i is a regular language for $i = 1, 2, ..., n$.
(b) $L = L_1 \cap L_2 \cap ... \cap L_n$ is regular language, where $L_1, L_2, ... L_n$ are regular language.
(c) $L = \Sigma^* - L_1 - L_2 ... L_n$ is regular language, where $L_1, L_2, ... L_n$ are regular language over Σ.
(d) None of these

Q.25 If L_1 and L_2 are two regular languages, then concatenation of L_1 and L_2 is written as
(a) $L_1 L_2$ (b) $L_1.L_2$
(c) Both (a) and (b) (d) None of these

Q.26 Let $r = 1(1 + 0)^*$, $s = 11^* 0$ and $t = 1^* 0$ be three regular expressions. Which one of the following is true?
(a) $L(s) \subseteq L(r)$ and $L(s) \subseteq L(t)$
(b) $L(r) \subseteq L(s)$ and $L(s) \subseteq L(t)$
(c) $L(s) \subseteq L(t)$ and $L(s) \subseteq L(r)$
(d) $L(t) \subseteq L(s)$ and $L(s) \subseteq L(r)$

Q.27 Which one of the following is TRUE ?
(a) $(r^*)^* = r$ (b) $(r^* + s^*)^* = (r + s)^*$
(c) $(r + s)^* = r^* + s^*$ (d) $r^* s^* = r^* + s^*$

Q.28 In some programming language, an identifier is permitted to be a letter following by any number of letters of digits. If L and D denote the set of letters and digits respectively, which of the following expression defines an identifier?
(a) $(L \cup D)^+$ (b) $L(L \cup D)^*$
(c) $(LD)^*$ (d) $L(LD)^*$

Q.29 Consider a grammar with the following productions:
$S \rightarrow aab \mid bac \mid aB, B \rightarrow aS \mid a, aS \rightarrow bab$
The grammar is
(a) Context-free (b) Regular
(c) Context-sensitive (d) None of these

Q.30 A grammar G can be described by
(a) (V, T, S, P) (b) (V, S, P)
(c) (V, T) (d) both (a) and (c)

Q.31 The word "FORMAL" in formal languages means
(a) No meaning
(b) The symbols used have well-defined meaning
(c) Only strings are significant
(d) None of the above

Q.32 If G = ({S}, {a}, {S → SS}, S), then language L(G) is
(a) ϕ (b) aa(a)*
(c) (aa)* (d) None of these

Q.33 The grammar G : S → aS | bS | a | b generates
(a) $(a + b)^+$ (b) $(a + b)(a + b)^*$
(c) $(a + b)^*(a + b)$ (d) All of these

Q.34 The grammar S → aSb | ε generates
(a) $\{a^n b^n : n \geq 0\}$
(b) $\{a^n b^n : n \geq 1\}$
(c) $\{a^n b^m : m \geq 1, n \geq 0\}$
(d) $\{a^n b^m : m \geq 0, n \geq 1\}$

Q.35 The regular languages are accepted by
(a) Finite automaton
(b) Pushdown automaton only
(c) Turing machine only
(d) None of these

Q.36 The regular languages are accepted by
(a) Finite automaton with stack
(b) Pushdown automaton with finite stack
(c) Turing machine
(d) All of the above

Q.37 The CSLs are accepted by
(a) Finite automaton
(b) Pushdown automaton only
(c) Turing machine with read head only
(d) Linear bounded automata

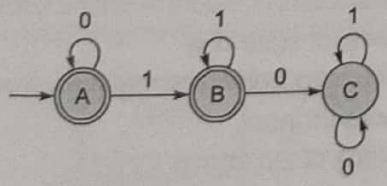
Q.38 The string created by the grammar $S \to aB \mid bA$, $A \to aS \mid a$, $B \to bS \mid b$ have
(a) Odd number of a's and b's
(b) Equal number of a's and b's
(c) Even number of a's and b's
(d) Even number of a's and odd number of b's

Q.39 Choose the regular productions
(a) $S \to aS \mid a$
(b) $S \to aaSbb \mid \varepsilon$
(c) $S \to aA \mid bB$, $A \to \varepsilon \mid aS$, $B \to \varepsilon \mid bS$
(d) Both (a) and (c)

Q.40 If for a grammar G and a language L, L = L(G), then
(a) All the words of L(G) are in L
(b) All words of L are in L(G)
(c) All words of L(G) are in L and all words of L are in L(G)
(d) All words generated by G are in L

Q.41 Choose the correct statement
(a) All regular grammars are CFG
(b) All CFGs are CSG
(c) Regular grammars are most restricted grammars
(d) All the above

Q.42 The regular expression for the language recognized by the following finite state automata is



(a) $0^* + 0^* 1^+$
(b) $0^* + 0^*1^+ + 0^* 1^+(0 + 1)^*$
(c) $0^* + 11^*$
(d) None of these

Q.43 Which of the following definitions below generates the same language as L, where $L = \{x^n y^n$ such that $n \geq 1\}$?
1. $E \to xEy \mid xy$
2. $xy + (x^+ xyy^+)$
3. $x^+ y^+$

(a) 1 only (b) 1 and 2 only
(c) 2 and 3 only (d) 2 only

Q.44 Let $L \subseteq \Sigma^*$, where $\Sigma = \{a, b\}$. Which of the following is true?
(a) $L = \{x \mid x$ has an equal number of a's and b's$\}$ is regular
(b) $L = \{a^n b^n \mid n \geq 1\}$ is regular
(c) $L = \{x \mid x$ has more a's than b's$\}$ is regular
(d) $L = \{a^m b^n \mid m, n \geq 1\}$ is regular

Q.45 Match **List-I** with **List-II** and select the correct answer using the codes given below the lists:

List-I
A. Finite automaton
B. Turing machine
C. Pushdown automaton
D. Linear bounded automaton

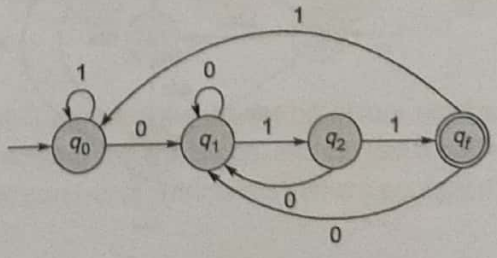
List-II
1. Type 0 language
2. Type 1 language
3. Type 3 language
4. Type 2 language

Codes:

	A	B	C	D
(a)	3	1	4	2
(b)	4	1	3	2
(c)	3	4	1	2
(d)	3	1	2	4

Q.46 The following grammar $S \to bS \mid b \mid Aa$, $A \to abc$ is
(a) Type 3 (b) Type 2
(c) Type 1 (d) Type 0

Q.47 Consider the following finite state automata M.



Which of the following regular expressions denotes the set of words accepted by M?

(a) $011(0+1)^*$ (b) 1^*00^*11

(c) $(0+1)^*011$ (d) None of these

Q.48 Choose the incorrect statement:

(a) A-productions $A \to a_1a_2 \ldots a_nA \mid a_1a_2 \ldots a_n$, where a_i is terminal for $i = 1, 2, \ldots, n$ is right linear grammar

(b) S-productions $S \to wS \mid w$ or $S \to w \mid Sw$, where w is a terminal string such that $1 \le |w| \le n$ for some fixed $n \ge 1$ is regular grammar

(c) C language is regular language

(d) Both (b) and (c) are incorrect

Q.49 Which one of the following languages is not regular?

(a) $L = \{a^{2n} : n \ge 0\}$

(b) $L = \{(ab)^n : n \ge 1\}$

(c) $L = \{a^nb^n : n = 1, 2, 3, \ldots, 1000\}$

(d) None of these

Q.50 Which one of the following languages is not regular?

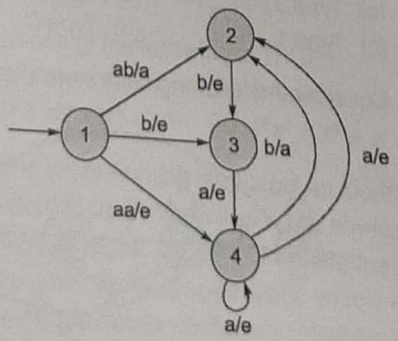
(a) $L = \{w : w \in (a+b)^*\}$ and has even number of a's and odd number of b's}

(b) $L = \{w : w \in (a+b)^*$ and has either "*aaa*" or "*bb*" as substring}

(c) $L = \{w : w \in (a+b)^*$ and has neither "*aa*" nor "*bb*" as substring}

(d) None of the above

Q.51 Consider the following finite state machine. The transition arrows/edges are labeled with input/output.



Which one of the following is true about the above finite state machine? (Assume e is the null string)

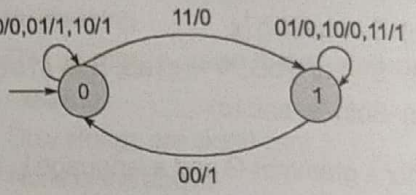
(a) On input string $w \in (a+b)^*$, produces output a^n, where n is the number of occurrence substring "*aa*" in *w*.

(b) On input string $w \in (a+b)^*$, produces output a^n, where n is the number of occurrence substring "*ab*" in *w*.

(c) On input string $w \in (a+b)^*$, produces output a^n, where n is the number of occurrence substring "*ba*" in *w*.

(d) None of these

Q.52 Consider the following finite state machine:



The above finite state machine outputs as

(a) Half adder

(b) 2's complementary machine

(c) Full adder

(d) None of these

Q.53 Choose the correct statement.

(a) Moore and Mealy machines are FA with output capability

(b) Moore machine is more powerful than FA

(c) Mealy machine is more powerful than FA

(d) Moore and Mealy machines are same on FA

Q.54 The major difference between a Moore and a Mealy machine is that

(a) The output of Moore machine depends upon present state and input

(b) The output of Moore machine depends upon present state only

(c) The output of Moore machine depends upon present input

(d) None of the above

Q.55 Choose the correct statement:

(a) Output length of equivalent Moore machine is one greater than output length of Mealy machine

(b) Output length of equivalent Mealy is equal to output length of Moore machine
(c) Output length of Moore and Mealy machine is equal
(d) Both (a) and (b) are correct

Q.56 If an input string w has n symbols and can be recognized by a Mealy machine M_1 and equivalent Moore machine M_2 then number of output symbols by M_1 and M_2 are respectively
(a) n, n (b) $n+1, n$
(c) $n, n+1$ (d) $n+1, n+1$

Q.57 If the input is null string then output produced by a Mealy machine is
(a) No output
(b) Depends on starting state
(c) Depends on given machine
(d) Can't decide

Q.58 If the input is null string then output produced by a Moore machine is
(a) No output
(b) Depends on starting state
(c) Depends on given machine
(d) Can't decide

Q.59 The output of Moore machine does not depend on
(a) State (b) Input
(c) State and Input (d) None of these

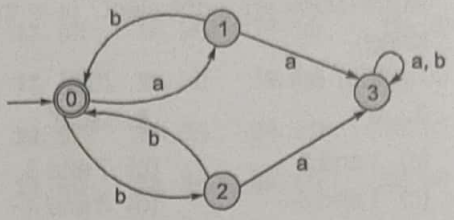
Q.60 The output of Mealy machine depends on
(a) State (b) Input
(c) State and Input (d) None of these

Q.61 If a DFA is represented by the following transition table then how many states does the corresponding minimal DFA contains?

State	Input 0	Input 1
(start) → A	B	C
B	B	D
C	B	C
D	B	E
(final) → E	B	C

(a) 2 (b) 3
(c) 4 (d) 5

Q.62 Which language does the following DFA accept?



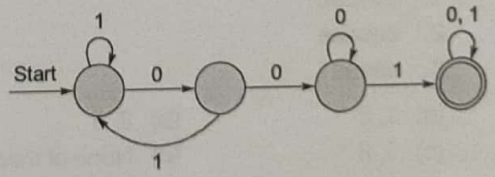
(a) $(ab)^*$ (b) $(ab + bb)^*$
(c) $(ab + ba)^*$ (d) $(aa + bb)^*$

Q.63 Let $r_1 = (a + b^2)^*$, $r_2 = (a^* + b^*)^*$, $r_3 = (a^2 + b)^*$. Which of the following is true?
(a) $L(r_1) \subseteq L(r_2)$ and $L(r_3) \subseteq L(r_2)$
(b) $L(r_2) \subseteq L(r_1)$ and $L(r_2) \subseteq L(r_3)$
(c) $L(r_1) = L(r_3) \subseteq L(r_2)$
(d) $L(r_1) \cup L(r_3) = L(r_2)$

Q.64 Let $r_1 = (b^*\, ab^*\, ab^*\, ab^*)^*$, $r_2 = (b^*\, ab^*\, ab^*)^*$. What is $L(r_1) \cap L(r_2)$?
(a) $L[(b^*\, ab^*\, ab^*\, ab^*)^*]$
(b) $L[(b^*\, ab^*\, ab^*)^*]$
(c) $L[(b^*\, ab^*\, ab^*)^6]$
(d) $L[(b^*\, ab^*\, ab^*\, ab^*\, ab^*\, ab^*)^*]$

Q.65 Let L = {W : W has $3k + 1$ b's, $\forall k \geq 0$}, construct a minimized finite automata D accepting L. How many states are there in D?
(a) 4
(b) 3
(c) 2
(d) The language is not regular

Q.66 Consider the following DFA automation M.



Let S denotes the seven bit binary strings in which the first, the fourth, and the last bits are 1. The number of strings in S that are accepted by M is

(a) 1 (b) 5
(c) 7 (d) 8

Q.67 Which two of the following four regular expressions are equivalent? (ε is the empty string).

1. $(00)^*(\varepsilon + 0)$
2. $(00)^*$
3. 0^*
4. $0(00)^*$

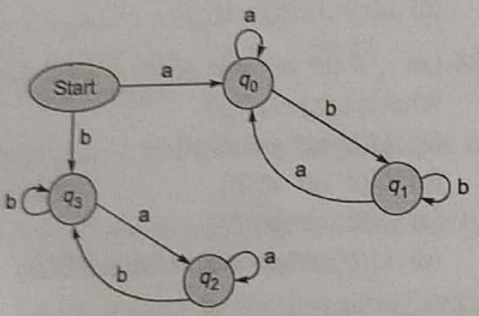
(a) 1 and 2 (b) 2 and 3
(c) 1 and 3 (d) 3 and 4

Q.68 What is the regular expression for the language generated by $S \rightarrow aS \mid bA,\ A \rightarrow d \mid ccA$

(a) a^*bd (b) $a^*(bd)(bcc)^*d$
(c) $a^*b(cc)^*d$ (d) None of these

Common Data (Q.69 to Q.71):

Consider the transition diagram of an DFA as given below:



Q.69 Which should be the final state(s) of the DFA if it should accept strings starting with 'a' and ending with 'b'?

(a) q_0 (b) q_1
(c) q_0, q_1 (d) q_3

Q.70 Which of the following strings will be accepted it q_0 and q_1 are accepting states?

1. ababab
2. babaaa
3. aaaba

(a) 1, 2 (b) 2, 3
(c) 1, 3 (d) None of these

Q.71 Which of the following represents the set of accepting states if the language to be accepted that contain all strings having the same starting and ending symbols?

(a) $\{q_0\}$ (b) $\{q_0, q_3\}$
(c) $\{q_3\}$ (d) $\{q_3, q_1\}$

Q.72 Which of the following can be recognized by a DFA?

(a) The number 1, 2, 4, ---2^n--- written in binary
(b) The number 1, 2, 4 --2^n--- written in unary
(c) The set of binary strings in which the number of 0's is same as the number of 1's
(d) The set {0, 101, 11011, 1110111, ----}

Q.73 The given transition table is for the FSM that accepts a string if it ends with 'aa'. Which is the final state?

δ	a	b
start	q_0	q_2
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_0	q_2

(a) q_0
(b) q_1
(c) q_2
(d) Can't be determined

Q.74 A minimum state DFA accepting the language (from **Q.73**) has how many states?

(a) 1 (b) 2
(c) 3 (d) 4

Common Data (Q.75 to Q.78):

Which is the equivalent regular expression (RE) for the following?

Q.75 Strings having exactly one occurrence of ab and ba

(a) $(a + b)^*(ab + ba)(a + b)^*$
(b) $(ab)^+(a + b)^*(ba)^+$
(c) Both (a) and (b)
(d) None of these

Q.76 Strings having at most one occurrence of ab or ba but not both

(a) $a^*b^* + b^*a^*$
(b) $b^*ab + a^*ba$
(c) Both (a) and (b)
(d) None of these

Q.77 Strings in which every group of three symbols should contain atleast one a.

(a) $[(a + b)(a + b)a]^*$

(b) $[(a + b)(a + b)a]^* [(a + b)(a + b)a]^*$

(c) $[(\varepsilon + b + bb)a]^* [\varepsilon + b + bb]$

(d) $(abb)^* (bab) b^* (bba)^*$

Q.78 Strings containing at most 2 a's

(a) $b^* ab^* a$

(b) $b^* (a + \varepsilon) b^* (a + \varepsilon)$

(c) $b^* + b^* ab^* + b^* ab^* ab^*$

(d) None of these

Common Data (Q.79 and Q.80):

In each case find length of string of minimum length in $\{0, 1\}^*$ not in the language corresponding to the given R.E.

Q.79 $0^*(01^*)^*$

(a) 1 (b) 2

(c) 3 (d) None of these

Q.80 $1^* (0 + 10^{**})$

(a) 2 (b) 3

(c) 4 (d) 5

Answer Key:

1. (d)	**2.** (c)	**3.** (c)	**4.** (d)	**5.** (d)
6. (a)	**7.** (d)	**8.** (d)	**9.** (b)	**10.** (a)
11. (d)	**12.** (d)	**13.** (d)	**14.** (c)	**15.** (d)
16. (c)	**17.** (a)	**18.** (d)	**19.** (c)	**20.** (a)
21. (c)	**22.** (b)	**23.** (b)	**24.** (d)	**25.** (c)
26. (a)	**27.** (b)	**28.** (b)	**29.** (c)	**30.** (a)
31. (c)	**32.** (a)	**33.** (d)	**34.** (a)	**35.** (a)
36. (d)	**37.** (d)	**38.** (b)	**39.** (d)	**40.** (c)
41. (d)	**42.** (a)	**43.** (a)	**44.** (d)	**45.** (a)
46. (b)	**47.** (c)	**48.** (c)	**49.** (d)	**50.** (d)
51. (b)	**52.** (c)	**53.** (a)	**54.** (b)	**55.** (a)
56. (c)	**57.** (a)	**58.** (b)	**59.** (b)	**60.** (c)
61. (c)	**62.** (b)	**63.** (a)	**64.** (d)	**65.** (b)
66. (c)	**67.** (c)	**68.** (c)	**69.** (b)	**70.** (c)
71. (b)	**72.** (a)	**73.** (b)	**74.** (c)	**75.** (d)
76. (a)	**77.** (c)	**78.** (c)	**79.** (a)	**80.** (a)

CHAPTER 03

Context Free Languages & Push down Automata

3.1 Context Free Grammars

3.1.1 Definition

A context-free grammar G is a quadruple (V, T, P, S), where:

- V is the set of non-terminals. (Non-terminals used in the grammar and they do not appear in strings of the language)
- T is the set of terminals,
- P is the set of productions. It is a finite subset of $V \times (T + V)^*$.
  $P: V \rightarrow (T + V)^*$.
- S is the start symbol, $S \in T$.

Example: The following context free grammar generates all strings of balanced parentheses.

$$S \rightarrow SS \mid (S) \mid \varepsilon$$

3.1.2 Derivation of a String

1. **Left Most Derivation (LMD):** In each sentential form, the left most non-terminal is substituted with corresponding production to derive a string.
   Example: Let G be the following grammar
   $$S \rightarrow aAB \mid a$$
   $$A \rightarrow aBA \mid a$$
   $$B \rightarrow b$$
   Then LMD derivation for a string = "*aabab*" is: $S \rightarrow aAB \Rightarrow aaBAB \Rightarrow aabAB \Rightarrow aabaB \Rightarrow aabab$
2. **Right Most Derivation (RMD):** In each sentential form, the right most non-terminal is substituted with corresponding production to derive a string.
   Example: Let G be the following grammar
   $$S \rightarrow aAB \mid a$$
   $$A \rightarrow aBA \mid a$$
   $$B \rightarrow b$$
   Then RMD derivation for a string = "*aabab*" is:
   $$S \rightarrow aAB \Rightarrow aAb \Rightarrow aaBAb \Rightarrow aaBab \Rightarrow aabab$$

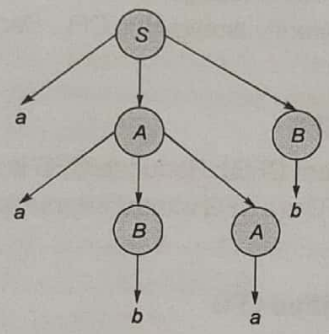
3. **Parse tree (derivation tree):** Let $G = (V, T, P, S)$ be a CFG where each production of G can be represented with a tree satisfying the following:
   (*i*) If $A \to a_1 a_2 \ldots a_n$ is a production in G, then A becomes parent of nodes labeled $a_1, a_2, \ldots a_n$.
   (*ii*) The collection of children from left to right yields $a_1 a_2 a_3 \ldots a_n$.

   If $w \in L(G)$, then it can be represented by a tree called **derivation tree** satisfying the following:
   (*i*) The root has label S (the starting symbol),
   (*ii*) All internal vertices are labeled with variables,
   (*iii*) The terminal nodes are labeled with ε or terminal symbols,
   (*iv*) If $A \to \alpha_1 \alpha_2 \ldots \alpha_n$ is a production in G, then A becomes the parent of nodes labeled $\alpha_1 \alpha_2 \ldots \alpha_n$
   (*v*) The collection of leaves from left to right yields the string w.

   Example: Let G be the following grammar

$$S \to aAB \mid a$$
$$A \to aBA \mid a$$
$$B \to b$$

   Then parse tree construction for a string = "*aabab*" is:



3.1.3 Unambiguous Grammar and Ambiguous Grammar

Unambiguous grammar: The grammar is called as unambiguous grammar if :

- Every string derived from the grammar has exactly one LMD.

or

- Every string derived from the grammar has exactly one RMD.

or

- Every string generated from the grammar has exactly one parse tree.

For every string derived from the context free grammar has:

Number of parse tree's = Number of LMD's = Number of RMD's = 1

Example: Consider the following CFG

$$S \to aS \mid a$$

Every string derived from the above grammar contain one parse tree. Hence the given grammar is unambiguous.

Ambiguous grammar: The grammar is called as ambiguous grammar if :

- There exists a string which is derived from the grammar has more than one LMD.

or

- There exists a string which is derived from the grammar has more than one RMD.

or

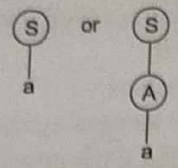
- There exists a string which is generated from the grammar has more than one parse tree.

For some string derived from grammar has: *Number of parse tree's* >1

Example: Consider the following CFG

$$S \rightarrow aS|A|a$$
$$A \rightarrow aA|a$$

Parse tree (s) for the string *a*:



For string "a", more than one parse tree is possible. Hence the given grammar is ambiguous.

