

# Computer Science & Information Technology

## Theory of Computation

---

Comprehensive Theory

*with* Solved Examples and Practice Questions



**MADE EASY**  
Publications



### **MADE EASY Publications Pvt. Ltd.**

Corporate Office: 44-A/4, Kalu Sarai (Near Hauz Khas Metro Station), New Delhi-110016

E-mail: [infomep@madeeasy.in](mailto:infomep@madeeasy.in)

Contact: 011-45124660, 8860378007

Visit us at: [www.madeeasypublications.org](http://www.madeeasypublications.org)

### **Theory of Computation**

© Copyright by MADE EASY Publications Pvt. Ltd.

All rights are reserved. No part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photo-copying, recording or otherwise), without the prior written permission of the above mentioned publisher of this book.

First Edition: 2015

Second Edition : 2016

Third Edition : 2017

Fourth Edition : 2018

Fifth Edition : 2019

Sixth Edition : 2020

Seventh Edition : 2021

**Eighth Edition : 2022**

# Contents

## Theory of Computation

### Chapter 1

#### Grammars, Languages & Automata..... 2

1.1	Introduction.....	2
1.2	Chomsky Hierarchy.....	6
1.3	Automaton.....	7
1.4	Grammars.....	9
1.5	Equivalence of Languages, Grammars and Automata.....	9
1.6	Expressive Power of Automata.....	10
1.7	Applications of Automata.....	10
	<i>Student Assignments</i> .....	12

### Chapter 2

#### Regular Languages & Finite Automata....17

2.1	Introduction.....	17
2.2	Deterministic Finite Automata (DFA).....	19
2.3	Non-deterministic Finite Automata (NFA or NDFA).....	46
2.4	Epsilon NFA (e-NFA).....	61
2.5	Regular Expressions.....	62
2.6	Equivalence between Finite Automata and Regular Expressions.....	65
2.7	Regular Grammar.....	77
2.8	Closure Properties of Regular Languages.....	83
2.9	Identifying Non Regular Language.....	92
2.10	Decision Properties of Regular Language.....	93
2.11	Moore and Mealy Machines.....	94
2.12	Minimization of Finite Automata.....	103
2.13	Myhill – Nerode Theorem.....	109
	<i>Student Assignments</i> .....	113

### Chapter 3

#### Context Free Languages & Push down Automata .....141

3.1	Context Free Grammars.....	141
3.2	Context Free Language.....	149
3.3	Push Down Automata (PDA).....	149
3.4	Equivalence between CFL and CFG.....	164
3.5	Closure Properties of CFL's.....	164
3.6	Closure Properties of DCFL's.....	167
3.7	Decision Properties of CFL's.....	169
3.8	Non-Context Free Languages.....	173
3.9	Non-Regular Languages and CFL's.....	173
	<i>Student Assignments</i> .....	175

### Chapter 4

#### REC, RE Languages & Turing Machines, Decidability.....185

4.1	Turing Machine (TM).....	185
4.2	Turing Machine Construction.....	186
4.3	Turing's Thesis.....	190
4.4	Variation of Turing Machines.....	190
4.5	A Universal Turing Machine.....	192
4.6	Countable and Uncountable Sets.....	192
4.7	Recursively and Recursively Enumerable Languages.....	195
4.8	Closure Properties.....	199
4.9	Some Decidable Problems (Recursive Languages) on Turing Machines.....	203
4.10	Chomsky Hierarchy Vs Other Classes.....	204
4.11	Decidability of Formal Languages.....	205
4.12	Terminology of Problems.....	207
4.13	Important Points of Undecidability.....	207
4.14	Reduction.....	209
	<i>Student Assignments</i> .....	212





# 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 in detail 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.



# Grammars, Languages & Automata

## 1.1 Introduction

### 1.1.1 Alphabet( $\Sigma$ )

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

- $\Sigma = \{a, b\}$  is alphabet with 2 symbols  $a$  and  $b$  and hence binary alphabet.
- $\Sigma = \{0, 1\}$  is a binary alphabet.
- $\Sigma = \{0, 1, 2, \dots, 9\}$  is decimal alphabet.
- ' $\epsilon$ ' [epsilon] cannot be used as alphabet symbol because ' $\epsilon$ ' is a special symbol used to denote null string.
- Symbols in alphabet are atomic things.

