COMPUTER SCIENCE & INFORMATION TECHNOLOGY

Theory of Computation

Comprehensive Theory
with Solved Examples *and* Practice Questions

MADE EASY PUBLICATIONS

www.madeeasypublications.org

MADE EASY Publications Pvt. Ltd.

Corporate Office: 44-A/4, Kalu Sarai (Near Hauz Khas Metro Station), New Delhi-110016 | **Ph. :** 9021300500

Email : infomep@madeeasy.in | **Web :** www.madeeasypublications.org

Theory of Computation

EDITIONS

First Edition : 2015
Second Edition : 2016
Third Edition : 2017
Fourth Edition : 2018
Fifth Edition : 2019
Sixth Edition : 2020
Seventh Edition : 2021
Eighth Edition : 2022
Ninth Edition : 2023
Tenth Edition : 2024
Eleventh Edition : 2025

Twelfth Edition : 2026

CONTENTS

Theory of Computation

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

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.

■■■■

CHAPTER

1

Grammars, Languages and Automata

1.1 INTRODUCTION

1.1.1 Alphabet(Σ)

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

- Σ = {a, b} is alphabet with 2 symbols *a* and *b* and hence binary alphabet.
- Σ = {0, 1} is a binary alphabet.
- Σ = {0, 1, 2,... 9} is decimal alphabet.
- '∈' [epsilon] cannot be used as alphabet symbol because '∈' 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) {∈}
(f) {0, 1, ... 01}
(g) {*a*, *b*, *c*, ...}

Solution :

1. {∈} is not an alphabet since ∈ is special symbol.
2. { } is not an alphabet since empty set.
3. {*a*, *b*, *c*, ...} 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 Σ.

"*abb*" is the string over Σ = {*a*, *b*}. Every symbol is a string of length 1.

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.
   - $|a| = 1$, $|ab| = 2$

Example 1.2 How many strings are possible with alphabets $\{a, b\}$ of length n?

Solution :

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

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

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

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

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

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: ε
   - One length substrings: a, b
   - Two length substrings: ab, bb
   - Three length substrings: abb

   Mathematically, 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:
   - 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*.

Example 1.4 What is the cardinality of prefix ("Education") ∩ Suffix ("To_all")?

Solution :

Prefix of Education:
1. λ
2. E
3. Ed
4. Edu
5 Educ
6. Educa
7. Educat
8. Educatio
9. Educatio
10. Education

Suffix of To_all
1. λ
2. l
3. ll
4. all
5. _all
6. o_all
7. To_all

Hence, intersection will be λ 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: 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.
- $\Sigma^+ = \Sigma^* - \{\varepsilon\} = \Sigma^* - \Sigma^0$
- $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

1.17 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, ll, 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 ≥ 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

⋮

⋮

⋮

Length *n* — 1

$$\text{Total subwords} = 1 + 2 + 3 + \ldots\ldots n - 1 + n$$

$$= \frac{n(n+1)}{2}$$

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 ≥ 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 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.

1.1.9 Grammar

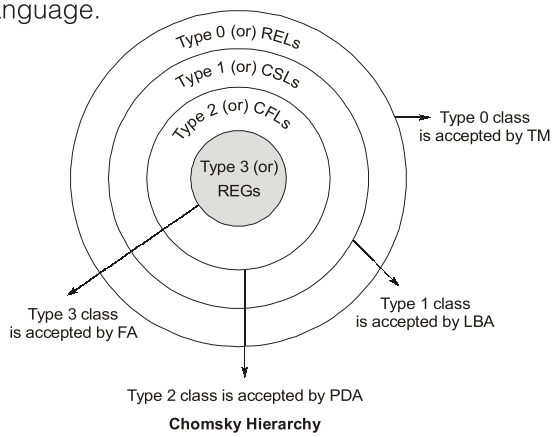
Grammar has set of rules to generate the strings of a language.

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".



Chomsky Hierarchy

- Type 3 ⊆ Type 2 ⊆ Type 1 ⊆ Type 0.
- Regular languages ⊆ CFLs ⊆ CSLs ⊆ RELs.

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

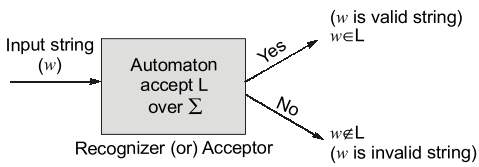
   If a language can be enumerated in a lexicographic order (Particular order) by some Turing machine, then such a language is called recursive or *REC*.
6. **Recursive Enumerable Language (REL):** A language accepted by Turing machine is called a *recursive enumerable language* or REL.

   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 ∑.



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 ∑.

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, L(NFA) is the class of languages accepted by NFA's and L(ε-NFA) is the class of languages accepted by ε-NFA i.e., all finite automata are having same expressive power.

   (d) Same expressive power means for any language if there exists DFA than there must be NFA's or ε-NFA.

   Set of language accepted by DFA is equal to set of language accepted by NFA which is also equal to ε-NFA.

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.

NOTE: 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) ≅ 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) ⊂ L(TM).

- 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 if 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