3.1.4 Inherently Ambiguous Context Free Grammar

- If there is no equivalent unambiguous grammar present for a given grammar, then such a grammar is called as *inherently ambiguous grammar.*
- For a given language if there is no equivalent unambiguous CFG then such language is called as *inherently ambiguous context free language.*
- $L = \{a^k b^l c^k\} \cup \{a^m b^m c^n\}$ is inherently ambiguous CFL. Because L has no equivalent unambiguous CFG.

3.1.5 Reduced CFG and Simplified CFG

- **Reduced CFG (non-redundant CFG):** Reduced CFG is a CFG without any useless symbols.
- **Simplified CFG:** Simplified CFG is a CFG without any null-productions, unit-productions and useless symbols.

3.1.6 Conversion of CFG into Simplified CFG

The following order applied on a given CFG to convert into simplified CFG.

1. **Elimination of null productions:** $X \rightarrow \varepsilon$ is called null production where X is a variable.
2. **Elimination of unit productions:** $A \rightarrow B$ is called unit production where A,B are variables.
3. **Elimination of useless symbols:** The symbols that can't be used in any derivation are known as useless symbols.

Elimination of Null Productions

- $X \rightarrow \varepsilon$ is a null production in the grammar. All such null productions should be eliminated.
- If language contains a null string for given grammar G, after removal of null productions from G equivalent language will not contain the null string.
- Let G be a CFG and G' be an equivalent to G but without null productions. Then $L(G')=L(G)-\{\varepsilon\}$

Steps to Eliminate Null productions:

1. ***Find the nullable non-terminal set:*** A set of all non-terminals where every non terminal derives "ε" as a string either directly or indirectly.

   Example: Consider the following CFG with null productions

$$S \rightarrow aA|bB|AB$$
$$A \rightarrow aA|\varepsilon$$
$$B \rightarrow bB|\varepsilon$$

In above CFG $A \rightarrow \varepsilon$ and $B \rightarrow \varepsilon$ are null productions, so both A and B non-terminals are included in the nullable set. S can also derive ε from $S \rightarrow AB$ where A and B both derives ε.

Nullable non-terminal set = {A, B, S}

2. **Elimination of null production:** For every non-terminal in nullable set, first remove null production then add new productions in the grammar by substituting ε inplace of that particular non-terminal.

$S \to aA \mid bB \mid AB$
$A \to aA \mid \varepsilon$
$B \to bB \mid \varepsilon$

In above CFG $A \to \varepsilon$ and $B \to \varepsilon$ are eliminated. Add new productions by substituting ε.
From $S \to aA$ production, by substituting ε in place of A the new production $S \to a$ is added. Similarly $S \to b$ is added for $S \to bB$,

$S \to A$ and $S \to B$ are added for $S \to AB$,
$A \to a$ is added for $A \to aA$, and
$B \to b$ is added for $B \to bB$.

After removal of Null-productions from the above given grammar is:

$S \to aA \mid bB \mid AB \mid a \mid b \mid A \mid B$
$A \to aA \mid a$
$B \to bB \mid b$

Elimination of Unit Productions

Let $X \to Y$ be the unit production. Then unit production $X \to Y$ is eliminated by substituting all Y productions.

Example: Consider the following grammar which contain unit productions.

$S \to ab \mid bA \mid A$
$A \to aB \mid B$
$B \to bA \mid b$

After removal of unit productions ($S \to A$ and $A \to B$), the resultant grammar is:

$S \to ab \mid bA \mid \mathbf{aA} \mid \mathbf{bA} \mid \mathbf{b}$
$A \to aB \mid \mathbf{bA} \mid \mathbf{b}$
$B \to bA \mid b$

Elimination of Useless Symbols

1. **Find useless productions:** Find the non-terminals which cannot derive the string (any sequence of terminal symbols) either directly or indirectly from the grammar and then eliminate productions associated with these non-terminals.

Example: Consider the following CFG with useless productions

$S \to AB$
$A \to aC \mid aA \mid a$
$B \to bB \mid bC \mid b$
$D \to e$

A derives a, B derives b, D derives e and S derives ab but C can not derive any string. So, eliminate C from all productions which contain non-terminal C.
Therefore $A \to aC$ and $B \to bC$ are eliminated.

$S \to AB$
$A \to aA \mid a$
$B \to bB \mid b$
$D \to e$

2. **Find the useless symbols:** Find the symbols (terminals and non-terminals) which never appear in any sentential form and then eliminate the productions associated with these symbols.

- Eliminate all symbols which are not reachable through start symbol S either directly or indirectly in the grammar.
- Non-terminals A and B are reachable from S, the terminal a can be reachable from A (hence reachable indirectly from S), b is reachable from B (hence reachable indirectly from S).
- Eliminate the productions which contain the symbols D and e, because D and e are useless symbols.

Example: Consider the following CFG with useless symbols

$$S \rightarrow AB$$
$$A \rightarrow aA \mid a$$
$$B \rightarrow bB \mid b$$
$$D \rightarrow e$$

The following grammar is after elimination of useless symbols where $D \rightarrow e$ is eliminated from the grammar.

$$S \rightarrow AB$$
$$A \rightarrow aA \mid a$$
$$B \rightarrow bB \mid b$$

3.1.7 Chomsky Normal Form (CNF)

- If a language L without null string (ε) is generated by a CFG G then there exists a CNF grammar G' which is also generates the language L.
- There are two types of productions in CNF grammar:

  <variable> → <terminal>

  <variable> → <variable> <variable>
- In RHS of every production of the CNF grammar contain *only two non terminals* or *a single terminal.*

$$A \rightarrow BC$$
$$A \rightarrow a$$

- CNF is also called as *binary standard form.*
- The number of productions are required for generating x-length string from the given CNF context free grammar is: $(2x - 1)$

  Example: To produce the string "*ab*", the number of productions are required is 3.

  i.e., $2^*|ab| - 1 = 2^*2 - 1 = 3$. The productions are as follows: $S \rightarrow AB$, $A \rightarrow a$ and $B \rightarrow b$.
- Let 'G' be the given CNF grammar and 'T' be the derivation tree for some string 'x' in G. If the length of the longest path is equal to k then **yield length** $\leq 2^{k-1}$.
- For the generation of 'l' length yield, the minimum height of the derivation tree for the given CNF context free grammar is $\lceil \log_2 l \rceil + 1$ and maximum height will be l.

  Let h be the minimum height of the derivation tree, which gives yield of length 'l'.

  $l = 2^{h-1} \Rightarrow h - 1 = \log_2 l \Rightarrow h = \log_2 l + 1$.
- Let 'G' be the given CFG without null productions and unit productions and 'k' be the maximum number of symbols on the right hand side of any production.

  Then the equivalent CNF contain a **maximum of:** $(k-1)|P| + |T|$ **productions.**

  where, $|P|$ = the number of productions in 'G', $|T|$ = the number of terminals, k = maximum number of symbols on right hand side.

Example: Consider the following CFG.

$S \to aAbB$
$A \to aA \mid a$
$B \to bB \mid b$

The equivalent CNF grammar for the above CFG is

$S \to C_aX$
$X \to AY$
$Y \to C_bB$
$A \to C_aA \mid a$
$B \to C_bB \mid b$
$C_a \to a$
$C_b \to b$

In given CFG: $|P| = 5, |T| = 2, k = 4$

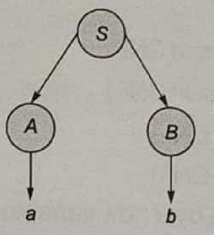
The resultant CNF CFG has 9 productions.

Example - 3.1 Find the maximum yield length for the following CNF CFG.

$S \to AB$
$A \to a$
$B \to b$

Solution:

Derivation tree for the string *ab*:



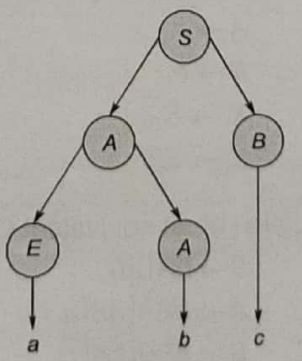
Longest path length = 2 = k
Maximum yield length = $2^{k-1} = 2$

Example - 3.2 Find the maximum yield length for the following CNF CFG.

$S \to AB$
$A \to EF$
$B \to C$
$E \to a$
$F \to b$

Solution:

Derivation tree for the string *ab*:



$k = 3$
Maximum yield length = $2^{3-1} = 4$

Conversion from CFG to CNF-CFG

- If right hand side of any production contain more than two variables then consecutive two variables can be replaced by a new variable.

  Example: $A \rightarrow XYZ$ is converted to $A \rightarrow XP$ and $P \rightarrow YZ$.

- If RHS of productions contain combination of terminal and non-terminal symbols then each terminal symbol is replaced by a new non-terminal.

  Example:

  1. $A \rightarrow aB$ is replaced by $A \rightarrow C_aB$ and $C_a \rightarrow a$
  2. $S \rightarrow ABCDEFG$ is replaced by the following productions

     $S \rightarrow AA_1$
     $A_1 \rightarrow BA_2$
     $A_2 \rightarrow CA_3$
     $A_3 \rightarrow D A_4$
     $A_4 \rightarrow E A_5$
     $A_5 \rightarrow FG$

Example - 3.3 Construction of an equivalent CNF for the following CFG.

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

Solution:

$P_1 : S \rightarrow aSa$ (Not in CNF)
$P_2 : S \rightarrow bSb$ (Not in CNF)
$P_3 : S \rightarrow a$ (In CNF)
$P_4 : S \rightarrow b$ (In CNF)

Consider P_1: Replace the terminal a by a new variable A, we get $S \rightarrow ASA$. Now, replace SA by a new variable X_1 and so, $S \rightarrow AX_1$, where $X_1 \rightarrow SA$, $A \rightarrow a$ are in CNF.

$S \rightarrow AX_1$, $X_1 \rightarrow SA$, and $A \rightarrow a$ are new productions equivalent to $S \rightarrow aSa$.

Consider P_2: Replace the terminal b by a new variable B, we get $S \rightarrow BSB$. Now, replace SB by a new variable X_2 and so, $S \rightarrow BX_2$, where $X_2 \rightarrow SB$, $B \rightarrow b$ are in CNF.

$S \rightarrow BX_2$, $X_2 \rightarrow SB$, and $B \rightarrow b$ are new productions equivalent to $S \rightarrow bSb$.

The resultant CFG in CNF:

$S \rightarrow AX_1 \mid BX_2 \mid a \mid b$,
$A \rightarrow a$,
$B \rightarrow b$,
$X_1 \rightarrow SA$,
$X_2 \rightarrow SB$.

Example - 3.4 Construction of an equivalent CNF for the following CFG.

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

Solution:

Each production which is not in CNF converted as following.

$S \to bA \Rightarrow S \to C_b A,\ C_b \to b$

$S \to aB \Rightarrow S \to C_a B,\ C_a \to a$

$A \to bAA \Rightarrow A \to C_b AA \to A \to C_b E$ and $E \to AA$

$A \to aS \Rightarrow A \to C_a S,\ A \to a$

$B \to aBB \Rightarrow B \to C_a F,\ F \to BB$

$B \to bS \Rightarrow B \to C_b S$

$B \to b$

Therefore the equivalent CNF for given CFG is

$S \to C_b A \mid C_a B$

$A \to C_b E \mid C_a S \mid a$

$B \to C_a F \mid C_b S \mid b$

$E \to AA$

$F \to BB$

$C_b \to b$

$C_a \to a$

Total productions of an equivalent CNF grammar = 12

Example - 3.5 Construction of an equivalent CNF for the following CFG.

$S \to aAD$

$A \to aB \mid bAB$

$B \to b$

$D \to d$

Solution:

Equivalent CNF for the above CFG is:

$S \to C_a E$

$E \to AD$

$A \to C_a B$

$A \to BF$

$F \to AB$

$B \to b$

$D \to d$

$C_a \to a$

Total 8 productions in CNF CFG.

3.1.8 Greibach Normal Form (GNF)

A CFG $G = (V, T, P, S)$ is in Greibach normal form (GNF) if its all productions are of type $A \to a\alpha$, where $\alpha \in V^*$ (string of variables including null string) and a is single terminal ($a \in T$).

- Every CFG production of GNF grammar contains a single terminal (a) followed by any sequence of non-terminals (α). i.e., $A \to a\alpha$
- Every CFL L without null string (ε) can be generated by GNF grammar has productions are of type $A \to a\alpha$, where $\alpha \in V^*$ and $a \in T$.

- No CNF or no GNF context free grammar generates the null string. Hence for ε-free CFG's only we can construct an equivalent CNF or an equivalent GNF.
- In GNF context free grammar, the restrictions only on the positions, but not on length of right side of a production.
- The number of productions required to generate the x-length string from the given GNF-context free grammar is: $|x|$

  Example: To produce a string "*ab*".

  Number of productions = length of the string = $|ab| = 2$. The productions are: $S \to aB$ and $B \to b$.
- **Substitution (Lemma1):** If $A \to B\alpha$ and $B \to \beta_1 |\beta_2| ... \beta_n$ then following productions are without B at left most place in RHS of production of A. It is called as substitution of B in the production of A.

$$A \to \beta_1 \alpha | \beta_2 \alpha | ... \beta_n \alpha$$

- **Elimination of Left Recursion (Lemma2):** If $A \to A\alpha_1 | A\alpha_2 | ..... A\alpha_m$ and $A \to \beta_1 | \beta_2 | ... \beta_n$ then following productions are without left recursion for $1 \le i \le n$ and $1 \le j \le m$.

$$A \to \beta_i | \beta_i Z$$
$$Z' \to \alpha_j | \alpha_j Z'$$

3.2 Context Free Language

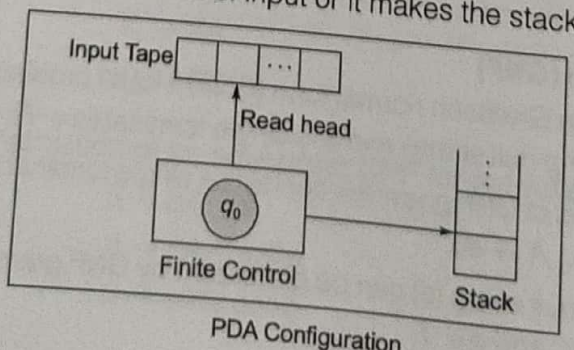
- Let $G = (V, T, P, S)$ is context free grammar. The language generated by G is $L(G) = \{w | w \in T^*$ and S derives $w\}$. A language L is a CFL iff there is a CFG G so that $L = L(G)$.
- CFL's are type-2 formal languages.
- Push down automata recognizes the context free language.
- Class of Context free languages are superset of regular languages.
- Class of Context free languages are subset of context sensitive languages.

3.3 Push Down Automata (PDA)

The push down automaton is an automaton that accepts a context free language. Push down automaton is non-deterministic by default. The language accepted by push down automaton also called as non-deterministic context free language. PDA also called as NPDA and CFL is also called as NCFL.

3.3.1 PDA Configuration

- It contains a finite control, an input tape and a stack.
- Finite control contain finite number of states.
- Read head (read pointer) is used to read the input symbol from the input tape.
- The symbols can be pushed or popped from the stack, used to keep track of previous symbols.
- For every string, finite control starts from the initial state. If the string is valid, either finite control reaches to the final state at the end of input or it makes the stack is empty.



PDA Configuration

3.3.2 PDA Acceptance

- **Acceptance by a final state (universal mechanism):** To accept the valid string, PDA reaches to the final state. This mechanism is used in finite automata, push down automata and turing machine. Hence this mechanism is called "universal mechanism".
- **Acceptance by an empty stack (local mechanism):** Empty stack indicated for acceptance of the input string instead of reaching final state. This mechanism used only in push down automata hence it is called local mechanism. No final state is needed for this mechanism.
- **Acceptance by both empty stack and final state:** String can be accepted by reaching the final state and by making the stack is empty.

CFL ≅ PDA by empty stack ≅ PDA by final state ≅ PDA by empty stack and final state ≅ CFG.

3.3.3 PDA Specifications

$PDA = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a seven tuple notation.

where, Q is the set of finite states.

Σ is an input alphabet.

Γ is the stack alphabet which contains a finite number of stack symbols.

δ is a transition function.

- (DPDA) $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to Q \times \Gamma^*$
- (NPDA) $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$ (only finite subsets)

q_0 is a start state *where* $q_0 \in Q$.

Z_0 is the initial top of stack symbol $Z_0 \in \Gamma$.

F is the set of final states $F \subseteq Q$. (If acceptance is by empty stack method then $F = \phi$)

3.3.4 PDA Operations

The operations on push down automata are as follows:

- **PUSH:** Symbol is pushed onto the stack by reading some input symbol.
  δ (present state, i/p symbol, top of stack symbol) = (next state, (i/p symbol).(top of stack symbol))
- **POP:** Symbol at the top of the stack is popped by reading some input symbol.
  δ (present state, input symbol, top of stack symbol) = (next state, ε).
- **BIPASS:** Stack content is unchanged by reading some input symbol.
  δ (present state, input symbol, top of stack symbol) = (next state, top of stack symbol).

3.3.5 PDA Models

1. **Non-deterministic PDA (NPDA or PDA)**
   (*i*) NPDA accepts non-deterministic CFL
   (*ii*) PDA is by default non-deterministic
   (*iii*) NPDA has non-deterministic transition for the input and stack combination.
   (*iv*) NPDA accepts an input if a sequence of choices leads to some final state or empty stack, depends on acceptance mechanism.
   (*v*) NPDA transition function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$. For *example* the following transition is non-deterministic on input *a* and top of the stack symbol *a*.
   $\delta(q, a, a) = \{(q, \varepsilon), (q, aa)\}$ i.e., pop operation or push operation.

2. Deterministic PDA (DPDA)
   (*i*) DPDA accepts deterministic CFL (DCFL).
   (*ii*) DPDA has exactly one transition for every input and stack combination.
   (*iii*) DPDA accepts an input if a sequence of choices leads to a final state
   (*iv*) DPDA is less powerful than PDA and every DPDA is an NPDA but every NPDA and not be a DPDA.
   (*v*) DPDA transition function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to Q \times \Gamma^*$. For *example* the following transition is deterministic on input *a*.
   $\delta(q, a, a) = (q, \varepsilon)$ *i.e.*, pop operation.

NOTE: Assume all PDA's are having by default z_0 or Z_0 as initial top of stack symbol otherwise z_0 can be pushed with the special transition $\delta(q_0, \varepsilon, \varepsilon) = (q_0, z_0)$.

3.3.6 PDA Construction

Example - 3.6 Construct the PDA that accept $L = \{ a^n b^n | n \geq 1\}$

Solution:

(*i*) PDA transitions: PDA = $(\{q_0, q_1, q_f\}, \{a,b\}, \Gamma, \delta, q_0, z_0, q_f)$

$\left.\begin{aligned}\delta(q_0, a, z_0) &= (q_0, az_0)\\ \delta(q_0, a, a) &= (q_0, aa)\end{aligned}\right\}$ all a's are pushed onto stack

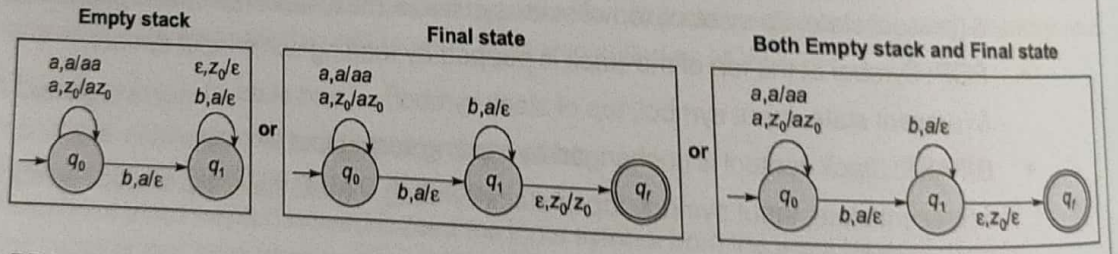
$\left.\begin{aligned}\delta(q_0, b, a) &= (q_1, \varepsilon)\\ \delta(q_1, b, a) &= (q_1, \varepsilon)\end{aligned}\right\}$ One '*a*' is popped for each '*b*'

$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$ Acceptance by empty stack. (No final state is needed)

$= (q_f, z_0)$ Acceptance by final state. (No need to empty the stack)

$= (q_f, \varepsilon)$ Acceptance by both empty stack and final state.

(*ii*) PDA transition diagram:



(*iii*) PDA transition table:

PDA Acceptance by Empty Stack

Σ_ε	a			b			ε		
Γ	a	z_0	ε	a	z_0	ε	a	z_0	ε
$\to q_0$			(q_0, a)	(q_1, ε)					
q_1				(q_1, ε)				(q_1, ε)	

or

	a	b	ε
$\to q_0$	q_0, ε/a	q_1, a/ε	q_0, ε/z_0
q_1		q_1, a/ε	q_1, z_0/ε

Example - 3.7 Construct the PDA that accept $L = \{a^n b^n c^m | m, n \geq 1\}$

Solution:

(*i*) PDA transitions: PDA = $(\{q_0, q_1, q_2, q_f\}, \{a,b,c\}, \Gamma, \delta, q_0, z_0, q_f)$

$\left.\begin{aligned}\delta(q_0, a, z_0) &= (q_0, az_0)\\ \delta(q_0, a, a) &= (q_0, aa)\end{aligned}\right\}$ All input a's are pushed onto stack

$\left.\begin{aligned}\delta(q_0, b, a) &= (q_1, \varepsilon)\\ \delta(q_1, b, a) &= (q_1, \varepsilon)\end{aligned}\right\}$ One 'a' is popped for each input 'b'

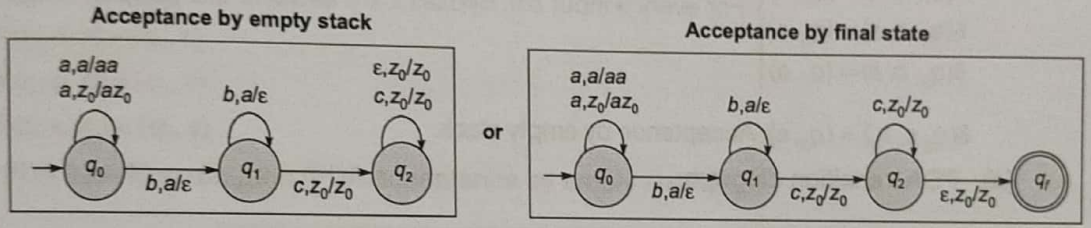
$\left.\begin{aligned}\delta(q_1, c, z_0) &= (q_2, z_0)\\ \delta(q_2, c, z_0) &= (q_2, z_0)\end{aligned}\right\}$ All input c's are ignored (no operation performed on the stack)

$\delta(q_2, \varepsilon, z_0) = (q_2, \varepsilon)$ Acceptance by empty stack.

$= (q_f, z_0)$ Acceptance by final state.

$= (q_f, \varepsilon)$ Acceptance by both empty stack and final state.