**Example-1.1**

Which of the following is not an alphabet?

- |                          |                           |
|--------------------------|---------------------------|
| (a) $\{0, 1\}$           | (b) $\{ \}$               |
| (c) $\{x\}$              | (d) $\{a\}$               |
| (e) $\{\epsilon\}$       | (f) $\{0, 1, \dots, 01\}$ |
| (g) $\{a, b, c, \dots\}$ |                           |

**Solution:**

1.  $\{\epsilon\}$  is not an alphabet since  $\epsilon$  is special symbol.
2.  $\{ \}$  is not an alphabet since empty set.
3.  $\{a, b, c, \dots\}$  is not an alphabet since infinite set.
4.  $\{0, 1, 01\}$  is not an alphabet since 0 is proper subset of 01 (atomic things).

### 1.1.2 String

**Definition:** A string is any sequence of zero or more finite number of symbols over the given alphabet  $\Sigma$ .

" $abb$ " is the string over  $\Sigma = \{a, b\}$ . Every symbol is a string of length 1.

**Empty string ( $\epsilon$  or  $\lambda$ ):** 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  $\Sigma = \{a, b\}$
- $|\epsilon| = 0$ . The length of empty string is zero.
- $|a| = 1$ ,  $|ab| = 2$

#### Example-1.2

How many strings are possible with alphabets  $\{a, b\}$  of length  $n$ ?

**Solution:**

$$\text{Number of string} = 2 \times 2 \times \dots \times 2 = 2^n$$

Where each '2' represents the number of choices possible i.e. either  $a$  or  $b$ .

#### Example-1.3

How many strings possible with alphabet  $\Sigma = \{a, b, c\}$  upto length  $n$ ?

**Solution:**

Number of strings = Number of string of length 0 + Number of string of length 1 + ..... Number of string of length  $n$

$$\Rightarrow |\Sigma|^0 + |\Sigma|^1 + \dots + |\Sigma|^n$$

$$\Rightarrow |\Sigma|^0 \left[ \frac{|\Sigma|^{n+1} - 1}{|\Sigma| - 1} \right]$$

$$3^0 \left[ \frac{3^{n+1} - 1}{3 - 1} \right] = \frac{3^{n+1} - 1}{2}$$

**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" OR zero or more consecutive symbols of a string.

For string  $abb$  over  $\Sigma = \{a, b\}$ . The possible substrings are:

- Zero length substring:  $\epsilon$
- One length substrings:  $a, b$
- Two length substrings:  $ab, bb$
- Three length substrings:  $abb$

Mathematically,  $\text{substring}(w) = \{y \mid w = xyz\}$

"Proper Substring" 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:

- $\epsilon$ , (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:  $\epsilon$ ,  $b$ ,  $bb$ ,  $abb$ .

**Example-1.4**

What is the cardinality of prefix ("Education")  $\cap$  Suffix ("To\_all")?

**Solution:****Prefix of Education:**

1.  $\lambda$
2. E
3. Ed
4. Edu
5. Educ
6. Educa
7. Educat
8. Educatio
9. Educatio
10. Education

**Suffix of To\_all**

1.  $\lambda$
2. I
3. II
4. all
5. \_all
6. o\_all
7. To\_all

Hence, intersection will be  $\lambda$  only. So, cardinality = 1.

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

For string  $abb$ , the possible proper prefixes are:  $\epsilon$ ,  $a$ ,  $ab$ .

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

For string  $abb$ , the possible proper suffixes are:  $\epsilon$ ,  $b$ ,  $bb$ .

**1.1.4 Power of an Alphabet**

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

$\Sigma^0 = \{\epsilon\}$  : 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 ( $\Sigma^*$ )**

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

**1.1.6 Positive Closure ( $\Sigma^+$ )**

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



**1.17 Subwords**

Subwords are all the substrings except null string.