(*ii*) PDA transition diagram:



Example - 3.8 Construct the PDA that accept $L = \{a^m b^{m+n} c^n | m, n \geq 1\}$

Solution:

(*i*) PDA transitions: PDA = $(\{q_0, q_1, q_2, q_f\}, \{a,b,c\}, \Gamma, \delta, q_0, z_0, q_f)$

$\left.\begin{aligned}\delta(q_0, a, z_0) &= (q_0, az_0)\\ \delta(q_0, a, a) &= (q_0, aa)\end{aligned}\right\}$ All input a's are pushed onto stack

$\left.\begin{aligned}\delta(q_0, b, a) &= (q_1, \varepsilon)\\ \delta(q_1, b, a) &= (q_1, \varepsilon)\end{aligned}\right\}$ One 'a' is popped for each input 'b'

$\left.\begin{aligned}\delta(q_1, b, z_0) &= (q_1, bz_0)\\ \delta(q_1, b, b) &= (q_1, bb)\end{aligned}\right\}$ Remaining all input b's are pushed onto the stack

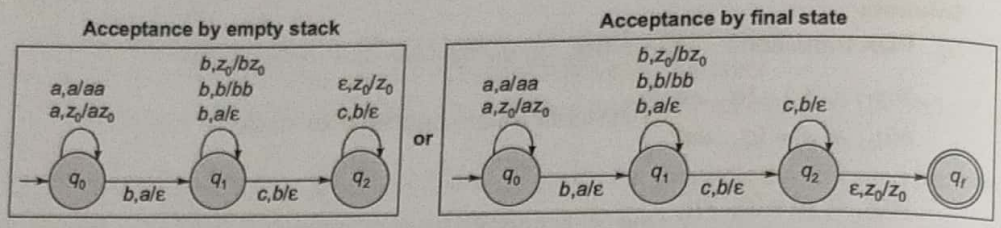
$\left.\begin{aligned}\delta(q_1, c, b) &= (q_2, \varepsilon)\\ \delta(q_2, c, b) &= (q_2, \varepsilon)\end{aligned}\right\}$ Pop one b for each input c

$\delta(q_2, \varepsilon, z_0) = (q_2, \varepsilon)$ Acceptance by empty stack.

$= (q_f, z_0)$ Acceptance by final state.

$= (q_f, \varepsilon)$ Acceptance by both empty stack and final state.

(*ii*) PDA transition diagram:



Example - 3.9 Construct PDA for $L = \{a^n b^{3n} | n \geq 1\}$

Solution:

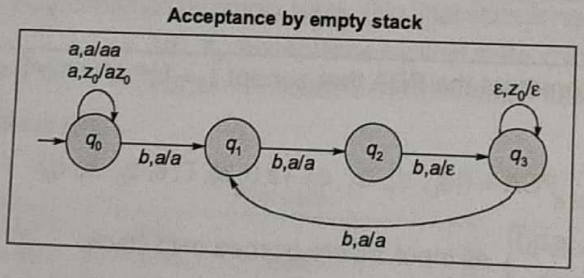
(*i*) PDA transitions: PDA = $(\{q_0, q_1, q_2, q_3\}, \{a,b\}, \Gamma, \delta, q_0, Z_0, \phi)$

$$\left.\begin{aligned}\delta(q_0, a, Z_0) &= (q_0, aZ_0)\\ \delta(q_0, a, a) &= (q_0, aa)\end{aligned}\right\}$$ All a's are pushed

$$\left.\begin{aligned}\delta(q_0, b, a) &= (q_1, a)\\ \delta(q_1, b, a) &= (q_2, a)\\ \delta(q_2, b, a) &= (q_3, \varepsilon)\\ \delta(q_3, b, a) &= (q_1, a)\end{aligned}\right\}$$ For every 3 input b's: Bypass 2 b's and one a is popped for the third b

$\delta(q_3, \varepsilon, Z_0) = (q_3, \varepsilon)$; Acceptance by empty stack.

(*ii*) PDA transition diagram:



Example - 3.10 Construct PDA for $L = \{w \in (a + b)^* | \text{number of a's}(w) \geq \text{number of b's } (w)\}$

Solution:

PDA transitions: PDA = $(\{q_0\}, \{a,b\}, \Gamma, \delta, q_0, Z_0, \phi)$

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, b, Z_0) = (q_0, bZ_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, a, b) = (q_0, \varepsilon)$
$\delta(q_0, b, a) = (q_0, \varepsilon)$
$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, \varepsilon, a) = (q_0, \varepsilon)$; Stack contain a's after reading entire input

$\delta(q_0, \varepsilon, Z_0) = (q_0, \varepsilon)$; Acceptance by empty stack.

Example - 3.11 Construct PDA that accept all strings of balanced parenthesis

Solution:

PDA transitions: PDA = $(\{q_0\}, \{(\ ,\ )\}, \Gamma, \delta, q_0, z_0, \phi)$

$\delta(q_0, (, z_0) = (q_0, (z_0)$

$\delta(q_0, (, () = (q_0, (()$

$\delta(q_0,), () = (q_0, \varepsilon)$

$\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$

Example - 3.12 Construct PDA that accepts a language $L = \{ww^R | w \in (a + b)^+\}$

Solution:

(*i*) PDA transitions: PDA = $(\{q_0, q_1\}, \{a, b\}, \Gamma, \delta, q_0, z_0, \phi)$

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, a, a) = (q_0, aa)$ or (q_1, ε)

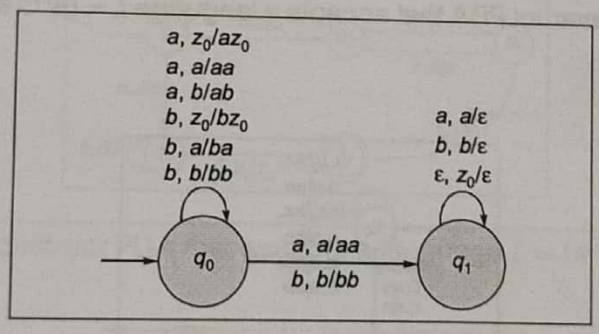
$\delta(q_0, b, b) = (q_0, bb)$ or (q_1, ε)

$\delta(q_1, a, a) = (q_1, \varepsilon)$

$\delta(q_1, b, b) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$

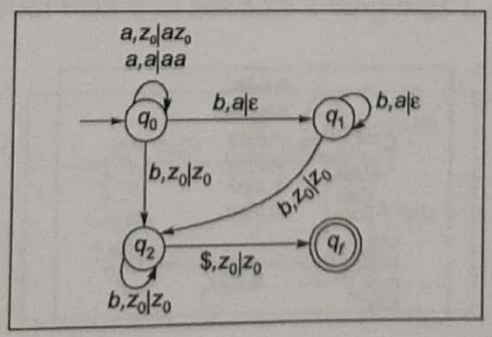
(*ii*) PDA transition diagram (PDA acceptance by empty stack):



Example - 3.13 Construct PDA that accepts a language $L = \{a^m b^n | m < n\}$

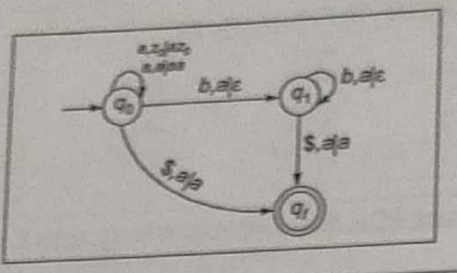
Solution:

Assume $ is end of the input string.



Example-3.14 Construct PDA that accepts a language $L = \{a^m b^n | m > n\}$

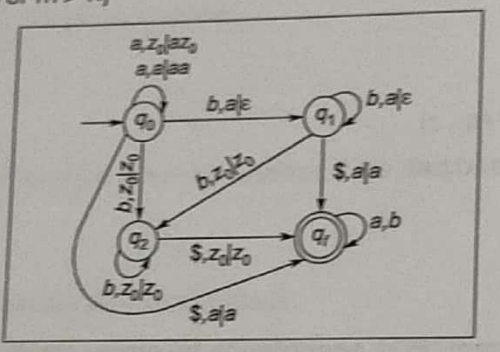
Solution:



Example-3.15 Construct PDA that accepts a language $L = \{a^m b^n | m \neq n\}$

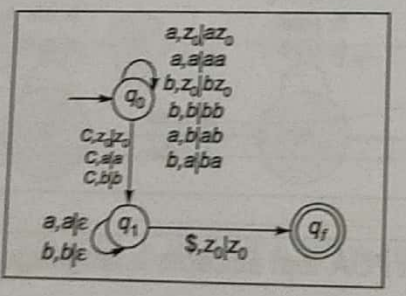
Solution:

$L = \{a^m b^n | m < n \text{ or } m > n\}$



Example-3.16 Construct PDA that accepts a language $L = \{w C w^R | w \in \{a, b\}^* C \text{ is a special input symbol}\}$

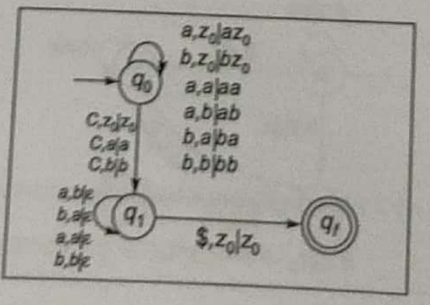
Solution:



Example-3.17 Construct PDA that accepts a language

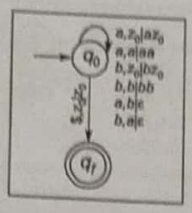
$$L = \{w_1 C w_2 | w_1, w_2 \in \{a + b\}^*, |w_1| = |w_2|\}$$

Solution:



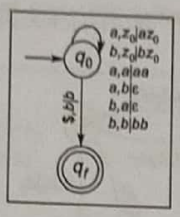
Example - 3.18 Construct PDA that accepts a language $L = \{w \mid w \in (a + b)^*$, w has equal number of a's and b's$\}$

Solution:



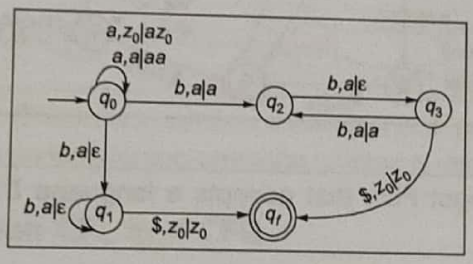
Example - 3.19 Construct PDA that accepts a language $L = \{w \mid w \in (a + b)^*$, w has more b's than a's$\}$

Solution:



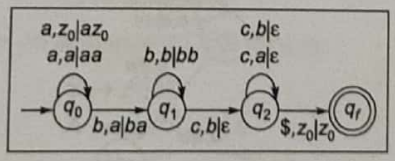
Example - 3.20 Construct PDA that accepts a language $L = \{a^m b^n \mid (m=n)$ or $(2m=n)$, $m,n \geq 1\}$

Solution:



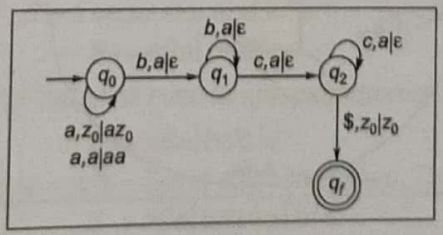
Example - 3.21 Construct PDA that accepts a language $L = \{a^m b^n c^k \mid m + n = k,\ m,\ n,\ k \geq 1\}$

Solution:



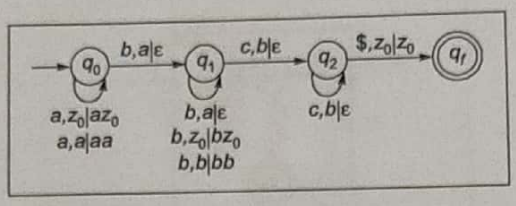
Example - 3.22 Construct PDA that accepts a language $L = \{a^m b^n c^k \mid m = n + k,\ m,\ n \geq 1\}$

Solution:



Example - 3.23 Construct PDA that accepts a language $L = \{a^m b^n c^k | m + k = n, m, n \geq 1\}$

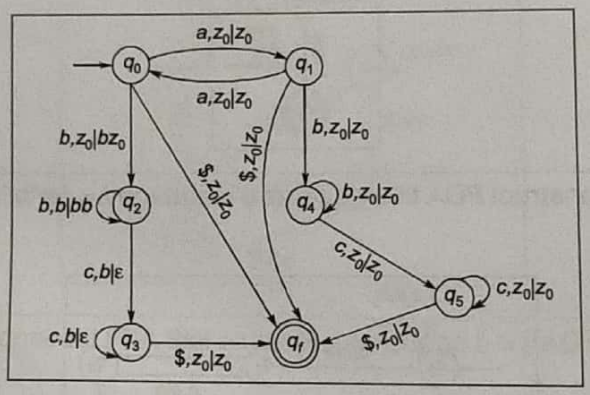
Solution:



Example - 3.24 Construct PDA that accepts a language $L = \{a^m b^n c^k |$ if (m is even) then $n = k, m \geq 0, n, k \geq 0\}$

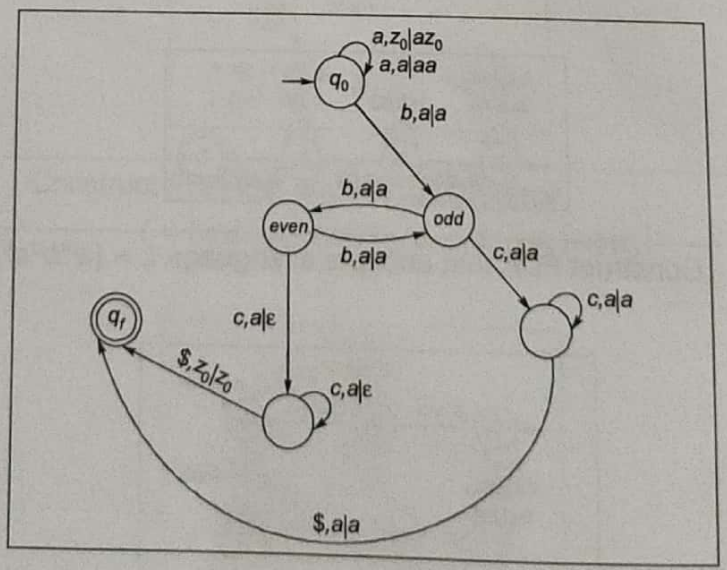
Solution:

$$L = \{a^{2i} b^n c^n | i, n \geq 0\} \cup \{a^{2i+1} b^j c^k | i, j, k \geq 0\}$$



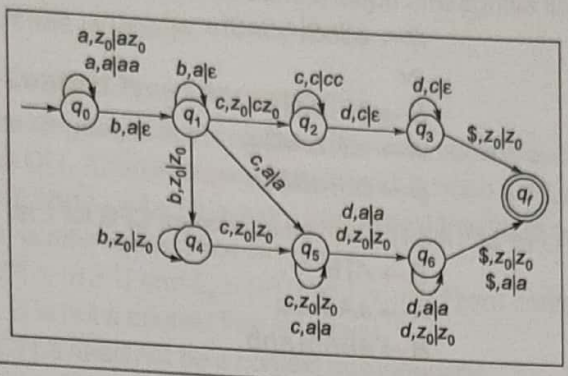
Example - 3.25 Construct PDA that accepts a language $L = \{a^m b^n c^k |$ if (n is even) then $m = k, m, n, k \geq 1\}$

Solution:



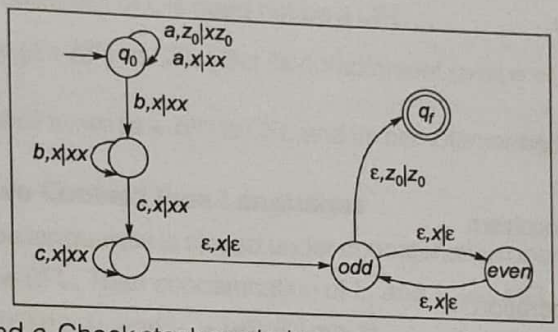
Example - 3.26 Construct PDA that accepts a language $L = \{a^m b^n c^k d^l \mid \text{if } (m = n) \text{ then } (k = l), m, n, k, l \geq 1\}$

Solution:



Example - 3.27 Construct PDA that accepts a language $L = \{a^m b^n c^k \mid (m + n + k) \text{ is odd}, m, n, k \geq 1\}$

Solution:



Push x for every a, b and c. Check stack symbols are even or odd. pop x's until z_0 appears. If x's are odd then goto final state. While performing pop operation no input is reading in the above PDA.

3.4 Equivalence between CFL and CFG

- Let $L = \{a^n b^n \mid n \geq 0\}$. Then an equivalent CFG for L is: $S \rightarrow aSb \mid \varepsilon$
- Let L be the language contain all balanced parentheses. Then an equivalent CFG for L is:

$$S \rightarrow SS \mid (S) \mid \varepsilon$$

- Let $L = \{a^m b^n \mid m \neq n\}$. Then an equivalent CFG for L is:

$$S \rightarrow A \mid B$$
$$A \rightarrow aAb \mid aA \mid a$$

(for $m > n$)

$$B \rightarrow aBb \mid Bb \mid b$$

(for $m < n$)

- Let $L = \{wow^R \mid w \in \{a, b\}^*$, o is a separator and w^R is the reverse of $w\}$. Then an equivalent CFG for L is: $S \rightarrow aSa \mid bSb \mid o$
- Let $L = \{ww^R \mid w \in \{a, b\}^*\}$ = Set of all even length palindromes. Then an equivalent CFG for L is:

$$S \rightarrow aSa \mid bSb \mid \varepsilon$$

- Let $L = \{w \mid w \in \{a, b\}^*$ and $w = w^R\}$ = Set of all palindromes. Then an equivalent CFG for L is:

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

- Let L be the set of all odd length palindromes. Then an equivalent CFG for L is:

$$S \to aSa \mid bSb \mid a \mid b$$

- Let L be the set of all strings with equal number of a's and b's. Then an equivalent CFG for L is:

$$S \to aSbS \mid bSaS \mid \varepsilon$$

or

$$S \to aB \mid bA \mid \varepsilon$$
$$A \to aS \mid bAA \mid a$$
$$B \to bS \mid aBB \mid b$$

- Let $L = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$. Then an equivalent CFG for L is:

$$S \to A \mid B$$
$$A \to aAb \mid ab$$
$$B \to aBbb \mid abb$$

3.5 Closure Properties of CFL's

- The context free languages are closed under for the following operations.
  - (*i*) Union operation
  - (*ii*) Concatenation
  - (*iii*) Kleene closure
  - (*iv*) Reversal operation
  - (*v*) Homomorphism
  - (*vi*) Inverse homomorphism
  - (*vii*) Substitution
  - (*viii*) Init or prefix operation
  - (*ix*) Right quotient with regular language (CFL / Regular)
  - (*x*) Cycle operation
  - (*xi*) Union with regular language (CFL $\cup$ Regular)
  - (*xii*) Intersection with regular language (CFL $\cap$ Regular)
  - (*xiii*) Right difference with regular language (CFL – Regular)
- The context free languages are not closed under for the following operations.
  - (*i*) Intersection
  - (*ii*) Complement
  - (*iii*) Difference, symmetric difference (XOR), NAND and NOR.
  - (*iv*) Subset
  - (*v*) Superset
  - (*vii*) Left difference with regular language (Regular – CFL)

3.5.1 Union of Two Context Free Languages

The class of context free languages is closed under union operation.

Let L_1 be CFL and L_2 be CFL. Then union of L_1 and L_2 is a CFL.

Example: Let $L_1 = \{a^n b^n \mid n \geq 1\}$ and $L_2 = \{a^n b^{2n} \mid n \geq 1\}$

CFG for L_1 : $A \to aAb \mid ab$

CFG for L_2 : $B \to aBbb \mid abb$

CFG for $L_1 \cup L_2$: $S \to A \mid B$
$A \to aAb \mid ab$
$B \to aBbb \mid abb$

Hence $L_1 \cup L_2$ is context free language.

3.5.2 Intersection of Two Context Free Languages

The class of context free languages is not closed under intersection operation.
Let L_1 be CFL and L_2 be CFL. Then intersection of L_1 and L_2 need not be a CFL.
Example: $L_1 = \{a^n \mid n \geq 0\}$ and $L_2 = \{a^n \mid n \text{ is even}\}$ are context free languages.
$L_1 \cap L_2 = \{a^n \mid n \text{ is even}\}$ is also context free language.
Example: $L_1 = \{a^n b^n c^m \mid n, m \geq 1\}$ and $L_2 = \{a^n b^m c^n \mid n, m \geq 1\}$ are context free languages.
$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$ is not a context free language.
So intersection of two CFL's need not be a context free language.

3.5.3 Complement of a Context Free Language

The class of context free languages is not closed under complement operation.
Let L be CFL. Then complement of L is need not be a CFL.

Example: $L = \{\overline{ww} \mid w \in (a+b)^*\}$ is CFL . But its complement $\{ww \mid w \in (a+b)^*\}$ is not a CFL.

$L = \{ww^R \mid w^R \text{is reverse of } w,\ w \in (a+b)^*\}$ is CFL and its complement $\{\overline{ww^R} \mid w \in (a+b)^*\}$ is also CFL.

3.5.4 Concatenation of Two Context Free Languages

The class of context free languages is closed under concatenation operation.
Let L_1 be CFL and L_2 be CFL. Then concatenation of L_1 and L_2 is a CFL.
Example: Let $L_1 = \{a^n b^n \mid n \geq 1\}$ and $L_2 = \{c^m d^m \mid m \geq 1\}$

CFG for L_1: $A \to aAb \mid ab$
CFG for $L2$: $B \to cBd \mid cd$
CFG for $L_1 \cdot L_2$: $S \to AB$
$A \to aAb \mid ab$
$B \to cBd \mid cd$

Hence $L_1 \cdot L_2$ is context free language.

3.5.5 Kleene Closure of a Context Free Language

The class of context free languages is closed under Kleene closure operation.
Let L be CFL. Then Kleene closure of L (L^*) is a CFL.
Example: $L = \{a^n b^n \mid n \geq 1\}$ is context free language.

CFG of L: $A \to aAb \mid ab$
$L^* = \{(a^n b^n)^* \mid n \geq 1\}$
CFG of L^*: $S \to \varepsilon \mid AS$
$A \to aAb \mid ab$

3.5.6 Reversal of a Context Free Language

The class of context free languages is closed under reversal operation.
Let L be CFL. Then reversal of L (L^R) is a CFL.

Example: $L = \{a^n b^n \mid n \geq 1\}$ is context free language.

CFG of L: $S \rightarrow aSb \mid ab$

By reversing the right hand side of every production, the reversal grammar can be obtained.

CFG of L^R: $S \rightarrow bSa \mid ba$

$\Rightarrow$ $L^R = \{b^n a^n \mid n \geq 1\}$ is context free language

Hence context free languages are closed under reversal operation.

3.5.7 Substitution of a Context Free Language

The class of context free languages is closed under substitution operation.

Let L be CFL. Then substitution of L is a CFL.

Example: Let $L = \{a^n b^n \mid n \geq 1\}$, $f(a) = 0^k 1^k$ and $f(b) = 1^m 0^m$

CFG for L: $A \rightarrow aAb \mid ab$

CFG for $f(L)$: $A \rightarrow XAY \mid XY$

$X \rightarrow 0X1 \mid \varepsilon$

$Y \rightarrow 1Y0 \mid \varepsilon$

Hence $f(L)$ is context free language.

3.5.8 Homomorphism of a Context free Language

The class of context free languages is closed under homomorphism operation.

Let L be CFL. Then homomorphism of L is a CFL.

Example: Let $L = \{a^n b^n \mid n \geq 1\}$, $h(a) = 01$ and $h(b) = 111$

CFG for L: $A \rightarrow aAb \mid ab$

CFG for $h(L)$: $A \rightarrow 01A111 \mid 01111$

Hence $h(L)$ is context free language.

3.6 Closure Properties of DCFL's

- The deterministic context free languages are closed under for the following operations.
  - (*i*) Complement
  - (*ii*) Inverse homomorphism
  - (*iii*) Union with regular language (DCFL $\cup$ Regular)
  - (*iv*) Intersection with regular language (DCFL $\cap$ Regular)
  - (*v*) Right difference with regular language (DCFL – Regular)
  - (*vi*) Right quotient with regular language (DCFL / Regular)
  - (*vii*) Left difference with regular language (Regular – DCFL)
  - (*viii*) Prefix
  - (*ix*) Min
  - (*x*) Max
- The deterministic context free languages are not closed under for the following operations.
  - (*i*) Union
  - (*ii*) Intersection
  - (*iii*) Difference
  - (*iv*) Concatenation

(v) Kleene closure
(vi) Positive closure
(vii) Reversal
(viii) Homomorphism
(ix) Substitution

3.7 Decision Properties of CFL's

3.7.1 Emptiness (Is L(CFG) = ϕ ?)

Check whether the start symbol(S) can derive some string or not. If S can derive some string, then the language is non-empty, otherwise it is empty language.

Checking emptiness is decidable. If simplified CFG has a start symbol then the grammar generates non-empty languages. Otherwise it generates empty language.

Example: Consider the following CFGs.

G_1: $S \to AB$, $A \to a$, $B \to b$

G_2: $S \to ABS$, $A \to a$, $B \to b$

$L(G_1)$ is a non-empty language and $L(G_2)$ is empty language.

Example - 3.28 Check whether the following CFG generates empty language or not

$S \to XY$
$X \to AX \mid AA$
$A \to a$
$Y \to BY \mid BB$
$B \to b$

Solution:

Here the start symbol 'S' is in the set of variables which can derive some string directly or indirectly.

$\{A, B, X, Y, S\}$ has all variables which derives some string.

$A \to a$, $B \to b$, $X \to AA$, $Y \to BB$. Therefore $S \to XY$ derives some string. So the language generated by the grammar is non-empty.

Example - 3.29 Check whether the following CFG generates empty language or not

$S \to XY$
$X \to AX$
$A \to a$
$Y \to BY \mid BB$
$B \to b$

Solution:

Here the start symbol 'S' can not be in the set of variables those can derive terminals directly or indirectly.

$\{A, B, Y\}$ has all variables which derives terminals.

$A \to a$, $B \to b$ and $Y \to BB$

Therefore S can not derive any terminal. So the language is empty.

3.7.2 Finiteness (Is L(CFG) Finite?)

Steps to check the finiteness problem:

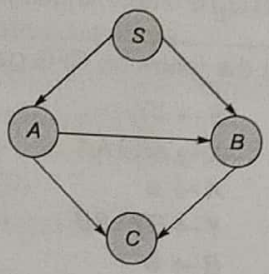
1. Convert the context free grammar to CNF-CFG.
2. Eliminate useless symbols in the CNF-CFG.
3. Construct the transition graph for the CNF-CFG using non-terminals.
4. If any cycle is present in the transition graph, then the grammar (CFG) generates an infinite language. Otherwise the given grammar (CFG) generates a finite language

Example - 4.30 Check whether the following context free grammars generates the finite language or not

$$S \to AB$$
$$A \to BC \mid a$$
$$B \to CC \mid b$$
$$C \to a$$

Solution:

The given grammar is in CNF and no useless symbols. Now, construct the transition graph



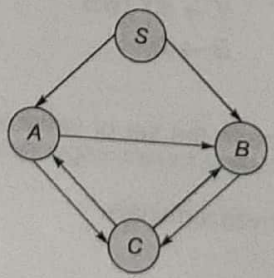
The transition graph not contains a cycle. Hence the given grammar generates the finite language.

Example - 3.31 Check whether the following context free grammars generates the finite language or not.

$$S \to AB$$
$$A \to BC \mid a$$
$$B \to CC \mid b$$
$$C \to a \mid AB$$

Solution:

Given CFG is in CNF and it has no useless symbols. Hence construct the transition graph.

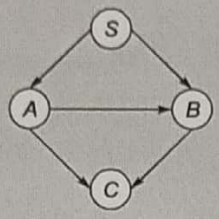


The transition graph contains a cycle. Hence the given grammar generates the infinite language.

- Rank of a variable can be computed if the dependency transition graph has no cycle. (If Grammar generates finite language then rank of variable is computed)
- Rank of a variable = Length of the longest path from that variable.
- Consider the following grammar that generates the finite language.

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

The transition graph for above grammar is



Rank of $S = R(S) = 3$ ($\because$ $S \rightarrow A \rightarrow B \rightarrow C$)
Rank of $A = R(A) = 2$ ($\because$ $A \rightarrow B \rightarrow C$)
Rank of $B = R(B) = 1$
Rank of $C = R(C) = 0$

- If 'A' has rank 'r' then any string generated from 'A' is of length *less than or equal to* 2^r.
- If $A \rightarrow BC$ is in the non-redundant CNF then the rank of A is greater than rank of B and Rank of A is greater than rank of C. i.e., $R(A) > R(B)$ and $R(A) > R(C)$.

3.7.3 Membership (Is string $w \in L$(CFG)?)

Whether the CNF context free grammar generates the string 'w'. This problem is known as "membership problem". CYK algorithm can be used to solve the membership problem.

- CYK algorithm is Dynamic programming algorithm.
- The time complexity of CYK algorithm is $O(n^3)$.
- The CYK algorithm follows the bottom-up approach.
- CYK algorithm is also called as parsing algorithm.

3.8 Non-Context Free Languages

- $L = \{a^n b^n c^n \mid n \geq 0\}$
- $L = \{a^n b^m c^n d^m \mid m, n \geq 0\}$
- $L = \{a^{n^2} \mid n \geq 0\}$
- $L = \left\{a^{2^n} \mid n \geq 0\right\}$
- $L = \{a^n \mid n \text{ is prime number}\}$
- $L = \left\{a^{n!} \mid n \geq 0\right\}$

- $L = \{a^i b^j c^k \mid i \le j \le k\}$
- $L = \{ww^R w^R \mid w \in (a+b)^*\}$
- $L = \{ww \mid w \in (a+b)^*\}$
- $L = \{0^c \mid c \text{ is composite number}\}$
- $L = \{0^i 1^j \mid j = i^2\}$
- $L = \{a^n b^n c^i \mid i \le n\}$
- $L = \{0^i 1^j 0^k \mid j = \max(i, k)\}$
- $L = \{a^n b^n c^i \mid i \ne n\}$
- $L = \{ww^R w \mid w(a+b)^*\}$

3.9 Non-Regular Languages and CFL's

- $L = \{a^n b^n \mid n \ge 1\}$
- $L = \{a^n b^n c^m \mid m,n \ge 1\}$
- $L = \{a^m b^{m+n} c^n \mid m, n \ge 1\}$
- $L = \{a^n b^{3n} \mid n \ge 1\}$
- $L = \{w \in (a+b)^* \mid \text{number of a's}(w) = \text{number of } b\text{'s } (w)\}$
- $L = \{\text{all strings of balanced parenthesis}\}$
- $L = \{ww^R \mid w \in (a+b)^+\}$
- $L = \{wcw^R \mid w \in (a+b)^+\}$
- $L = \{a^m b^m c^n d^n \mid m, n \ge 0\}$
- $L = \{a^m b^n c^n d^m \mid m, n \ge 0\}$
- $L = \{a^n c b^{2n} : n \ge 0\}$
- $L = \{wcw^R : w \in (a+b)^*\}$
- $L = \{a^m b^n : m \ne n\}$
- $L = \{ca^n b^n : n \ge 0\} \cup \{da^n b^{2n} : n \ge 0\}$
- $L = \{a^n c b^n : n \ge 0\} \cup \{a^n d b^{2n} : n \ge 0\}$

Summary

- **CFG:** Every production of CFG: $V \rightarrow (V+T)^*$
- **Derivation of a string:** LMD, RMD and parse tree.
- **Types of CFG:** Ambiguous and unambiguous grammar.
- **Reduced CFG:** A CFG without useless symbols.
- **Simplified CFG:** A CFG without *"NULL productions, Unit productions and useless symbols"*.
- **Normal forms:** CNF-CFG and GNF-CFG.
- **Push down automata:** It accepts a CFL.
- **PDA acceptance:** Empty stack, Final state, Both empty stack and final state.
- **PDA types:** NPDA and DPDA.
  DPDA is less powerful than NPDA.
  - **Equivalence:** CFLs ≅ CFGs ≅ PDA ≅ NPDA
  - **CFL closure properties:** CFLs are not closed under complementation and intersection.
  - **DCFL closure properties:** DCFLs are closed under complement.

- **Decision properties:** Emptyness, Finiteness and Membership properties are decidable for CFLs.
- Pumping Lemma is used to prove certain languages are non-CFL.
- If L is a CFL and R is a regular language then $L \cap R$ is always CFL but need not be regular language.
- CFL – Regular = CFL, CFL ∩ Regular = CFL, CFL ∪ Regular = CFL, and CFL/Regular = CFL.
- DCFL – Regular = DCFL, DCFL ∩ Regular = DCFL, DCFL ∪ Regular = DCFL.
- DCFL / Regular = DCFL, and Regular – DCFL = DCFL.
- Regular – CFL = need not be CFL.
- CFL is not closed under complement, difference, and intersection operations.
- DCFL is closed under complement.
- DCFL ≡ DPDA ≡ LR(k) grammar ($k \geq 1$)
- Every LR(k) grammar is unambigous and equivalent to a DCFL.

Student's Assignments 1

Q.1 Consider the following CFG 'G':

$$S \to aA \mid bSS \mid SS$$
$$A \to aAb \mid bAa \mid AA \mid \varepsilon$$

The language generated by G is _____.

(a) Set of all strings with atleast one 'a'
(b) Set of all strings with atleast two a's
(c) Set of all strings with atleast one more 'a' than number of b's
(d) None of these

Q.2 Which of the following language is not a deterministic context free language?

(a) Set of all binary strings where every string not contain an equal number of zeros and ones.
(b) Set of all binary strings where every string starts and ends with the same symbol, and have the same number of zeros as ones.
(c) $\{10^n 1^n \mid n > 0\} \cup \{11\ 0^K 1^{2K} \mid k > 0\}$
(d) None of these

Q.3 Consider the following CFG with a start symbol S.

$$S \to AAaSb \mid \varepsilon$$
$$A \to a \mid \varepsilon$$

Which of the following language is generated by G?

(a) $\{a^m b^n \mid m \geq n \geq 0\}$
(b) $\{a^m b^n \mid 0 \leq n \leq m \leq 2n\}$
(c) $\{a^m b^n \mid 0 \leq n \leq m \leq 3n\}$
(d) None of these

Q.4 What is the maximum possible string length that can be generated from the following CFG.

$$S \to aAb \mid aBd$$
$$A \to ab \mid Bd \mid e$$
$$B \to abe \mid d \mid f$$

(a) 5 (b) 6
(c) 7 (d) 8

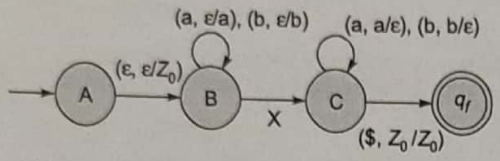
Q.5 Consider a language L which is generated by the following CFG.

$$S \to aSa \mid B$$
$$B \to \varepsilon \mid bBa$$

Find the number of strings in L such that the string length is less than 10.

(a) 15 (b) 16
(c) 17 (d) 18

Q.6 The following machine is designed with PDA acceptance by final state mechanism to accept the language L where all strings of L are odd length palindromes.



What are the transitions at X to accept L?

(a) (a, a/ε), (b, b/b)
(b) (a, a/ε), (b, b/ε)
(c) (a, ε/ε), (b, ε/ε)
(d) None of these

Q.7 Assume PDA stack is limited to 2^{10} symbols. Now stack can contain maximum of 2^{10} symbols. The language accepted by such PDA is _____.

(a) Regular language but not finite
(b) DCFL but not regular
(c) CFL but not DCFL
(d) None of these

Q.8 Let L_1 be a regular language over $\Sigma = \{a, b, c\}$ and $L_2 = \{a^n b^n c^n\}$. Concatenation of L_1 and L_2 is _____ (i.e., $L_1.L_2$)

(a) Regular language
(b) Non-regular language
(c) Context sensitive language
(d) None of these

Q.9 Consider the following CFG G.

$$S \rightarrow aA|bB|AB|\varepsilon$$
$$A \rightarrow aAb|\varepsilon$$
$$B \rightarrow bBa|\varepsilon$$

L(G) is a language generated by G. If $L_1 = \{w | w \in L(G), |w| \leq 3\}$, how many strings are in L_1?

(a) 5 (b) 6
(c) 7 (d) 8

Q.10 Consider the following CFG G.

$$S \rightarrow Aa|Bb|Cd|AB$$
$$A \rightarrow aA|bB|eC|BC|d|E$$
$$B \rightarrow A|aA|B|\varepsilon$$
$$C \rightarrow aC|d|\varepsilon$$
$$E \rightarrow f|g$$

How many non-terminals in "Nullable set" of G? (Nullable set is the set of non-terminals which can derive empty string)

(a) 1 (b) 2
(c) 3 (d) 4

Q.11 Pushdown automata accepts

(a) Regular language
(b) Context-free language
(c) Context-sensitive language
(d) Both (a) and (b)

Q.12 What is addition in pushdown automata as compare to finite state automata?

(a) Stack memory (b) Input tape
(c) Head (d) None of these

Q.13 A pushdown automata behaves like an finite state automata when the number of auxiliary memory it has, is

(a) One
(b) Zero
(c) Two
(d) Can't behave like that

Q.14 Pushdown automata behaves like Turing Machine when it has the number of auxiliary memory

(a) One or more
(b) Two or more
(c) Zero
(d) Can't behave like Turing machine

Q.15 Choose the correct statement:

(a) The power of non-deterministic pushdown automata and deterministic pushdown is same.
(b) The power of non-deterministic pushdown automata and deterministic pushdown automata is not same.
(c) A non-deterministic pushdown automata can be converted into an equivalent DPDA.
(d) None of these

Q.16 Choose the correct statement, where FSM stands for finite state machine and PDA stands for pushdown automata
(a) A FSM with one stack is more powerful than FSM without stack
(b) A FSM with one stack is more powerful than PDA
(c) A FSM with two stacks is equal powerful as PDA
(d) All statements are correct

Q.17 Choose the correct statement, where FSM, PDA, TM stand for finite state machine, pushdown automata and Turing machine respectively.
(a) A FSM with two stacks is more powerful than TM
(b) A FSM with one stack is more powerful than PDA
(c) A FSM with two stacks is equal powerful as TM
(d) All statement are correct

Q.18 If L_1 is CFL and L_2 is regular language. Which one of the following is false?
(a) $L_1 - L_2$ is CFL (b) $L_1 \cap L_2$ CFL
(c) $\overline{L_1}$ is CFL (d) $\overline{L_2}$ is Regular

Q.19 Which one of the following is the most powerful machine?
(a) FSM with 1 stack (b) FSM with 2 stacks
(c) FSM with 3 stacks (d) Both (b) and (c)

Q.20 Let L_D be the set of all languages accepted by a PDA by final state and L_E be the set of all languages accepted by empty stack. Which of the following is true?
(a) $L_D = L_E$ (b) $L_D \supset L_E$
(c) $L_D \subset L_E$ (d) None of these

Q.21 Which of the following language is accepted by at DPDA?
(a) $\{wcw^R : w \in (a+b)^*\}$
(b) $\{ww^R : w \in (a+b+c)^*\}$
(c) $\{a^n b^n c^n : n \geq 0\}$
(d) $\{w : w$ is a palindrome over $\{a, b, c\}\}$

Q.22 A DPDA accepts any language which is
(a) DCFL (b) CFL
(c) Regular language (d) Both (a) and (c)

Q.23 An NPDA accepts any language which is
(a) DCFL (b) CFL
(c) Regular language (d) All of these

Q.24 Which one of the following languages is not deterministic context-free?
(a) $L = \{a^n b^n : n \geq 0\}$
(b) $L = \{a^n b^n : n \geq 0\} \cup \{a^n b^{2n} : n \geq 0\}$
(c) $L = \{a^n c b^n : n \geq 0\}$
(d) None of these

Q.25 Which one of the following languages is deterministic context-free?
(a) $L = \{wxw^R : w \in (a+b)^*, x \in (a+b)^+\}$
(b) $L = \{ww^R : w \in (a+b)^*\}$
(c) $L = (X : X$ is a palindrome over $\{a, b\}\}$
(d) None of these

Q.26 Choose one of the following languages that is not recognized by deterministic pushdown automata
(a) $L = \{wcw^R : w \in (a+b)^*\}$
(b) $L = \{a^n b^m : m \neq n\}$
(c) $L = \{a^n c b^n : n \geq 0\} \cup \{a^m d c^{2m} : m \geq 0\}$
(d) None of these

Q.27 Which language does the following PDA accept $M = \{\{q_0, q_1\}, \{0, 1\}, \{z_0, x\}, \delta, q_0, z_0, \phi\}$ and δ is given by

$\delta(q_0, 0, z_0) = (q_0, xz_0)$
$\delta(q_0, 0, x) = (q_0, xx)$
$\delta(q_0, 1, x) = (q_1, \varepsilon)$
$\delta(q_1, 1, x) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, z_0) = (q_0, \varepsilon)$

(a) $L = \{0^n 1^n \mid n \geq 0\}$
(b) $L = \{0^n 1^n \mid n \geq 1\}$
(c) $L = \{0^n 1^{n+1} \mid n \geq 0\}$
(d) $L = \{0^n 1^{n+1} \mid n \geq 1\}$

Q.28 Which language does M accept if the following transition is added to transitions of Q.27?

$\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$

(a) $L = \{0^n 1^n \mid n \geq 0\}$
(b) $L = \{0^n 1^n \mid n \geq 1\}$
(c) $L = \{0^n 1^{n+1} \mid n \geq 0\}$
(d) $L = \{0^n 1^{n+1} \mid n \geq 1\}$

Q.29 The CFG: $S \to aS|bS|a|b$ is equivalent to which of the following regular expression

1. $(a^* + b)^*$
2. $(a + b)^+$
3. $(a + b)(a + b)^*$
4. $(a + b)^*(a + b)$

(a) 2 and 3 only (b) 2, 3 and 4
(c) 1 only (d) 3 and 4 only

Q.30 The following CFG

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

generates strings of terminals that have
(a) equal number of a's and b's
(b) odd number of a's and even number of b's
(c) even number of a's and even number of b's
(d) odd number of a's and even number of a's

Answer Key:

1. (d)	**2.** (d)	**3.** (c)	**4.** (b)	**5.** (a)
6. (c)	**7.** (d)	**8.** (c)	**9.** (c)	**10.** (d)
11. (d)	**12.** (a)	**13.** (b)	**14.** (b)	**15.** (b)
16. (a)	**17.** (c)	**18.** (c)	**19.** (d)	**20.** (a)
21. (a)	**22.** (d)	**23.** (d)	**24.** (b)	**25.** (a)
26. (d)	**27.** (b)	**28.** (a)	**29.** (b)	**30.** (a)

Student's Assignments 1 Explanations

1. (c)

$S \to aA|bSS|SS$
$A \to aAb|bAa|AA|\varepsilon$

'A' generates equal number of a's and b's
'S' generates atleast one more a than A generates
∴ Grammar G generates all strings with atleast one more 'a' than number of b's.

2. (d)

All given languages are DCFL.

(a) $\{w \mid \#_0(w)! = \#_1(w), w \in (0+1)^*\}$ is DCFL

(b) $\{xwx \mid x \in (0+1), w \in (0+1)^*, \#_0(w) = \#_1(w)\}$ is DCFL

(c) If string starts with 1 then it accepts 0^n1^n as next symbols of the string. If string starts with 11 then it accepts 0^K1^{2K} as next symbols of the string, which is also DCFL.

3. (c)

$$\left.\begin{array}{l} S \to AAaSb|\varepsilon \\ A \to a|\varepsilon \end{array}\right\} \equiv S \to aSb|aaSb|aaaSb|\varepsilon$$

$$L(G) = \{a^m b^n \mid n \le m \le 3n\}$$

4. (b)

$S \to aAb$
$\Rightarrow aBdb$
$\Rightarrow aabedb$
$|aabedb|$ = Maximum length = 6

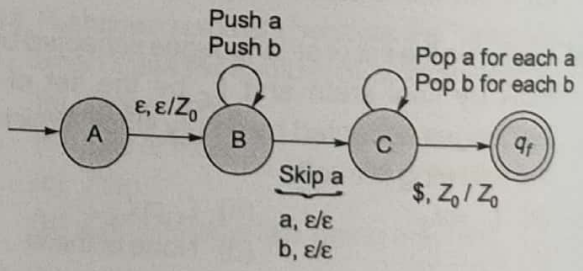
5. (a)

Language L generated by CFG is:

$$L = \{a^m b^n a^k \mid m+n=k\}$$

Let $w \in L$.
$|w| = 0 \Rightarrow w = \varepsilon \Rightarrow$ # strings = 1
$|w| = 2 \Rightarrow w =$ ba,aa $\Rightarrow$ #strings = 2
$|w| = 4 \Rightarrow w =$ abaa, bbaa, aaaa $\Rightarrow$ #strings = 3
⋮
$|w| = 8 \Rightarrow$ # strings = 5
$|w| < 10 \Rightarrow$ # strings = 1 + 2 + 3 + 4 + 5 = 15

6. (c)



$$L = \left\{ \underbrace{w}_{\text{push}} \underbrace{x}_{\text{skip}} \underbrace{w^R}_{\text{pop}} \mid w \in (a+b)^*\ x \in (a+b) \right\}$$

∴ Option (c) is correct.

7. (d)

The language accepted by the PDA with finite Stack is always regular language. Regular language may be finite or infinite.
∴ Option (d) is correct.

8. (c)
$L_1.L_2$ = (Regular) . (CSL)
L_2 is $a^n b^n c^n$, but L_1 can be any regular language
Case 1: If $L_1 = \phi$,
$\Rightarrow L_1.L_2 = \phi.\{a^n b^n c^n\} = \phi$ is regular
Case 2: If $L_1 = \{\varepsilon\}$
$\Rightarrow L_1.L_2 = \{\varepsilon\}.\{a^n b^n c^n\} = \{a^n b^n c^n\}$ is CSL
$L_1.L_2$ is always CSL but it may or may not be regular.