**Example:** Consider a string 'vexillology'

Substrings of length 1 = v, e, x, i, l, o, g, y

Substrings of length 2 = ve, ex, xi, il, lo, ol, og, gy

Substrings of length 3 = vex, exi, xil, ill, llo, lol, olo, log, ogy

**Notice** all substrings of length 3 are unique thus any substring of length  $\geq 3$  can be calculated without considering any constraint related to context.

Number of substrings length 4 = 8

Number of substrings length 5 = 7

Number of substrings length 6 = 6

Number of substrings length 7 = 5

Number of substrings length 8 = 4

Number of substrings length 9 = 3

Number of substrings length 10 = 2

Number of substrings length 11 = 1

Hence, number of subwords

$$\Rightarrow 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 9 + 8 = 62$$

**Example-1.5**

How many subwords are there of string length  $n$  in which all the letters are unique?

**Solution:**

For a string of length  $n$  in which all the letters are unique

Length 1 —  $n$

Length 2 —  $n - 1$

Length 3 —  $n - 2$

$\vdots$

Length  $n$  — 1

$$\begin{aligned}\text{Total subwords} &= 1 + 2 + 3 + \dots + n - 1 + n \\ &= \frac{n(n+1)}{2}\end{aligned}$$

**Example-1.6**

How many subwords are there for AXIOMATIZABLE?

**Solution:**

Number of subwords of length 1 = A, X, I, D, M, T, Z, B, L, E

Number of subwords of length 2 = AX, XI, ID, OM, MA, AT, TI, IZ, ZA, AB, BL, LE

**Note:** Subwords of length 2 are unique thus any subwords of length  $\geq 2$ , does not depends on context.  
Hence,

Number of subwords of length 3 = 11

Number of subwords of length 4 = 10

Number of subwords of length 5 = 9  
 Number of subwords of length 6 = 8  
 Number of subwords of length 7 = 7  
 Number of subwords of length 8 = 6  
 Number of subwords of length 9 = 5  
 Number of subwords of length 10 = 4  
 Number of subwords of length 11 = 3  
 Number of subwords of length 12 = 2  
 Number of subwords of length 13 = 1

Hence total subwords =  $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 10 = 88$

**Example -1.7**

The number of distinct subwords present in "NAMITA" is equal to

(a) 16

(b) 19

(c) 20

(d) 22

**Solution : (c)**

Subwords of 1 letter = 5 (N, A, M, I, T)  
 2 letters = 5 (NA, AM, MI, IT, TA)  
 3 letters = 4  
 4 letters = 3  
 5 letters = 2  
 6 letters = 1  
 Total =  $5 + (5 + 4 + 3 + 2 + 1)$   
 $= 5 + \frac{5(6)}{2} = 20$

**1.1.8 Language**

**Definition:** A set of strings over the given alphabet  $\Sigma$  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.

**1.1.9 Grammar**

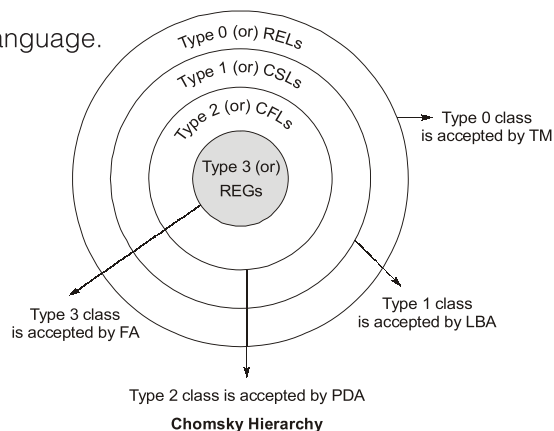
Grammar has set of rules to generate the strings of a language.

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".



- $\text{Type 3} \subseteq \text{Type 2} \subseteq \text{Type 1} \subseteq \text{Type 0}$ .
- $\text{Regular languages} \subseteq \text{CFLs} \subseteq \text{CSLs} \subseteq \text{REs}$ .