9. (c)
$S \rightarrow aA \mid bB \mid AB \mid \varepsilon$
$A \rightarrow aAb \mid \varepsilon$
$B \rightarrow bBa \mid \varepsilon$
L(G) = {ε, a, b, ab, ba, aab, bba, aabb, bbaa, abba, . . .}
L_1 = {ε,a, b, ab, ba, aab, bba}
strings in L_1 = 7

10. (d)
{S, A, B, C} is Nullable set
∴ 4 non-terminals are in nullable set.

11. (d)
PDA accepts context free languages. All regular languages accepted by PDA
∴ Option (d) is correct.

12. (a)
FA with stack is equivalent to PDA
∴ Option (a) is correct.

13. (b)
If PDA has no stack then it behaves like FA.
∴ Option (b) is correct.

14. (b)
PDA already has one and additionally atleast one stack is needed to act like turing machine. Number of auxiliary memory in PDA is atleast two needed.
∴ Option (b) is correct.

15. (b)
DPDA is less powerful than NPDA
DPDA accepts only DCFLs, but NPDA accepts subset of DCFLs called NCFLs or CFLS.
∴ Option (b) is correct.

16. (a)
FSM with one stack is more powerful than FSM and equivalent to PDA.

17. (c)
FSM with two stacks ≅ TM
∴ Option (c) is correct.

18. (c)
L_1 is CFL then $\overline{L_1}$ need not be a CFL.
$L_1 = \{\overline{ww} \mid w \in (a+b)^*\}$ is a CFL
But $\{ww \mid w \in (a+b)^*\}$ is not a CFL
∴ Option (c) is FALSE.

19. (d)
FSM with 2 stacks ≅ FSM with 3 stacks ≅ TM
∴ Option (d) is correct.

20. (a)
$L_D = L_E$
PDA acceptance by final state ≅ PDA acceptance by empty stack.
∴ Option (a) is correct.

21. (a)
$\{w\,C\,w^R \mid w \in (a+b)^*\}$ is DCFL.
Push all symbols before symbol C and pop matched symbols after C. DPDA construction is possible with deterministic transitions.
∴ Option (a) is correct.

22. (d)
DPDA accepts DCFL
Every regular language is DCFL.
∴ Option (d) is correct.

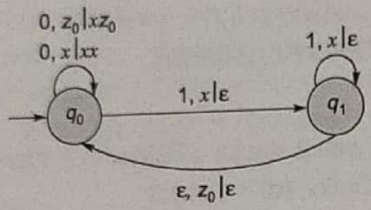
23. (d)
NPDA accepts CFL, every regular is CFL and every DCFL is also CFL.
∴ Option (d) is correct.

24. (b)
$\{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$ is not DCFL.
Because no DPDA exist to accept above language.

25. (a)
$L = \{wx\,w^R \mid w \in (a+b)^*, x \in (a+b)^+\}$ is regular.
$\Rightarrow \quad L = (a+b)^+$
Hence L is also DCFL.

26. (b)
$L = \{wcw^R \mid w \in (a+b)^*\}$ is a DCFL
$L = \{a^n b^m \mid m \neq n\}$ is a DCFL
$L = \{a^n cb^n \mid n \geq 0\} \cup \{a^m dc^{2m} \mid m \geq 0\}$ is a DCFL

27. (b)



Language accepted by above PDA is $\{a^n b^n \mid n \geq 1\}$

28. (a)
If $\delta\{q_0, \varepsilon, Z_0\} = \{q_0, \varepsilon\}$ is added to initial state of PDA then $L = \{a^n b^n \mid n \geq 0\}$.
Only any input initial state can empty the stack without reading any input symbol. Hence it accepts empty string.
$\therefore$ Option (a) is correct.

29. (b)
$S \rightarrow aS \mid bS \mid a \mid b$
$L(S) = (a+b)^+$
1. $(a^* + b)^* = (a+b)^* \neq L(S)$
2. $(a+b)^+ = L(S)$
3. $(a+b).(a+b)^* = (a+b)^+ = L(S)$
4. $(a+b)^*(a+b) = (a+b)^+ = L(S)$

30. (a)
$S \rightarrow aB \mid bA$
$A \rightarrow a \mid aS \mid bAA$
$B \rightarrow b \mid bS \mid aBB$
S generates all strings having equal number of a's and b's.

Student's Assignments 2

Q.1 What is the context free grammar for odd length palindromes?
(a) $S \rightarrow aSb \mid bSa \mid \varepsilon$ (b) $S \rightarrow aSa \mid bSb \mid a \mid b$
(c) $S \rightarrow aSb \mid bSa \mid a \mid b$ (d) $S \rightarrow aAb \mid bAa \mid \varepsilon$

Q.2 Context free languages are not closed under ___.
(a) union (b) init operation
(c) complementation (d) kleene closure

Q.3 Which of the following CFG is in CNF
(a) $S \rightarrow AB, B \rightarrow b \mid \varepsilon, A \rightarrow \varepsilon$
(b) $S \rightarrow AB, A \rightarrow a, B \rightarrow b$
(c) $S \rightarrow ABC, A \rightarrow a, B \rightarrow b, C \rightarrow c$
(d) $S \rightarrow AB \rightarrow \varepsilon, B \rightarrow b, A \rightarrow a$

Q.4 Which of the following is context free language?
(a) $L = \{a^i b^j c^{i+j} \mid i, j > 0\}$
(b) $L = \{a^p \mid p \text{ is prime}\}$
(c) $L = \{ww \mid w \in (a+b)^*\}$
(d) $L = \{ww^R w \mid w \in (0+1)^*\}$

Q.5 If L_1 and L_2 are context free languages, choose the correct statement from the following
(a) $L_1 \subseteq L_2$ is decidable
(b) Intersection of L_1 and L_2 is context free language.
(c) whether $L_1 \cap L_2 = \phi$ is decidable
(d) whether L_1 is finite is decidable

Q.6 What is the language accepted by following transitions of PDA which uses acceptance by emtpy stack
$\delta(q_0, a, S) = (q_0, SA)$
$\delta(q_0, b, S) = (q_0, SB)$
$\delta(q_0, a, S) = (q_0, \varepsilon)$
$\delta(q_0, b, S) = (q_0, \varepsilon)$
$\delta(q_0, \varepsilon, S) = (q_0, \varepsilon)$
$\delta(q_0, a, A) = (q_0, \varepsilon)$
$\delta(q_0, b, B) = (q_0, \varepsilon)$
(a) set of all palindromes
(b) set of all odd length palindromes
(c) set of all even length palindromes
(d) set of equal number of a's and b's

Q.7 Given, two languages L_1 and L_2, the problem of checking whether $L_1 \subseteq L_2$ is decidable
(a) only when both are CFLs
(b) only when both are regular
(c) when atleast one of them is regular
(d) None of these

Q.8 Language generated by the following CFG is
$S \rightarrow aB|bA|\varepsilon$
$A \rightarrow a|aS|bAA$
$B \rightarrow b|bS|aBB$
(a) Equal number of a's and b's
(b) Odd number of a's and even number of b's
(c) Even number of a's and b's
(d) None of the above

Q.9 Which of the following is context free language
(a) $\{a^i b^j c^k | i < j < k\}$
(b) $\{a^n b^m c^{n+m} d^n | n, m > 0\}$
(c) $\{w | w$ contain number of a's = number of b's + 1$\}$
(d) $\{a^i b^j | j = i^2\}$

Q.10 Which of the following is a DCFL
(a) $\{ww^R | w \in (a+b)^*\}$
(b) $\{ww | w \in (0+1)^*\}$
(c) $\{a^n b^{2n} | n > 0\}$
(d) $\{a^n b^{2n} | n > 0\}\{a^n b^n | n > 0\}$

Q.11 The order of constructing simplified grammar
(a) eliminate null productions, unit productions, useless symbols
(b) eliminate useless symbols, null productions, unit productions
(c) eliminate useless symbols , unit productions and null productions
(d) any order can be followed

Q.12 CYK algorithm is used to check the following
(a) Finiteness of CFL's
(b) Membership of CFL
(c) Equivalence of CFL's
(d) None of the above

Q.13 Choose correct statement from the following
(a) GNF do not contain null productions
(b) GNF may contain null productions
(c) CNF may contain null productions
(d) CNF and GNF may contain unit productions

Q.14 Which of the following statements is correct
(a) DPDA is more powerful than NPDA
(b) DPDA is less powerful than NPDA
(c) DPDA is equally powerful to NPDA
(d) we can't say

Q.15 CNF may contain the following
(a) null productions (b) unit productions
(c) useless symbols (d) All of these

Q.16 Union of two deterministic context free languages is
(a) never DCFL
(b) need not be DCFL
(c) not a context free language
(d) none of the above

Q.17 Which of the following languages is context free language
$L_1 : \{a^n b^{5n} | n > 0\}$
$L_2 : \{a^n b^{2n} c^{3n} | n > 0\}$
$L_3 : \{a^n b^n c^m | n < m \leq 2n\}$
$L_4 : \{a^i b^j c^k | i < j < k\}$
(a) L_1 and L_2 (b) L_1 only
(c) L_2 and L_4 (d) L_1, L_2, L_3 and L_4

Q.18 Context free language is
(a) finite automata with one stack
(b) finite automata with one stack and R/W head
(c) finite automata with a tape of two sides moving capability
(d) None of the above

Q.19 $L = \{\overline{ww} | w \in (a+b)^*\}$ is
(a) regular language
(b) context free language
(c) not a context free language
(d) both context free and regular

Q.20 GNF context free grammar equivalent
(a) ϵ-free CFLs (b) CNF-CFG
(c) Both (a) and (b) (d) None of these

Q.21 Find the rank of A in given context free grammar
$A \to BC$
$B \to a$
$C \to b$
(a) 2 (b) 1
(c) 0 (d) 3

Q.22 Push down automata can accept the language by
(a) Empty stack (b) Final state
(c) both (a) or (b) (d) None of these

Q.23 Deterministic context free languages are not closed under following
(a) intersection with regular sets
(b) inverse homomorphism
(c) homomorphism
(d) complementation

Common Data for (Q.24 and Q.25)
Consider grammar G:
$S \to AB, A \to aAA|\varepsilon, B \to bBB|\varepsilon.$

Q.24 Find the nullable symbols in the given grammar
(a) A (b) B
(c) A, B and S (d) A and B

Q.25 If G_1 is constructed from G, after eliminating ϵ-productions, then G_1 is given by
(a) $S \to AB$
$A \to aAA|aA$
$B \to bBB|bB$
(b) $S \to AB|A|B$
$A \to aAA|aA$
$B \to bBB|bB$
(c) $S \to AB$
$A \to aAA|aA|a$
$B \to bBB|bB|b$
(d) $S \to AB|A|B$
$A \to aAA|aA|a$
$B \to bBB|bB|b$

Q.26 The grammar $S \to aaSbb|ab$ can generate the set
(a) $\{a^n b^n | n = 1, 2, 3, ...\}$
(b) $\{a^{2n+1} b^{2n+1} | n = 0, 1, 2, ...\}$
(c) $\{a^{2n+1} b^{2n+1} | n = 1, 2, 3, ...\}$
(d) None of the above

Q.27 Which of the following CFG's can't be simulated by an FSM?
(a) $S \to aSa|a$
(b) $S \to abX, X \to cY, Y \to d|aX$
(c) $S \to aSb|ab$
(d) None of these

Q.28 Consider the grammar $S \to PQ|SQ|PS, P \to x, Q \to y$. To get string of n terminals, the number of productions to be used is
(a) n^2 (b) $2n$
(c) 2^{n+1} (d) $2n-1$

Q.29 Which of the following languages can't be accepted by a deterministic PDA?
(a) The set of palindromes over alphabet $\{a, b\}$
(b) The set of all strings of balanced parenthesis
(c) $L = \{WCW^R | W \text{ in } (0+1)^*\}$
(d) $L = \{0^n 1^n | n \geq 0\}$

Q.30 Which language does the following PDA accept?
$M = (\{q_0, q_1\}, \{0, 1\}, \{z_0, x\}, \delta, q_0, z_0, \phi)$ and δ is given by
$\delta(q_0, 0, z_0) = (q_0, xz_0)$
$\delta(q_0, 0, x) = (q_0, xx)$
$\delta(q_0, 1, x) = (q_1, x)$
$\delta(q_1, 1, x) = (q_1, \varepsilon)$ and $\delta(q_1, \varepsilon, z_0) = (q_0, \varepsilon)$
(a) $L = \{0^n 1^n | n \geq 0\}$
(b) $L = \{0^n 1^n | n \geq 1\}$
(c) $L = \{0^n 1^{n+1} | n \geq 0\}$
(d) $L = \{0^n 1^{n+1} | n \geq 1\}$

Answer Key:

1. (b)	**2.** (c)	**3.** (b)	**4.** (a)	**5.** (d)
6. (a)	**7.** (b)	**8.** (a)	**9.** (c)	**10.** (c)
11. (a)	**12.** (b)	**13.** (a)	**14.** (b)	**15.** (c)
16. (b)	**17.** (b)	**18.** (a)	**19.** (b)	**20.** (c)
21. (b)	**22.** (c)	**23.** (c)	**24.** (c)	**25.** (d)
26. (b)	**27.** (c)	**28.** (d)	**29.** (a)	**30.** (d)

CHAPTER 04

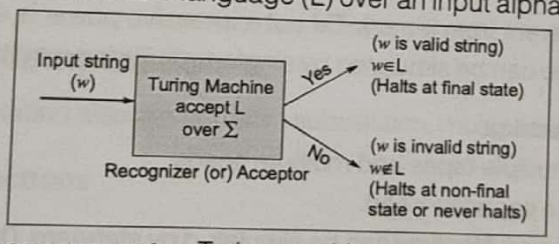
REC, RE Languages & Turing Machines, Decidability

4.1 Turing Machine (TM)

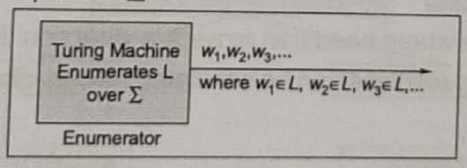
Turing Machine recognizes the recursive enumerable language. Turing Machine is more powerful than any other automata such as finite automata, PDA and LBA. TM accepts recursive enumerable language by using universal acceptance mechanism Turing Machine enumerates the recursive enumerable language.

Turing Machine computes the partial recursive function. Turing Machine can be modeled as Deterministic Turing Machine (DTM) or Non-deterministic Turing Machine (NTM). Bydefault Turing machine is DTM. Power of DTM and NTM are same.

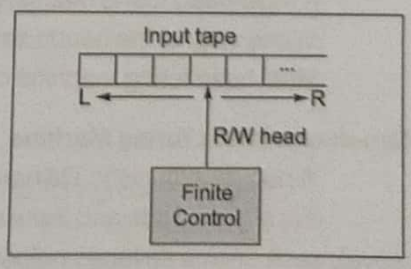
1. **Turing Machine Acts as Recognizer or Acceptor:** Turing machine that accepts or recognizes the strings of a recursive enumerable language (L) over an input alphabet Σ.



2. **Turing Machine Acts as enumerator:** Turing machine enumerates the string of recursive enumerable language over the input alphabet Σ.



4.1.1 Configuration of Turing Machine



- It contains a finite control and unbounded input tape.
- Finite control contain finite number of states.
- R/W head reads the input symbol from the input tape in either direction (left or right).
- For every string finite control starts from the initial state. If the string is valid finite control reaches to the final state.
- R/W head can move in any direction left or right, this capability is called "turn around capability".

4.1.2 Specification of Turing Machine

The Turing machine is represented as a seven tuple as follows:

$$TM = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Where Q = set of finite states
Σ = input alphabet ($\Sigma \subseteq \Gamma$)
Γ = tape alphabet
δ = transition function;
DTM $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
NTM $\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$
B = blank symbol ($B \in \Gamma$)
F = final states ($F \subseteq Q$)

4.2 Variation of Turing Machines

Multiple Tracks Turing Machine

A *k-track* Turing machine (for some fixed $k > 0$) has k-tracks and one R/W head that reads and writes all of them one by one.

A *k-track* Turing machine can be simulated by a *single track TM.*

Two-way Infinite Tape Turing Machine

Infinite tape of *two-way infinite tape TM* is unbounded in both directions left and right.

Two-way infinite tape TM can be simulated by *one-way infinite tape TM* (standard Turing machine).

Multi-Tape Turing Machine

It has multiple tapes and controlled by a single head.

The Multi-tape TM is different from k-track TM but expressive power is same.

Multi-tape Turing machine can be simulated by single-tape Turing machine.

Multi-Tape Multi-Head Turing Machine

The Multi-tape TM has multiple tapes and multiple heads.

Each tape controlled by a separate head.

Multi-Tape Multi-Head Turing Machine can be simulated by standard TM.

Multi-dimensional Tape Turing Machine

It has multi-dimensional tape where head can move any direction that is left, right, up or down.

Multi-dimensional tape Turing machine can be simulated by one-dimensional Turing machine.

Multi-head Turing Machine

A multi-head turing machine contain two or more heads to read the symbols on the same tape.

In one step all the heads sense the scanned symbols and move or write independently.

Multi-head turing machine can be simulated by single head turing machine.

Non-deterministic Turing Machine

A non-deterministic TM has a single, one way infinite tape.

For a given state and input symbol has at least one choice to move (finite number of choices for the next move), each choice several choices of path that it might follow for a given input string.

A non-deterministic TM is equivalent to a deterministic TM.

Off-line Turing Machine

This TM is similar to multi-tape TM, but its input is read-only and has two end markers for input on the input tape. The remaining tapes are two-way infinite tapes.

This TM does not allow its head to move across the marked region on the input tape.

Off-line TM is equivalent to a one-way infinite TM.

Universal Turing Machine

It is a programmable turing machine and it is designed with three tapes. One tape contain input string, second tape contain encoding of TM and third tape contain current state of TM.

Universal turing machine can simulate any other turing machine.

Multi-stack Turing Machine

It contains multi-stacks apart from input tape for the simulation.

Multi-stack turing machine can be simulated by standard turing machine.

Counter Machine

A counter machine can store finite number of integers. Each number store on separate counter.

It is a machine with counters where the counters can be either increased or decreased.

Counter is a stack with an alphabet of exactly two symbols, a stack start symbol and a counter symbol.

4-counter machine can simulate any other turing machine.

Two-Stack PDA

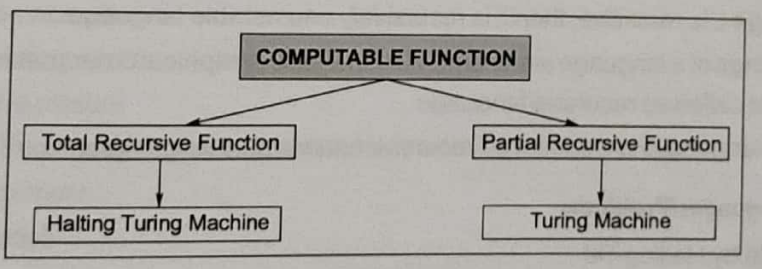
It is a machine with multi-stacks where the stacks can be used to push or pop symbols like PDA.

Two stack PDA can simulate standard turing machine.

4.3 Turing Machine Computation

TM can perform computation also like addition, subtraction, multiplication, division, etc.

4.3.1 Total and Partial Functions



4.3.2 Effective Computation

Allen Turing Computation

"A number-theoretic function f is effectively calculable if and only if f is Turing computable".

- If f is effectively calculable, then f is Turing-computable.
- If f is Turing-computable, then f is effectively calculable.

"If function *f* is effectively calculable, then *f* is Turing-computable; Equivalently, if a function *f* is not Turing computable, then *f* is not effectively calculable."

Church-Turing Computation

Any algorithm can be represented as a turing machine. So every algorithm can be realized as a turing machine. Hence any algorithm which is implemented using C++ or Java programming languages can also be implemented using the Turing Machine.

4.4 Turing Machine Recognizable Languages

4.4.1 Recursively Enumerable Language (REL)

- The class of languages recognized by the TM is known as *recursively enumerable languages.*
- Language *L* is a recursively enumerable set if and only if, there exists a TM *M* halts for all $w \in L$.
- All recursively enumerable languages are TM enumerable.
- The language enumerated by TM is called as *recursively enumerable language.*
- Not all languages are recursively enumerable.
- A language generated by the unrestricted grammar is called as *recursively enumerable language.*

Class of Recursively Enumerable Languages/Problems

- Acceptable / Recognizable by TM
- Enumerable by TM
- Partially recursive
- Partially decidable / semi-decidable
- Computable / Turing computable
- Partially algorithmically solvable

4.4.2 Recursive Language

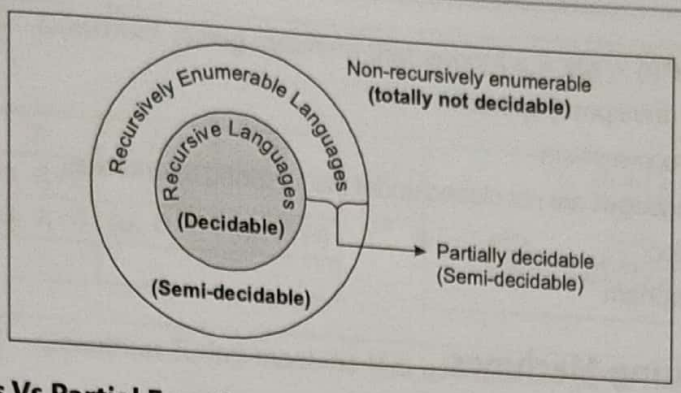
- Language *L* is a recursive set if and only if, for this some TM M halts for all $w \in L$ and rejects $w \notin L$.
- If a language *L* is recursive, then it is recursively enumerable language.
- If all the strings of a language are enumerated in a lexicographical order (particular) by TM then such language is called as recursive language.
- Recursive languages are subset of Recursive enumerable languages.

Class of Recursive Languages/Problems

- Acceptable by Halting TM
- Decidable / Turing decidable
- Enumerable by TM in lexicographical order
- Algorithmically solvable

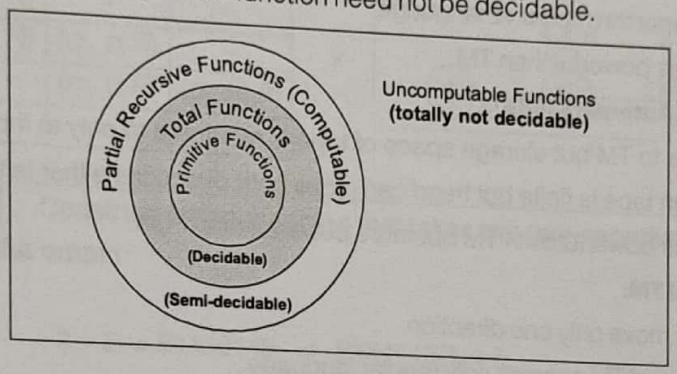
4.4.3 Recursive Languages Vs Recursively Enumerable Languages

Recursive languages are decidable but RELs are partially decidable. Non-RELs are totally not decidable. RELs and non-RELs are undecidable languages.



4.4.4 Total Functions Vs Partial Functions

Total recursive functions is decidable but partial recursive function is computable. Every decidable function is computable but every computable function need not be decidable.



4.5 Closure Properties

4.5.1 Closure Properties of Recursive Enumerable Languages

- Recursive Enumerable Languages are closed under the following operations:
  - (*i*) Union operation
  - (*ii*) Intersection operation
  - (*iii*) Concatenation operation
  - (*iv*) Kleene closure
  - (*v*) Reversal operation
  - (*vi*) Positive closure
- Recursive Enumerable Languages are not closed under the following operations:
  - (*i*) Complement
  - (*ii*) Difference

4.5.2 Closure Properties of Recursive Languages

- Recursive Languages are closed under the following operations:
  - (*i*) Union operation
  - (*ii*) Intersection operation
  - (*iii*) Complement
  - (*iv*) Concatenation operation
  - (*v*) Kleene closure, Positive closure

(vi) Difference

(vii) Reversal (transpose) operation

(viii) ε-free homomorphism

- Recursive Languages are not closed under the following operations:
  - (i) Substitution
  - (ii) Homomorphism

4.6 Restricted Turing Machines

1. **Halting Turing Machine (HTM):** If turing machine always halts for every input string then such turing machine is called HTM.
   - HTM accepts the recursive language
   - HTM is less powerful than TM.
2. **Linear Bound Automata (LBA):**
   - It is similar to TM but storage space of tape is restricted to only to input length.
   - Memory on tape is finite but head can move both directions either left or right.
   - LBA is less powerful than TM but more powerful than PDA.
3. **Unidirectional TM:**
   - Head can move only one direction.
   - Unidirectional TM accepts only regular language.
   - Unidirectional TM has same power as finite automata but less powerful than PDA.
4. **Read only TM:**
   - Read only TM is equivalent to finite automata.
   - It contain read head without writing capability.
   - It accepts only regular language.
5. **Read only-Unidirectional TM:**
   - It is equivalent to FA.
   - It contain read only head and it can move only in one direction.
   - It accepts a regular language.

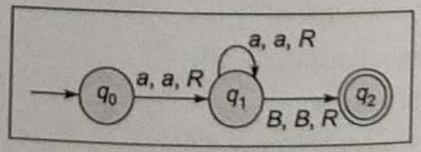
4.7 Turing Machine Construction

Example-4.1 Construct Turing machine that accepts $L = \{a^n | n \geq 1\}$

Solution:

	a	B
$\rightarrow q_0$	(q_1, a, R)	–
q_1	(q_1, a, R)	(q_2, B, R)
$^*q_2$	–	–

or

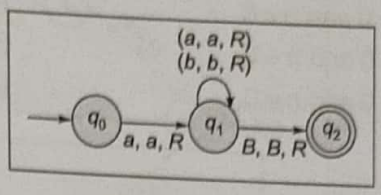


Example - 4.2 Construct Turing machine that accept $L = a(a + b)^*$

Solution:

	a	b	B
$\rightarrow q_0$	(q_1, a, R)	–	–
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, R)
$^*q_2$	–	–	–

or

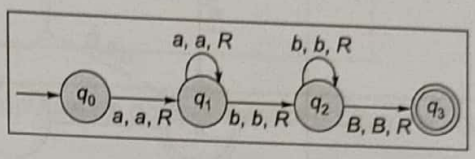


Example - 4.3 Construct Turing machine that accepts $L = \{a^m b^n | m, n \geq 1\}$

Solution:

	a	b	B
$\rightarrow q_0$	(q_1, a, R)	–	–
q_1	(q_1, a, R)	(q_2, b, R)	–
q_2	–	(q_2, b, R)	(q_3, B, R)
$^*q_3$	–	–	–

or



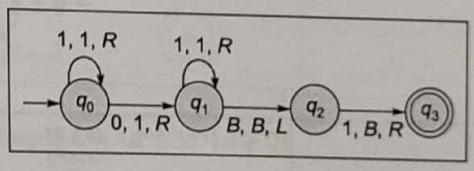
Example - 4.4 Construct turing machine that takes two non-negative integers as input and produces their sum as the output

Solution:

$$3 + 2 \Rightarrow \text{B111011B} \Rightarrow \text{B11111BB} \Rightarrow 5$$

	1	0	B
$\rightarrow q_0$	$(q_0, 1, R)$	$(q_1, 1, R)$	–
q_1	$(q_1, 1, R)$	–	(q_2, B, L)
q_2	(q_3, B, R)	–	–
$^*q_3$	–	–	–

or

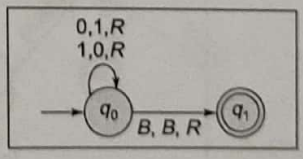


Example - 4.5 Construct turing machine that takes a binary number as input and produces its 1's complement as the output

Solution:

	1	0	B
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, R)
$^*q_1$	–	–	–

or



Example - 4.6 Construct a TM, which computes the addition of two positive integers $(m + n)$ where input string is $\#0^m 10^n\#$ format.

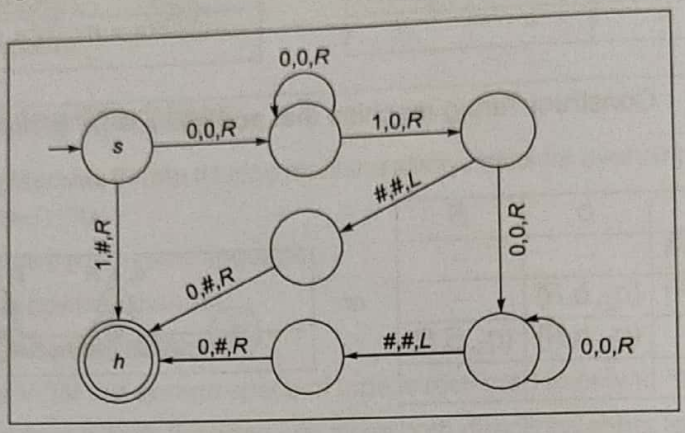
Solution:

Let TM $M = (Q, \{0, 1, \#\}, \delta, s, h)$ computes the addition of two positive integers m and n.

$$f(m, n) = \begin{cases} m + n & (\text{If } m, n \geq 0) \\ 0 & (\text{Otherwise}) \end{cases}$$

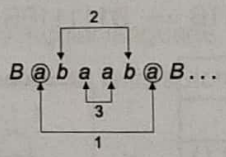
1 on the tape separates both the numbers m and n. Following values are possible for m and n:

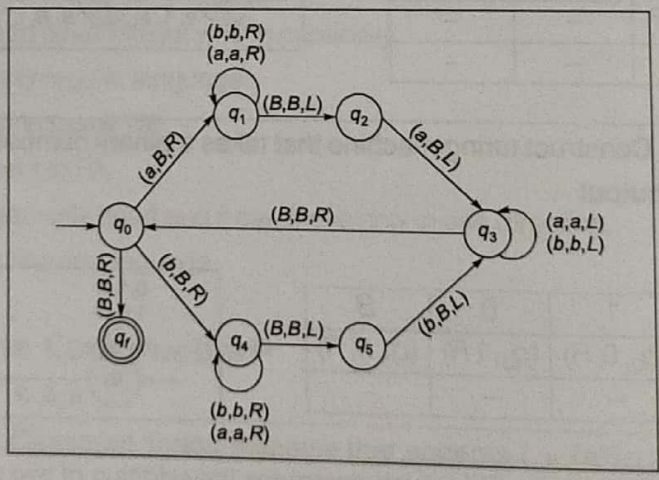
- $m = n = 0$ (#1# is the input)
- $m = 0$ and $n \neq 0$ (#10^n# is the input)
- $m \neq 0$ and $n = 0$ (#0^{m}1# is the input)
- $m \neq 0$ and $n \neq 0$ (#0^{m}10^n# is the input)



Example - 4.7 Construct a TM, which accepts $L = \{ww^R | w \in (a + b)^*, w^R \text{ is the reverse of } w\}$

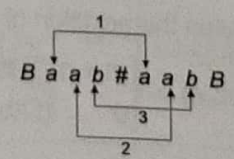
Solution:

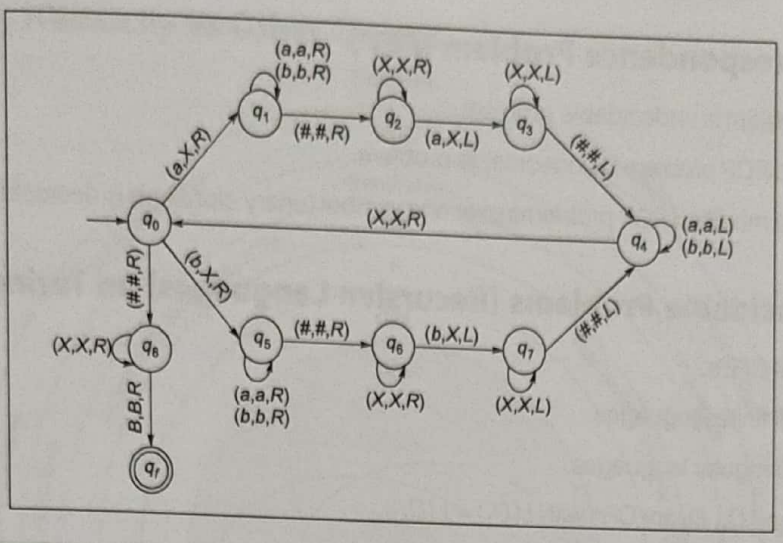




Example - 4.8 Construct a TM, which accepts $L = \{w \# w | w \in (a + b)^*\}$

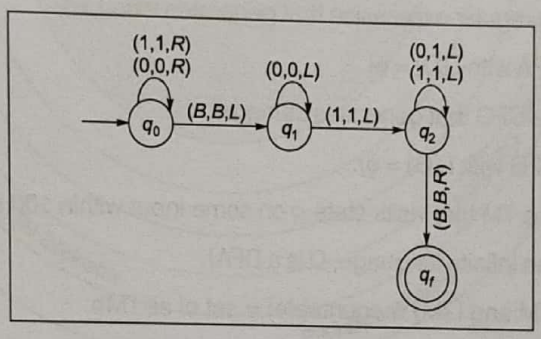
Solution:





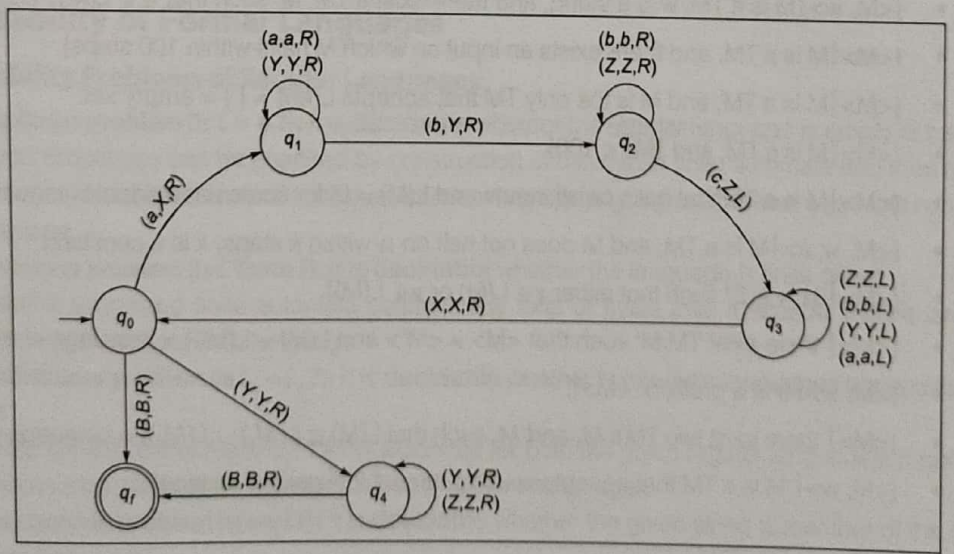
Example - 4.9 Construct a TM, which computes 2's complement of a binary number.

Solution:



Example - 4.10 Construct a TM, which accepts $L = \{a^n b^n c^n \mid n \geq 0\}$.

Solution:



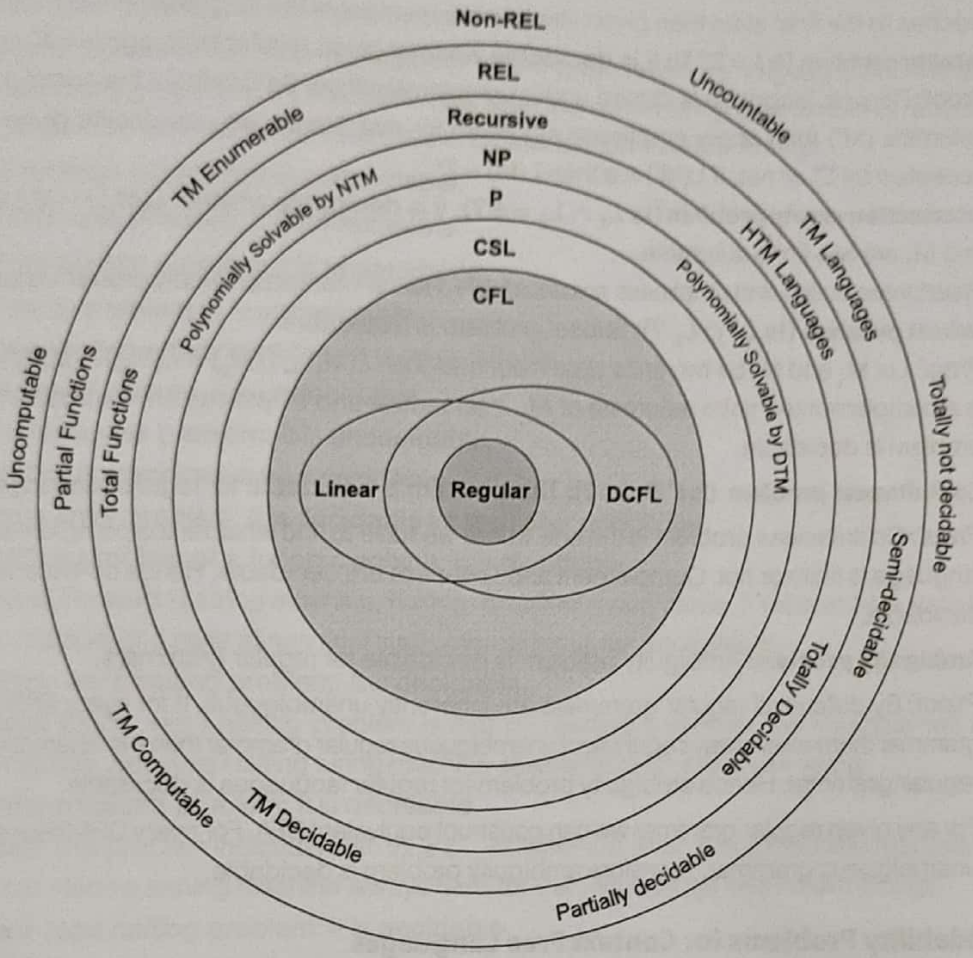
4.8 Post Correspondence Problem (PCP)

- PCP problem is undecidable problem.
- Modified PCP problem is undecidable problem.
- PCP and modified PCP problems over one symbol (unary alphabet) is decidable problem.

4.9 Some Decidable Problems (Recursive Languages) on Turing Machines

- Set of all CFL's.
- Set of all finite languages.
- Set of all regular languages.
- $\{<D_1, D_2> | D_1, D_2$ are DFA with $L(D_1) = L(D_2)\}$.
- $\{<D, w> | D$ is a DFA that accepts string $w\}$.
- $\{<N, w> | N$ is a NFA that accepts string $w\}$.
- $\{<R, w> | R$ is a regular expression that generates string $w\}$.
- $\{<D> | D$ is a DFA with $L(D) = \phi\}$.
- $\{<G, w> | G$ is a CFG that generates string $w\}$.
- $\{<G> | G$ is a CFG with $L(G) = \phi\}$.
- $\{<M, q> | M$ is a TM that visits state q on some input within 100 steps$\}$.
- $\{<D> | L(D)$ is an infinite language, D is a DFA$\}$.
- $\{<M> | M$ is a TM and L(M) is countable$\}$ = set of all TM's.
- $\{<M> | M$ is a TM and L(M) is uncountable$\}$ = $\{\}$.
- $\{<M> | M$ is a TM, M_h is a TM that halts on all inputs and $M \in L(M_h)\}$.
- $\{<M, w> | M$ is a TM, w is a string, and there exist a TM, M′ such that $w \notin L(M) \cap L(M')\}$.
- $\{<M> | M$ is a TM, and there exists an input on which M halts within 100 steps$\}$.
- $\{<M> | M$ is a TM, and M is the only TM that accepts L(M)$\}$ = $\{\}$ = empty set.
- $\{<M> | M$ is a TM, and $|M| < 100\}$.
- $\{<M> | M$ is a TM that halts on all inputs and L(M) = L′ for some undecidable language L′$\}$ = $\{\}$.
- $\{<M, w, k> | M$ is a TM, and M does not halt on w within k steps, k is a constant$\}$.
- $\{<M> | \exists x, y \in \Sigma^*$ such that either $x \in L(M)$ or $y \notin L(M)\}$.
- $\{<M> |$ there exist TM M′ such that $<M> \neq <M'>$ and $L(M) = L(M')\}$ = language of all TM's.
- $\{<M, w> | w$ is a prefix of $<M>\}$.
- $\{<M> |$ there exist two TM's M_1 and M_2 such that $L(M) \subseteq L(M_1) \cup L(M_2)\}$ = language of all TM's.
- $\{<M, w> |$ M is a TM that accepts w using atmost $2^{|w|}$ cells of its tape$\}$.

4.10 Chomsky Hierarchy Vs Other Classes



Chomsky Hierarchy Vs Other Classes

4.11 Decidability of Formal Languages

4.11.1 Decidability Problems of Regular Languages

1. **Emptiness problem (Is $L = \phi$?):** It is **decidable** whether the regular language is empty or not.
   Proof: Emptiness can be checked by construction of minimized finite automata and then check if automata contains a non-final state then it accepts empty language. Otherwise it accepts non-empty language.
2. **Finiteness problem (Is L finite ?):** It is **decidable** whether the language is finite or not.
   Proof: If minimized finite automata contains any loop or cycle then it accepts infinite language otherwise it accepts finite language.
3. **Equivalence problem (Is $L_1 = L_2$?):** It is **decidable** whether two regular languages are equivalent or not.
   Proof: Construct the minimized finite automata for both the given regular languages. If both finite automata are isomorphic then they accept the same language.
4. **Membership problem (Is $w \in L$?):** It is **decidable** whether the given string is member of the regular language or not.

Proof: Construct the minimized finite automata and run the given string on the finite automata. If it reaches to the final state then given string w is a member of the language.

5. **Totality problem (Is $L = \Sigma^*$?):** It is **decidable** whether given regular language $L = \Sigma^*$ or not.
   Proof: Regular language is closed under complementation. So construct the complemented finite automata (M′) then apply emptiness algorithm for that. Hence we can decide given language is accepted by Σ^* or not. If $L(M') = \phi$ then $L(M) = \Sigma^*$.
6. **Intersection empty problem (Is $L_1 \cap L_2 = \phi$?):** It is **decidable** whether $L(M_1) \cap (M_2) = \phi$ where M_1 and M_2 are two finite automata.
   Proof: Intersection and emptiness are decidable. Hence intersection empty problem is also decidable.
7. **Subset problem (Is $L_1 \subseteq L_2$?):** Subset problem is **decidable**.
   Proof: Let M_1 and M_2 be two finite state machines then $L(M_1) \subseteq L(M_2)$ iff $L(M_1)) \cap L(M_2') = \phi$. Here M_2' is a complemented finite automata of M_2. Intersection and emptiness are decidable hence subset problem is decidable.
8. **Co-finiteness problem (Is L' finite ?):** This problem is **decidable** for regular languages.
   Proof: Co finiteness problem is the one where we have to find whether the complement of the given language is finite or not. Complement and finiteness are decidable. Hence co-finiteness problem is decidable.
9. **Ambiguity problem:** Ambiguity problem is **decidable** for regular grammars.
   Proof: By default all regular grammars are inherently unambiguous. If for every ambiguous regular grammar there exist always equivalent unambiguous regular grammar then it is inherently unambiguous regular grammar. Hence ambiguity problem for regular languages is decidable.
   For any given regular grammar we can construct equivalent DFA. For every DFA there exist equivalent unambiguous grammar. Therefore ambiguity problem is decidable.

4.11.2 Decidability Problems for Context Free Languages

1. **Emptiness problem (Is $L = \phi$?):** It is **decidable** whether the context free language is empty or not.
   Proof: Emptiness can be checked by constructing simplified CFG. If simplified CFG has no start symbol then it generates empty language.
2. **Finiteness problem (Is L finite ?):** It is **decidable** whether the CFL is finite or not.
   Proof: Finiteness can be checked by constructing dependency graph over the non-terminals of given grammar. If dependency graph has no cycle and self-loop then it generates finite language.
3. **Equivalence problem (Is $L_1 = L_2$?):** It is **undecidable**.
4. **Membership problem (Is $w \in L$?):** It is **decidable** whether the given string is member of the regular language or not.
   Proof: Membership problem can be solved using CYK parsing.
5. **Totality problem (Is $L = \Sigma^*$?):** It is **undecidable**.
6. **Intersection empty problem (Is $L_1 \cap L_2 = \phi$?):** It is **undecidable**.
7. **Subset problem (Is $L_1 \subseteq L_2$?):** Subset problem is **undecidable**.
8. **Co-finiteness problem (Is L' finite ?):** This problem is **undecidable**.
9. **Ambiguity problem:** Ambiguity problem is **undecidable**.

4.11.3 Decidability Problems for Recursive Languages

1. Membership problem: It is **decidable**.
   Proof: Construct halting turing machine for the given recursive language. Run the given string on halting turing machine. If it reaches to final state then given string is a member of recursive language. Otherwise it reaches to non-final state and halts for non-member strings.
2. Emptiness problem: It is **undecidable**.
3. Finiteness problem: It is **undecidable**.
4. Equivalence problem: It is **undecidable**.
5. Totality problem: It is **undecidable**.
6. Intersection empty problem: It is **undecidable**.
7. Subset Problem: It is **undecidable**.
8. Co finiteness problem: It is **undecidable**.
9. Ambiguity problem: It is **undecidable**.
10. State entry problem: It is **decidable** for recursive languages.
11. Halting problem of a turing machine: It is **decidable**.
    Proof: For halting turing machine, halting problem is decidable. It halts at final state if the string is acceptable or it halts at non final state if the string is not acceptable.
12. Empty word halting problem: It is **decidable**.
    Proof: Empty word halting problem is that by reading ε whether the machine will halt or not is decidable. Because Halting turing machine always halts for any given string.
13. Uniform halting problem: It is **decidable**.
    Proof: This problem indicates that $\forall x \in \Sigma^*$, whether the given turing machine reaches the halt state or not. Halting turning machine always halts for the valid strings and invalid strings.
14. Blank tape halting problem: It is **decidable**.
    Proof: Starting from blank tape whether the turing machine goes to halt state or not.