### 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*.

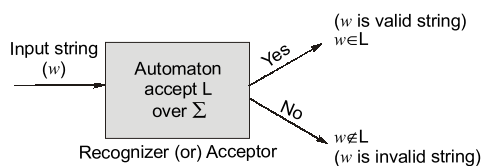
6. **Recursive Enumerable Language (REL):** A language accepted by Turing machine is called a *recursive enumerable language* or *REL*.

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  $\Sigma$ .



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

**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  $\epsilon$ -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.  
 $\therefore$  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.  
 $\therefore$  Option (c) is correct.
3. (d)  
FA can be used in pattern matching, sequential circuit design and lexical analysis of compiler design, etc.  
 $\therefore$  Option (d) is correct.
4. (c)  
A finite automata may be an acceptor or a transducer.  
 $\therefore$  Option (c) is correct.
5. (b)  
FA accepts regular languages  
 $\therefore$  Option (b) is correct.
6. (d)  
In computer, FA program can be stored using table, graph, array, etc.  
 $\therefore$  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.

**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

**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  $\epsilon$ -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.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 cannot be used in pattern recognition  
 (d) all of these
- Q.14** Regarding the power of recognizing the languages, which of the following statements is false?  
 (a) The NFA 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  $\epsilon$ -transitions then L is also accepted by an NFA without  $\epsilon$  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)  $Type0 \subseteq Type1 \subseteq Type2 \subseteq Type3$   
 (b)  $Type3 \subseteq Type1 \subseteq Type2 \subseteq Type0$   
 (c)  $Type0 \subseteq Type2 \subseteq Type1 \subseteq Type3$   
 (d) None of these
- Q.25** Which one of the following is correct for the input alphabet  $\Sigma$ ?  
 (a)  $\Sigma^* = \Sigma$  (b)  $\Sigma^* = \Sigma^+ - \{\epsilon\}$   
 (c)  $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$  (d)  $\Sigma^* = \Sigma^+ \cap \{\epsilon\}$

**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) |

**Student's  
Assignments****Explanations**

1. (a)  
Any FA having non deterministic transition capability is known as NFA.
2. (b)  
A FA having any number of input tape does not increase its power. Writing capability and two side measurement may increase.  
Hence, option (b) is correct.
4. (d)  
FA can be used for pattern recognition is compiler design in a lexical analysis phase. Although it does not have capability to write on the tape and because of finite number of states can only acts as an counter.
5. (d)  
FA can acts an acceptor, pattern matter and recognizer all.
6. (c)  
Expressive power of non-deterministic Turing Machine is same as deterministic Turing Machine.
7. (c)  
A finite automation can check the validity of input string by checking whether the FA going to final state or not.  
Moore and Mealy machine are the application of finite automata used to generate output based on some input.
8. (b)  
Finite state machine can recognize only regular language.
9. (d)  
An automation is a cognitive (acceptor) device and a grammar is generative device.
10. (d)  
Context-free language can be recognized by any device whose power is more than POA thus both linear bounded automata and POA can recognize CFL.
11. (b)  
All DFA's are NFA's. Thus, any DFA can similar NFA.
12. (d)  
Right linear grammar generates regular language any finite automation can only accept regular language.
13. (a)  
Basic limitation of FSM is that it cannot remember any arbitrary information.
14. (b)  
Non-deterministic push down automata has more power than deterministic push down automata.
15. (c)  
Expressive power of:  $DFA \equiv NFA \equiv NFA$  with  $\epsilon$ -moves and they are inter convertible and thus option (c) is false.
16. (c)  
Deterministic push down automata is used to accept deterministic context free language.
17. (a)  
Push down automata is used to accept context free languages and deterministic context free languages but CFLs are more stricter.
18. (a)  
Non deterministic push down automata is used to accept context free language but not deterministic context free language.
19. (b)  
Linear bounded automata can be used to accept CFL's, DCFL's, CSL's but if we gives answer is stricter from then CSL's.
20. (d)  
TM can accept any of the given language. But strictly RELS. Any other language come under this.