4.11.4 Decidability Problems for Recursively Enumerable Languages

1. Membership problem: It is **undecidable**.
2. Emptiness problem: It is **undecidable**.
3. Finiteness problem: It is **undecidable**.
4. Equivalence problem: It is **undecidable**.
5. Totality problem: It is **undecidable**.
6. Intersection empty problem: It is **undecidable**.
7. Subset Problem: It is **undecidable**.
8. Co finiteness problem: It is **undecidable**.
9. Ambiguity problem: It is **undecidable**.
10. State entry problem: It is **undecidable**.
11. Halting problem of a turing machine: It is **undecidable**.
12. Empty word halting problem: It is **undecidable**.
13. Uniform halting problem: It is **undecidable**.
14. Blank tape halting problem: It is **undecidable**.

4.12 Terminology of Problems

- **Decidable:** Decidable problem is called as totally decidable or turing decidable or recursive language.
- **Partially decidable:** Partially decidable problem is called as turing computable or recursively enumerable.
- **Undecidable:** Undecidable is partially decidable or totally not decidable. Partially decidables are recursive enumerable and totally not decidable are non-recursively enumerable.
- **Solvable:** There exists a halting machine or halting program for solvable problem. Solvable problems are decidable.
- **Partially solvable:** There exist a TM that solves the problem by halting only for some of inputs.
- **Unsolvable:** No machine exist for uncomputable function or unsolvable problem. Unsolvable problems are totally not decidable.

4.13 Important Points of Undecidability

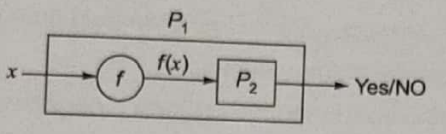
Problem	Regular Language	DCFL	CFL	Recursive Language	REL
Is W in L? (membership problem)	✓	✓	✓	✓	×
Is L = ϕ? (emptiness problem)	✓	✓	✓	×	×
Is L = Finite? (Finiteness problem)	✓	✓	✓	×	×
Is L1 = L2? (equivalence problem)	✓	✓	×	×	×
Is L1 $\subseteq$ L2? (subset problem)	✓	×	×	×	×
Is 'L' regular? (regularity problem)	✓	✓	×	×	×
Is L ambiguous (ambiguity problem)	✓	✓	×	×	×
Is L = Σ^* (Universality problem)	✓	✓	×	×	×
Is L1 $\cap$ L2 = ϕ? (disjoint problem)	✓	×	×	×	×
Is L = R? Where, R is regular language	✓	✓	×	×	×
Is L1 $\cap$ L2 = L3? Where L1, L2 and L3 are in same class	✓	×	×	✓	✓
Is the complement of a language is also a language of the same type.	✓	✓	×	✓	×

✓ is decidable, × is undecidable

4.14 Reduction

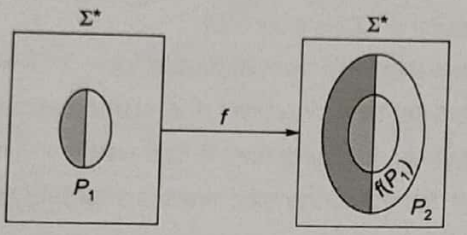
4.14.1 P_1 is Reducible to P_2 ($P_1 \leq P_2$)

- Problem P_1 is reducible to a problem P_2 means "*the problem P_2 is atleast as hard as problem P_1*".



or

- $\forall x, x \in P_1$ iff $f(x) \in P_2$; where f is many to one reduction from P_1 to P_2, which is denoted as $(P_1 \leq_m P_2)$



$P_1 \leq P_2$: Problem P_1 is reducible to a problem P_2
$P_1 \leq_m P_2$: Problem P_1 is many to one reducible to a problem P_2
$P_1 \leq_p P_2$: Problem P_1 is polynomially reducible to a problem P_2

4.14.2 Properties of Reduction

- If $P_1 \leq P_2$ and P_1 is undecidable then P_2 is also undecidable.
- If $P_1 \leq P_2$ and P_2 is undecidable then P_1 need not be undecidable.
- If $P_1 \leq P_2$ and P_2 is decidable then P_1 is also decidable.
- If $P_1 \leq P_2$ and P_1 is decidable then P_2 need not be decidable.
- If $P_1 \leq P_2$ and P_2 is recursive then P_1 is also recursive.
- If $P_1 \leq P_2$ and P_1 is recursive then P_2 need not be recursive.
- If $P_1 \leq P_2$ and P_2 is recursive enumerable then P_1 is also recursive enumerable.
- If $P_1 \leq P_2$ and P_1 is recursive enumerable then P_2 need not be recursive enumerable.
- If $P_1 \leq P_2$ and P_2 is P problem then P_1 is also P-problem.
- If $P_1 \leq P_2$ and P_1 is P problem then P_2 need not be P problem.
- If $P_1 \leq P_2$ and P_2 is NP problem then P_1 is also NP problem.
- If $P_1 \leq P_2$ and P_1 is NP problem then P_2 need not be NP problem.
- If $P_1 \leq P_2$ and $P_2 \leq P_3$ then $P_1 \leq P_3$ (transitivity).
- If $P_1 \leq P_2$ and $P_2 \leq P_1$ then P_1 and P_2 are polynomially equivalent.
- If $P_1 \leq P_2$ and P_1 is not REL then P_2 is also not REL.
- If $P_1 \leq P_2$ and P_1 is not P problem then P_2 is also not P problem.
- If $P_1 \leq P_2$ and P_1 is not recursive problem then P_2 is also not recursive problem.

Summary

- The power set of an infinite set is not countable.
- Both recursive enumerable set and recursive set are countable.
- If L is recursive then $\overline{L}$ is also recursive language.
- If both L and $\overline{L}$ are recursive enumerable then both L and $\overline{L}$ are recursive languages.
- Recursive language is turing decidable.
- Recursive enumerable language is turing recognizable.
- Algebraic numbers (root of polynomials with algebraic coefficients) are turing computable.
- Transcendental mathematical constants (e, π, etc.) are turing computable.
- Addition, Multiplication, Subtraction, Exponential, Factorial, etc. are turing computable.
- If Σ is a finite set then Σ^* is countable.
- If Σ is a finite set then 2^{Σ^*} is uncountable.
- If A is countable set then powerset of A is uncountable.
- If A and B are countable sets then $A \cup B$ and $A \times B$ are countable.
- The number of distinct turing machines is countably infinite.
- The number of distinct push down machines is countably infinite.
- The number of distinct finite state machines is countably infinite.
- All uncomputable functions are not turing computable.
- Halting problem of turing machine is undecidable.
- State entry problem of turing machine is undecidable.
- Empty word halting problem is undecidable.
- Does a given TM M halt on all inputs? is undecidable.
- Does TM M halt for any input i.e. is $L(M) = \phi$? is undecidable.
- Does TM M halt when given a blank input tape? is undecidable.
- Do two Turing machines M_1 and M_2 accept the same language i.e. $N(M_1) = N(M_2)$? is undecidable.
- Is the language $L(M)$ finite? is undecidable.
- Does $L(M)$ contain any two strings of the same length? is undecidable.
- Does $L(M)$ contain a string of length k, for some given $k \geq 1$? is undecidable.
- **Rice's theorem:** Every non-trivial property of the recursively enumerable language is undecidable. (L is non-trivial if both the sets L and $\overline{L}$ are non-empty)
- Post Correspondence Problem (PCP) is undecidable.
- Modified Post Correspondence Problem (MPCP) is undecidable.
- $L_u = \{<M, w> | M \text{ is a turing machine that accepts } w\}$ is recursive enumerable but not recursive.
- $L_d = \{w_i | M_i \text{ is a turing machine that does not accepts } w_i\}$ is non-recursive enumerable language.
- $L_{ne} = \{M | L(M) \neq \phi\}$ is recursive enumerable language but not recursive.
- The maximum number of 1's left on the tape after TM halts is uncomputable.

- The maximum number of moves than can made by TM is uncomputable.
- Every decidable language is enumerable.
- Every enumerable set is countable.
- Every countable set is computable.
- **Equivalent computation models:** Unrestricted grammars, Lambda calculus, Cellular automata, DNA computing, quantum computing, recursive enumerable language, and turing machines are equivalent models.
- If L_1 and L_2 are recursively enumerable languages then $L_1 \cup L_2$, $L_1 \times L_2$, $L_1 \cap L_2$, L_1^*, L_1^+ and L_1^{Rev} are also recursively enumerable languages.
- If S is an alphabet and $L \subseteq S^*$ is recursive then $S^* - L$ is also recursive. [Q $L^C = S^* - L$].
- If L_1 and L_2 are recursive languages then $L_1 \cup L_2$, $L_1 - L_2$, $L_1 \cup L_2$, L_1^*, L_1^+ and L_1^{Rev} are also recursive languages.
- L and L^C are recursively enumerable languages iff L is recursive.
- Recursive languages are enumerable in lexicographical order by TM.
- Recursive languages are decidable.
- Recursive enumerable languages are semi-decidable.
- Every recursive language is recursively enumerable.
- A language is recursively enumerable iff it is Turing-enumerable.
- All recursive enumerable languages are not recursive.
- Primitive Recursive Functions $\subseteq$ Total Recursive Functions $\subseteq$ Partial Recursive Functions.
- Non-RELs can be proved using diagonalization, complementation of REL, reduction from some non-REL, or rice theorem with non trivial property.
- If $L_1 \leq L_2$ and L_2 is decidable then L_1 is also decidable.
- If $L_1 \leq L_2$ and L_1 is undecidable then L_2 is also undecidable.
- *NP*-complete problems are hardest problems in NP.
- If NP-complete is in P then P = NP.
- If $L_1, L_2 \in$ NP, $L_1 \leq L_2$, and L_1 is NP complete then L_2 is NP complete.
- If $L_1, L_2 \in$ NP, $L_1 \leq L_2$, and L_2 is NP complete then L_1 need not be NP complete but in NP.
- P $\subseteq$ NP $\subseteq$ Recursive.
- NP complete $\subseteq$ NP and NP complete $\subseteq$ NP Hard.
- P problems are tractable.
- Whether P = NP is undecidable problem.
- Every P problem is in NP
- Every NPC problem is in NP
- Every NPC problem is in NPH
- NPC = NP $\cap$ NPH
- If $L_1 \in$ NPC and $L_1 \in$ P then P = NP

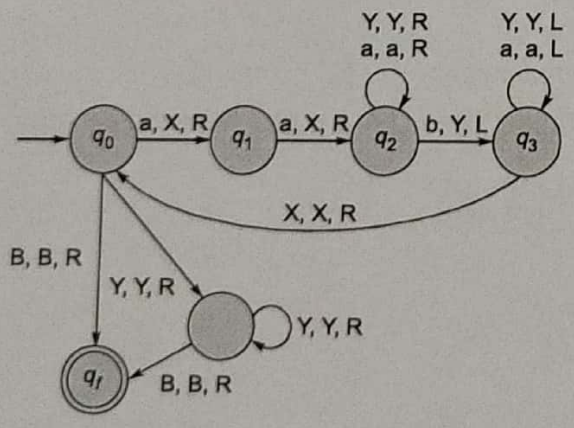
Student's Assignments 1

Q.1 Recursive languages are not closed under _____ operation.
(a) Union (b) Intersection
(c) Reversal (d) Substitution

Q.2 Consider the following languages.
A = {<M> | TM 'M' accepts atmost 2 distinct inputs}
B = {<M> | TM 'M' accepts more than 2 distinct inputs}
Identify the correct statement from the following.
(a) A is turing recognizable, B is not turing recognizable.
(b) B is turing recognizable, A is not turing recognizable.
(c) Both A and B are turing recognizable
(d) Neither A nor B is turing recognizable.

Q.3 Let L_1 be Recursive language and L_2 be Recursive enumerable language. Then $L_2 - L_1$ is _____
(a) Recursive language
(b) Recursive enumerable language
(c) Non-recursive enumerable language
(d) None of these

Q.4 Consider the following turing machine.
Identify the language accepted by the below TM:



(a) $\{a^m b^n | m > n\}$ (b) $\{a^m b^n | m = 2n\}$
(c) $\{a^m b^n | n = 2m\}$ (d) $\{a^m b^n | m \geq n\}$

Q.5 The complement of a recursively enumerable but not recursive language is:
(a) recursively enumerable
(b) recursive
(c) not recursively enumerable
(d) context free grammar

Q.6 Total recursive functions are similar to:
(a) recursive languages
(b) recursively enumerable languages
(c) both recursive and recursively enumerable languages
(d) recursively enumerable but not recursive languages

Q.7 Partial recursive functions are similar to
(a) recursive languages
(b) recursively enumerable languages
(c) recursively enumerable but not recursive
(d) None of these

Q.8 Turing machine is a:
(a) finite automata with R/W head
(b) finite automata with infinite tape
(c) finite automata with R/W head, infinite tape and turn around capability
(d) none of the above.

Q.9 The language accepted by halting turing machine is:
(a) recursively enumerable language
(b) recursive language
(c) context free language
(d) Context sensitive language

Q.10 Find the missing transitions *a*, *b*, *c* in the following transition diagram which does proper subtraction operation.

	0	1	B
→ q_0	c	(q_5, B, R)	–
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	–
q_2	a	$(q_2, 1, R)$	(q_4, B, L)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	$(q_4, 0, L)$	b	$(q_6, 0, L)$
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
(*q*6)	–	–	–

(a) $(q_1, B, R), (q_3, 1, L), (q_4, B, L)$
(b) $(q_3, 1, L), (q_4, B, L), (q_1, B, R)$
(c) $(q_3, 0, R), (q_4, B, L), (q_1, B, R)$
(d) None of these

Q.11 Recursive languages are closed under which of the following operations.
(a) Homomorphism (b) Substitution
(c) Both (a) and (b) (d) Neither (a) nor (b)

Q.12 If there exist a turing machine T for a language which halts for every word present in the language and either rejects or loops for every word that is not in the language, then that language is:
(a) recursive language
(b) recursively enumerable language
(c) both recursive and recursively enumerable languages
(d) None of the above

Q.13 Which of the following is true:
(a) Power of NDTM is less than DTM
(b) Power of NDTM is same as DTM
(c) Power of NDTM is greater than DTM
(d) Can't say

Q.14 What is the operation that is performed by following turing machine transitions:

	0	1	B
→ q_0	$(q_1, 1, R)$	$(q_0, 1, R)$	–
q_1	–	$(q_1, 1, R)$	(q_2, B, L)
q_2	–	(q_3, B, L)	–
q_3	–	$(q_3, 1, L)$	(q_4, B, R)
((q_4))	–	–	–

(a) Subtraction (b) Multiplication
(c) Addition (d) None of these

Q.15 Which of the following statements is true regarding finite length C programs and Turing machines
(a) Each one can simulate the other
(b) The turing machines always halts which represents all C programs
(c) The C programs that always halt can simulate all turing machines
(d) All of the above

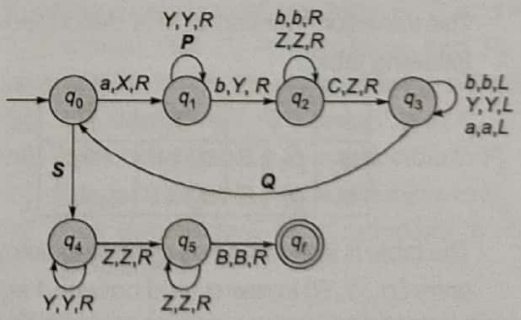
Q.16 Type 0 or unrestricted grammars
(a) generate all the sets that are accepted by halting turing machines only
(b) generate all the sets that are accepted by all types of turing machine
(c) generate languages that are not recursive
(d) generate languages that are not recursive language

Q.17 Choose incorrect statement
(a) If a problem P can be solved by a computer then it can also be solved by turing machine
(b) Problems that cannot be solved by a turing machine cannot be solved by any other machine
(c) Turing machine is most powerful among all machines such as FA, PDA, LBA and TM
(d) None of the above.

Q.18 All partial recursive functions can be computed by
(a) push down automata
(b) linear bounded automata
(c) turing machine
(d) None of the above

Q.19 Which of the following operations are closed for both recursive and recursively enumerable languages
(a) union, complement, intersection
(b) homomorphism, inverse homomorphism, substitution
(c) union, concatenation, reversal
(d) None of the above

Q.20 Find the missing transitions in following turing machine that accepts a language
$L = \{a^n b^n c^n \mid n \geq 1\}$



(a) $p = a,a,L \quad Q = Y,Y,L \quad S = Y,Y,R$
(b) $P = a,a,R \quad Q = X,X,R \quad S = Y,Y,R$
(c) $P = b,b,R \quad Q = X,X,L \quad S = X,X,R$
(d) None of the above

Q.21 $L = \{a^{n!} \mid n > 0\}$ then L is
(a) recursive only
(b) recursively enumerable but not recursive
(c) both recursive and recursively enumerable
(d) neither recursive nor recursively enumerable

Q.22 Nobody knows yet if P = NP. Consider the language L defined as follows:

$$L = \begin{cases} (0+1)^* \text{ if } P = NP \\ \phi \text{ otherwise} \end{cases}$$

Which of the following statements is true?
(a) L is recursive
(b) L is recursively enumerable but not recursive
(c) L is not recursively enumerable
(d) Whether L is recursive or not will be known after we find out if P = NP

Q.23 If the strings of a language L can be effectively enumerated in lexicographic (i.e., alphabetic) order, which of the following statements is true?
(a) L is necessarily finite
(b) L is regular but not necessarily finite
(c) L is context free but not necessarily regular
(d) L is recursive but not necessarily context free

Q.24 A single tape Turing Machine M has two states q_0 and q_1, of which q_0 is the starting state. The tape alphabet of M is {0, 1, B} and its input alphabet is {0, 1}. The symbol B is the blank symbol used to indicate end of an input string. The transition function of M is described in the following table.

	0	1	B
q_0	$q_1, 1, R$	$q_1, 1, R$	Halt
q_1	$q_1, 1, R$	$q_0, 1, L$	q_0, B, L

The table is interpreted as illustrated below. The entry $(q_1, 1, R)$ in row q_0 and column 1 signifies that if M is in state q_0 and reads 1 on the current tape square, then it writes 1 on the same tape square, moves its tape head one position to the right and transitions to state q_1. Which of the following statements is true about M?
(a) M does not halt on any string in $(0 + 1)^*$
(b) M does not halt on any string in $(00 + 1)^*$
(c) M halts on all string ending in a 0
(d) M halts on all string ending in a 1

Q.25 Let L_1 be a recursive language, and let L_2 be a recursively enumerable but not a recursive language. Which one of the following is TRUE?
(a) $\overline{L_1}$ is recursive and $\overline{L_2}$ is recursively enumerable
(b) $\overline{L_1}$ is recursive and $\overline{L_2}$ is not recursively enumerable
(c) $\overline{L_1}$ and $\overline{L_2}$ are recursively enumerable
(d) $\overline{L_1}$ is recursively enumerable and $\overline{L_2}$ is recursive

Q.26 For $s \in (0 + 1)^*$ let $d(s)$ denote the decimal value of s(e.g. $d(101) = 5$).
Let $L = \{s \in (0 + 1)^* \mid d(s) \bmod 5 = 2$ and $d(s) \bmod 7 \neq 4\}$
Which one of the following statements is true?
(a) L is recursively enumerable, but not recursive
(b) L is recursive, but not context-free
(c) L is context-free, but not regular
(d) L is regular

Q.27 Let L_1 be regular language, L_2 be a deterministic context-free language and L_3 a recursively enumerable, but not recursive, language. Which one of the following statements is false?
(a) $L_1 \cap L_2$ is a deterministic CFL
(b) $L_3 \cap L_1$ is recursive
(c) $L_1 \cup L_2$ is context free
(d) $L_1 \cap L_2 \cap L_3$ is recursively enumerable

Q.28 Which of the following is true for the language $\{a^p \mid p$ is a prime$\}$?
(a) It is not accepted by a Turning Machine
(b) It is regular but not context-free
(c) It is context-free but not regular
(d) It is neither regular nor context-free, but accepted by a Turing machine

Q.29 If L and $\overline{L}$ are recursively enumerable then L is _____.
(a) regular (b) context-free
(c) context-sensitive (d) recursive

Q.30 Let L1 be a recursive language. Let L2 and L3 be language that are recursively enumerable but not recursive. Which of the following statements is not necessarily true?
(a) L2 – L1 is recursively enumerable
(b) L1 – L3 is recursively enumerable
(c) L2 ∩ L3 is recursively enumerable
(d) L2 ∪ L3 is recursively enumerable

Q.31 Which of the following statements is/are FALSE?
1. For every non-deterministic Turing machine, there exists an equivalent deterministic Turing machine.
2. Turing recognizable languages are closed under union and complementation.
3. Turing decidable languages are closed under intersection and complementation.
4. Turing recognizable languages are closed under union and intersection.

(a) 1 and 4 only
(b) 1 and 3 only
(c) 2 only
(d) 3 only

Q.32 Let L be a language and $\overline{L}$ be its complement. Which one of the following is NOT a viable possibility?
(a) Neither L nor $\overline{L}$ is recursively enumerable (r.e.).
(b) One of L and $\overline{L}$ is r.e. but not recursive; the other is not r.e.
(c) Both L and $\overline{L}$ are r.e. but not recursive.
(d) Both L and $\overline{L}$ are recursive.

Q.33 Let $<M>$ be the encoding of a Turing machine as a string over $\Sigma = \{0, 1\}$. Let $L = \{<M> | M$ is a *Turing machine that accepts a string of length* 2014$\}$. Then, L is
(a) decidable and recursively enumerable
(b) undecidable but recursively enumerable
(c) undecidable and not recursively enumerable
(d) decidable but not recursively enumerable

Linked Data for (Q.34 and Q.35)

Let L_1 is reducible to L_2 and L_2 is reducible to L_3.

Q.34 If L_3 is decidable then which of the following statement is correct?
(a) L_1 is decidable but L_2 is undecidable
(b) L_2 is decidable but L_1 is undecidable
(c) Both L_1 and L_2 are decidable
(d) None of these

Q.35 Which of the following can be valid using the above languages L_1 and L_2?
(a) $L_1 \cap L_2$ is decidable
(b) Complement of L_1 is undecidable
(c) Homomorphism of L_1 is decidable
(d) None of these

Q.36 Given a turing machine T and a step-counting function f, is the language accepted by T in Time (f)? This problem is
(a) solvable
(b) unsolvable
(c) uncertain
(d) none of these

Q.37 A countable union of countable sets is not
(a) countable
(b) uncountable
(c) countably infinite
(d) denumerable

Q.38 Which of the following is undecidable
(a) Equivalence of regular languages
(b) Equivalence of context free languages
(c) Finiteness check on context free language
(d) Emptiness of regular languages

Q.39 Consider three decision problems P_1, P_2 and P_3. It is known that P_1 is decidable and P_2 is undecidable. Which one of the following is TRUE?
(a) P_3 is decidable if P_1 is reducible to P_3
(b) P_3 is undecidable if P_3 is reducible to P_2
(c) P_3 is undecidable if P_2 is reducible to P_3
(d) P_3 is decidable if P_3 is reducible to P_2's complement

Q.40 Which of the following problems is undecidable?
(a) Membership problem for CFGs
(b) Ambiguity problem for CFGs
(c) Finiteness problem for FSAs
(d) Equivalence problem for FSAs

Q.41 Which of the following are decidable?

1. Whether the intersection of two regular languages is infinite.
2. Whether a given context-free language is regular.
3. Whether two push-down automata accept the same language.
4. Whether a given grammar is context-free.

(a) 1 and 2 (b) 1 and 4
(c) 2 and 3 (d) 2 and 4

Q.42 Which of the following problems are decidable?

1. Does a given program ever produce an output?
2. If L is a context-free language, then, is $\overline{L}$ also context-free?
3. If L is a regular language, then, is $\overline{L}$ also regular.
4. If L is a recursive language, then, is $\overline{L}$ also recursive.

(a) 1, 2, 3, 4 (b) 1, 2
(c) 2, 3, 4 (d) 3, 4

Q.43 Which of the following is/are undecidable?

1. G is CFG. Is $L(G) = \phi$?
2. G is a CFG. Is $L(G) = \Sigma^*$?
3. M is a Turing machine. Is L(M) regular?
4. A is a DFA and N is an NFA. Is L(A) = L(N)?

(a) 3 only
(b) 3 and 4 only
(c) 1, 2 and 3 only
(d) 2 and 3 only

Q.44 Let $A \leq_m B$ denotes that language A is mapping reducible (also known as many-to-one reducible) to language B. Which one of the following is FALSE?

(a) If $A \leq_m B$ and B is recursive then A is recursive.
(b) If $A \leq_m B$ and A is undecidable then B is undecidable.
(c) If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.
(d) If $A \leq_m B$ and B is not recursively enumerable then A is not recursively enumerable.

Q.45 Let Σ be a finite non-empty alphabet and let 2^{Σ^*} be the power set of Σ^*. Which one of the following is TRUE?

(a) Both 2^{Σ^*} and Σ^* are countable
(b) 2^{Σ^*} is countable and Σ^* is uncountable
(c) 2^{Σ^*} is uncountable and Σ^* is countable
(d) Both 2^{Σ^*} and Σ^* are uncountable

Q.46 Which one of the following problems is undecidable?

(a) Deciding if a given context-free grammar is ambiguous.
(b) Deciding if a given string is generated by a given context-free grammar.
(c) Deciding if the language generated by a given context-free grammar is empty.
(d) Deciding if the language generated by a given context-free grammar is finite.

Q.47 Which one of the following is the strongest correct statement about a finite language over some finite alphabet Σ?

(a) It could be undecidable
(b) It is turing-machine recognizable
(c) It is a regular language
(d) It is a context-sensitive language

Answer Key:

1. (d)	**2.** (b)	**3.** (b)	**4.** (b)	**5.** (c)
6. (a)	**7.** (b)	**8.** (c)	**9.** (b)	**10.** (c)
11. (d)	**12.** (b)	**13.** (b)	**14.** (c)	**15.** (a)
16. (b)	**17.** (d)	**18.** (c)	**19.** (c)	**20.** (b)
21. (c)	**22.** (a)	**23.** (d)	**24.** (a)	**25.** (b)
26. (d)	**27.** (b)	**28.** (d)	**29.** (d)	**30.** (b)
31. (c)	**32.** (c)	**33.** (b)	**34.** (c)	**35.** (a)
36. (b)	**37.** (b)	**38.** (b)	**39.** (c)	**40.** (b)
41. (b)	**42.** (d)	**43.** (d)	**44.** (d)	**45.** (c)
46. (a)	**47.** (c)			

Student's Assignments 2

Q.1 A Turing machine is more powerful than finite state machine because
(a) head movement is confined to one direction
(b) head moment is in both directions (L, R)
(c) it has capability remember arbitrary long sequence of input string
(d) all the above

Q.2 A pushdown automata behaves like a Turing machine when it has number of auxiliary memory.
(a) 0 (b) Exactly 2
(c) 2 or more (d) Both (b) and (c)

Q.3 The languages $L = \{a^n b^n a^n : n \geq 1\}$ is recognised by
(a) Turing Machine (b) LBA
(c) Finite Automata (d) Both (a) and (b)

Q.4 If Turing machine accepts all the words of the languages L and rejects or loops for other words which are not in L, then L is said to be
(a) Recursive enumerable (b) Recursive
(c) CFL (d) None of these

Q.5 If a Turing machine halts for each and every word of a language L and rejects other, then L is said to be
(a) Recursive enumerable (b) Recursive
(c) CFL (d) None of these

Q.6 Universal Turing machine (UTM) influenced the concepts of
(a) Computability
(b) Implementation of programming language
(c) Program and data is in same memory
(d) All of the above

Q.7 The number of symbols necessary to simulate a Turing machine with m symbols and n states is
(a) $4mn + m$ (b) $4mn + n$
(c) $m + n$ (d) None of these

Q.8 A universal Turing machine is a
(a) Reprogrammable Turing machine
(b) Two-tape TM
(c) Single tape TM
(d) None of these

Q.9 The difference between a read-only Turing machine and a two-way finite state machine is
(a) Head movement (b) Finite control
(c) Storage capacity (d) Power

Q.10 An off-line Turing machine is
(a) A special type of multi-tape Turing machine
(b) A kind of multi-tracks Turing machine
(c) A kind of single-track Turing machine
(d) None of these

Q.11 Choose the wrong statement:
(a) Power of NTM and TM is same
(b) For $n \geq 2$, nPDA has same power as a TM
(c) For $n \geq 3$, nPDA and 2PDA have same power
(d) Power of NTM and TM is not same

Q.12 Four pairs are following. In each pair both objects have some common thing. Choose the odd pair
(a) (TM, 2PDA)
(b) (Computer, UTM)
(c) (2PDA, nPDA), Assume $n \geq 3$
(d) (FA, PDA)

Q.13 We think of a TM's transition functions as a
(a) Computer system (b) Software
(c) hardware (d) All the above

Q.14 Church's Thesis supports
(a) A TM as a general-purpose computer system
(b) A TM as an algorithm and an algorithm as a TM
(c) Both (a) and (b)
(d) None of the above

Q.15 A random access machine (RAM) and a TM are different in
(a) Power (b) Accessing
(c) Storage (d) Both (b) and (c)

Q.16 Choose the correct statement:
(a) Recursive set $\subseteq$ recursive enumerable set
(b) Total function is same as partial function
(c) Recursive sets are analogous to total functions
(d) Both (a) and (c)

Q.17 Given S = {*a*, *b*}, which one of the following sets is not countable?
(a) The set all strings over Σ
(b) The set of all language over Σ
(c) The set of all binary strings
(d) The set of all languages over Σ accepted by Turing machines

Q.18 In which of the following class of machines the statement given below is false?
"For every non-deterministic machine M_1, there exists as equivalent deterministic machine M_2 recognizing the same language."
(a) Push down automata
(b) Finite automata
(c) Turing machine
(d) None of the above

Q.19 Which of the following conversion is not possible (algorithmically)?
(a) Regular grammar to context-free grammar
(b) Non-deterministic finite state automata to deterministic finite state automata
(c) Non-deterministic pushdown automata to deterministic pushdown automata
(d) None deterministic Turing machine to deterministic Turing machine

Q.20 Match **List-I** with **List-II** and select the correct answer using the codes given below the lists:

List-I
A. Lexical analyzer
B. Parsing
C. Computing
D. Non-deterministic but

List-II
1. Pushdown Automata
2. Turing machine
3. Finite state automata
4. Non-deterministic FA finite machine

Codes:

	A	B	C	D
(a)	3	4	2	1
(b)	3	1	2	4
(c)	3	1	4	2
(d)	3	2	1	4

Q.21 The word enumerable means
(a) Countable
(b) Able to be put in some order
(c) Calculable
(d) Both (a) and (b)

Q.22 A recursive enumerable language is
(a) Accepted by TM
(b) Not accepted by TM
(c) Some times accepted and some time not
(d) None of these

Q.23 For a recursive enumerable language
(a) TM halts always
(b) TM does not halt
(c) Some time halts but some time not
(d) None of these

Q.24 Suppose *L* is a recursive enumerable language and TM M accepts *L*, then for $w \notin L$, the machine M
(a) Halts always
(b) Does not halt
(c) May or may not halt
(d) Rejects

Q.25 Suppose *L* is a recursive language and TM M accepts *L*, then for $w \notin L$, the machine M
(a) Halts and rejects
(b) Does not halt
(c) May or may not halt
(d) None of these

Q.26 Suppose L_1 and L_2 are recursive enumerable languages respectively over some alphabet Σ, then $L_1 \cup L_2$ is
(a) Recursively enumerable
(b) Recursive
(c) Context-sensitive
(d) None of these

Q.27 Choose the incorrect statement
(a) All recursive language are recursive enumerable
(b) All recursive enumerable languages are recursive
(c) All recursive languages are TM computable
(d) None of these

Q.28 Choose the correct statement
(a) The number of TMs over some alphabet is countable
(b) The number of TMs over some alphabet is countable but not enumerable
(c) Some languages are not TM computable
(d) Both (a) and (c)

Q.29 Some languages are not TM computable because
(a) No algorithm is there
(b) Algorithm is there but we can't implement
(c) These are not enumerable
(d) Both (a) and (c)

Q.30 If Σ is a finite set then
(a) 2^{Σ} is infinite but countable
(b) 2^{Σ} is finite but not countable
(c) 2^{Σ^*} is infinite but countable
(d) 2^{Σ^*} is infinite and not countable

Q.31 The diagonalization language, L_d is a
(a) Recursive language
(b) Recursive enumerable but not recursive
(c) Non-recursively - enumerable (non-RE) language
(d) Both (b) and (c)

Common Data for (Q.32 and Q.33)
Consider the transition table of a TM given below. Here "b" represents the blank symbol.

δ	0	1	b
$\rightarrow q_0$	$q_0, 0, R$	$q_1, 0, R$	$q_2, 0, R$
q_1	$q_1, 0, R$	$q_0, 0, R$	–
$^*q_2$	–	–	–

Q.32 Which of the following strings will not accepted?
(a) 010101 (b) 101010
(c) 110011 (d) Both (a) and (b)

Q.33 The given turing machine accept
(a) set of all even palindromes over {0, 1}
(b) strings over {0, 1} containing even number of 1's
(c) strings over {0, 1} containing even number of 1's and odd no. of 0's
(d) string over {0, 1} starting with zero

Q.34 Halting problem/language is
(a) RE as well as recursive
(b) Recursive and NP
(c) RE but not recursive
(d) Neither recursive nor RE

Q.35 If we want to show a problem not to be RE, then
(a) Only L_d can be used
(b) Only L_u can be used
(c) Either L_u or L_d can be used
(d) None of these

Q.36 Universal TM influenced the concept of
(a) stored-program computers
(b) interpretative implementation of programming language
(c) computability
(d) all of the above

Common Data for (Q.37 and Q.39):
Assume X_1, will always be symbol, 0, X_2, will be 1, and X_3 will be B (the blank symbol).
Refer direction L as D_1 and direction R as D_2.
Encode the transition rule, $\delta(q_i, X_j) = (q_k, X_\ell, D_m)$ for some integers i, j, k, l, m by the string $0^i\ 10^j\ 10^k\ 10^l\ 10^m$.
Let the TM M = $(\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q, B, \{q_2\})$

Q.37 Encode $\delta(q_3, 1) = (q_2, 0, R)$
(a) 0100100010100 (b) 00010101001010
(c) 00010010010100 (d) None of these

Q.38 If the transition function δ of given TM consists of the rules
$\delta(q_1, 1) = (q_3, 0, R)$ $\quad$ $\delta(q_2, B) = (q_3, 1, R)$
$\delta(q_1, 0) = (q_2, 1, R)$ $\quad$ $\delta(q_3, B) = (q_3, 1, L)$
$\delta(q_2, 1) = (q_2, 0, R)$
Then which of the following is valid code of a transition rule of the given TM?
(a) 01010100101 (b) 0011010100
(c) 0001000100010010 (d) 101001000100

Q.39 If a code for the entire TM M consists of all codes for the transitions, in some order, separated by pairs of 1's $C_1\ 11\ C_2\ 11\ C_3\ 11\ \ldots\ C_{n-1}\ 11\ C_n$ where each of the C's is the code form one transition of M; and if the given machine has only five transition rules, how may 1s will be their in the code for TM?

(a) 13
(b) 28
(c) 30
(d) Not fixed, Depends on transition rules

Q.40 Which of the following is false?
(a) The union of two recursive languages is recursive.
(b) The complement of a recursive language is recursive.
(c) The union of two recursively enumerable language is recursively enumerable.
(d) If a language L and $\overline{L}$ are both recursively enumerable then L is recursive but not $\overline{L}$.

Q.41 Which of the following functions are computable with turning Machine?
(a) $n^*(n-1)^*(n-2)...^*2^*1$
(b) $\lceil \log_2 n \rceil$
(c) 2^{2^n}
(d) All of the above

Q.42 Let M be a turing machine has Q = $\{q_0, q_1, q_2, q_3, q_4\}$ a set of states, input alphabet is {0, 1}. The tape alphabet is {0,1, B, X, Y}. The symbol B is used to represent end of input string. The final state is q_4. The transitions are as follows.
1. $(q_0, 0) = (q_1, X, R)$
2. $(q_0, Y) = (q_3, Y, R)$
3. $(q_1, 0) = (q_1, 0, R)$
4. $(q_1, 1) = (q_2, Y, L)$
5. $(q_1, Y) = (q_1, Y, R)$
6. $(q_2, 0) = (q_2, 0, L)$
7. $(q_2, X) = (q_0, X, R)$
8. $(q_2, Y) = (q_2, Y, L)$
9. $(q_3, Y) = (q_3, Y, R)$
10. $(q_3, B) = (q_4, B, R)$

Which of the following is true about M?
(a) M halts on L having 100 as substring
(b) M halts on L having 101 as substring
(c) M halts on $L = 0^n1^n, n \geq 0$
(d) M halts on L not having 1100 substring

Q.43 If L_1 and L_2 are a pair of complementary languages. Which of the following statement is not possible?
(a) Both L_1 and L_2 are recursive.
(b) L_1 is recursive and L_2 is recursively enumerable but not a cursive.
(c) Neither L_1 nor L_2 is recursively enumerable.
(d) One is recursively enumerable but not recursive, the other is not recursively enumerable.

Q.44 A TM M = $(\{q_0, q_1, ... q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, q_6)$. The transitions are

$\delta(q_0, 0) = (q_1, B, R)$, $\delta(q_0, 1) = (q_2, 1, R)$,
$\delta(q_1, 0) = (q_1, 0, R)$, $\delta(q_1, 1) = (q_2, 1, R)$,
$\delta(q_2, 0) = (q_2, 1, R)$, $\delta(q_2, 1) = (q_3, 1, L)$,
$\delta(q_2, B) = (q_4, B, L)$, $\delta(q_3, 0) = (q_3, 0, L)$,
$\delta(q_3, 1) = (q_3, 1, L)$, $\delta(q_3, B) = (q_0, B, R)$,
$\delta(q_4, 0) = (q_4, 0, L)$, $\delta(q_4, 1) = (q_4, B, L)$,
$\delta(q_4, B) = (q_6, 0, R)$, $\delta(q_5, 0) = (q_5, B, R)$,
$\delta(q_5, 1) = (q_5, B, R)$, $\delta(q_5, B) = (q_6, B, R)$

What is the language accepted by TM?
(a) $0^* 10^*$
(b) $0^{(m-n)} \mid m < n$
(c) $(0 + 1)^*$
(d) $0^n 10^m \mid m \leq n$

Q.45 Referring to Turing machine **(in Q. 44)**, what will be the output when the input is $I_1 = 0100$ and $I_2 = 0010$?
(a) $I_1 = 0$ and I_2 = Blank
(b) I_1 = Blank and $I_2 = 0$
(c) $I_1 = 1$ and $I_2 = 1$
(d) None of the above

Q.46 A single tape turing machine M has two states q_0 and q_1, of which q_0 is the starting state. The tape alphabet of M is {0, 1, B} and its input alphabet is {0, 1}. The symbol B is the blank symbol used to indicate end of an input string. The transition function of M is described in the following table.

	0	1	B
q_0	$(q_1, 1, R)$	$(q_1, 1, R)$	Halt
q_1	$(q_1, 1, R)$	$(q_0, 1, L)$	(q_0, B, L)

Which of the following statement is true about M?
(a) M does not halt on any string in $(0 + 1)^+$
(b) M does not halt on any string in $(00 + 1)^*$

(c) M halts on all string ending in 0
(d) M halts on all string ending in a 1

Q.47 A problem is decidable if there is
(a) An algorithm that answers either "YES" or "NO"
(b) An algorithm that answers both "YES" & "NO"
(c) A TM that decides the problem
(d) Both the (a) and (c) statements

Q.48 PCP is first suggested by
(a) Allen Turing (b) Emil Post
(c) Kurt Godel (d) None of these

Q.49 PCP problem is to determine a
(a) String
(b) Match between two sets of strings
(c) Match between two strings
(d) None of these

Q.50 If L_1 and L_2 are two regular languages then the problem "Is $L_1 \oplus L_2$ (L_1 EX-OR L_2) regular?" is
(a) Decidable
(b) Undecidable
(c) Regular
(d) Both (a) and (c) statements

Q.51 PCP over {a} is
(a) Decidable always
(b) Undecidable always
(c) Some time decidable
(d) Undecidable some time

Q.52 Let G_1 and G_2 are two grammars and G_1 is regular grammar. The problem $L(G_1) = L(G_2)$ or not is decidable when
(a) G_2 is type 0 grammar
(b) G_2 is CFG
(c) G_2 is regular grammar
(d) G_2 is CSG

Q.53 If $L_1, L_2, \ldots, L_n$ are regular languages, then $L = L_1 \cup L_2 \cup \ldots \cup L_n$ is regular. This is
(a) Undecidable
(b) Decidable
(c) Dependent on nature of $L_1, L_2, \ldots, L_n$
(d) Partially decidable

Q.54 If L_1 is a recursively enumerable language, then whether L_1^c is recursively enumerable or not is
(a) Undecidable
(b) Decidable
(c) Dependent on nature of L_1
(d) Partially decidable

Q.55 If $L_1, L_2, \ldots L_n$ are regular languages, then $L = L_1 \oplus L_2 \oplus \ldots \oplus L_n$ is regular. This is
(a) Undecidable
(b) Decidable
(c) Dependent on nature of $L_1, L_2, \ldots, L_n$
(d) Partially decidable

Q.56 Deciding ambiguity of a language generated by a CFG is
(a) Decidable
(b) Depend on nature of the grammar
(c) Undecidable
(d) None of the above

Q.57 Choose the correct statement
(a) Recursive languages are closed under union and intersection
(b) $L = \{a^n b^n c^n : n \geq 1\}$ is recursive language
(c) Every recursive language is recursive enumerable
(d) All the above statements

Q.58 If a language L and its complement $\overline{L}$ are recursively enumerable then choose statement
(a) L is recursive but not $\overline{L}$
(b) Both L and $\overline{L}$ are recursive
(c) $\overline{L}$ is recursive but not L
(d) None of these

Q.59 Choose the correct statement
(a) Recursive languages are closed under complement
(b) Recursive enumerable language are closed under union
(c) If language L and its complement $\overline{L}$ are recursively enumerable then L and $\overline{L}$ are recursive
(d) All the above statements

Q.60 If there is a TM M for a problem set P. M is applied to any problem in the set P and terminates when answer is "YES" and may or may not terminate otherwise then P is
(a) Solvable (b) Unsolvable
(c) Partially solvable (d) None of these

Q.61 The statement "A TM can't solve HP" is
(a) False
(b) True
(c) Still an open question
(d) None of these

Q.62 Consider the problem X: "Given a TM M over the input alphabet S any state of M and a word $w \in S$, does the computation of M on w visit the state q? Which of the following statements about X is correct?
(a) X is decidable
(b) X is undecidable but partially decidable
(c) X is undecidable and not partially decidable
(d) X is not a decision problem

Q.63 Which of the following problem is undecidable for regular language?
(a) Membership (is $w \in L$?)
(b) Subset (is $L_1 \subseteq L_2$?)
(c) Infinite union (is $L_1 \cup L_2 \cup \ldots$ = regular)
(d) None of these

Q.64 Find the correct statement from the following?
(a) Subset operation is closed for regular languages
(b) Membership problem is decidable for recursive languages
(c) Equivalence problem is decidable for CFL's
(d) Equivalence problem is decidable for REL's

Q.65 Recursive languages are called as
(a) Turing decidable (b) Turing computable
(c) Partially decidable (d) None of these

Q.66 P problems are
(a) Decidable
(b) Semi-decidable
(c) Totally not decidable
(d) None of these

Q.67 Consider the following problems:
P1: Emptyness [is given L empty?]
P2: Finiteness [is given L finite?]
P3: Membership [is given L accepts given string?]
Find which of the following can decide all the above problems?
(a) FA, PDA, HTM, TM (b) FA, PDA, HTM
(c) FA, PDA (d) FA

Q.68 Ambiguity problem is decidable for _____
(a) Regular grammars
(b) Regular grammars and CFG's
(c) CFG's and CSG's
(d) None of these

Q.69 Find which of the following class is not countable
(a) Class of all context free languages over Σ
(b) Class of all regular languages over Σ
(c) Class of all recursive enumerable languages over Σ
(d) Class of all languages over Σ

Answer Key:

1. (c)	**2.** (c)	**3.** (d)	**4.** (a)	**5.** (b)
6. (d)	**7.** (a)	**8.** (a)	**9.** (c)	**10.** (a)
11. (d)	**12.** (d)	**13.** (b)	**14.** (c)	**15.** (d)
16. (d)	**17.** (b)	**18.** (a)	**19.** (c)	**20.** (b)
21. (d)	**22.** (a)	**23.** (c)	**24.** (c)	**25.** (a)
26. (a)	**27.** (b)	**28.** (d)	**29.** (d)	**30.** (d)
31. (c)	**32.** (d)	**33.** (b)	**34.** (c)	**35.** (a)
36. (d)	**37.** (c)	**38.** (c)	**39.** (b)	**40.** (d)
41. (d)	**42.** (c)	**43.** (b)	**44.** (a)	**45.** (a)
46. (a)	**47.** (d)	**48.** (b)	**49.** (b)	**50.** (a)
51. (a)	**52.** (c)	**53.** (b)	**54.** (a)	**55.** (b)
56. (c)	**57.** (d)	**58.** (b)	**59.** (d)	**60.** (c)
61. (b)	**62.** (b)	**63.** (c)	**64.** (b)	**65.** (a)
66. (a)	**67.** (c)	**68.** (a)	**69.** (d)	