

THEORY OF COMPUTATION

Chapter 6

TURING MACHINE

Study Notes for Engineering Students

Topics Covered in this Chapter
6.1 Structure and working of a simple Turing Machine
6.2 Instantaneous description of a Turing Machine
6.3 Turing Machine as language acceptor
6.4 Universal Turing Machine

This Book Includes
Detailed theory in paragraph, list and table form • Neatly drawn diagrams • Fully solved examples with step-by-step traces • Key points for quick revision • 15 MCQs with Answer Key • 10 Broad Questions

CONTENTS

About this Chapter	3
6.1 Structure and Working of a Simple Turing Machine	4
6.1.1 Structure (Model) of a Turing Machine	4
6.1.2 Formal Definition of a Turing Machine	5
6.1.3 Working of a Turing Machine	5
6.1.4 Ways of Representing a Turing Machine	6
6.1.5 Solved Example 1: TM for $L = \{0^n1^n \mid n \geq 1\}$	6
6.1.6 Solved Example 2: TM for $L = \{a^nbn^n \mid n \geq 1\}$	8
6.1.7 Variations of the Turing Machine (Concept Only)	8
6.2 Instantaneous Description of a Turing Machine	10
6.2.1 The Move Relation $\vdash$	10
6.2.2 Types of Instantaneous Descriptions	10
6.3 Turing Machine as Language Acceptor	12
6.3.1 Recursively Enumerable and Recursive Languages	12
6.3.2 Solved Example 3: TM accepting strings over $\{a, b\}$ that end with ab	12
6.3.3 Power of Machines: The Hierarchy of Languages	13
6.4 Universal Turing Machine	15
6.4.1 Motivation	15
6.4.2 Encoding of a Turing Machine	15
6.4.3 Structure of the Universal Turing Machine	16
6.4.4 Working of the UTM	16
6.4.5 Importance and Limits	16
Quick Revision Table	18
Multiple Choice Questions (15)	19
Answer Key for MCQs	21
Broad Questions (10)	22

About this Chapter

In the previous chapter we saw that a pushdown automaton is more powerful than a finite automaton, but it still has serious limits. Its stack can be examined only at the top, and the input can be read only once from left to right. Because of this, a PDA cannot accept simple languages such as $\{a^n b^n c^n \mid n \geq 1\}$ or $\{ww \mid w \in \{a, b\}^*\}$, which need the machine to compare several parts of the input.

In 1936 the British mathematician **Alan Turing** proposed a much more powerful abstract machine, now called the **Turing Machine (TM)**. It has a finite control like an automaton, but its memory is an **unbounded tape** that the machine can read *and write*, moving its head in *both* directions. Despite its simplicity, a Turing machine can carry out any computation that a modern computer can. It is therefore the standard mathematical model of an **algorithm**, and it is the foundation of the theory of computability.

Learning Outcomes

- Describe the structure of a Turing machine and define it formally.
- Explain how a TM works, including what it means for a TM to halt.
- Write and read instantaneous descriptions and trace a computation.
- Design simple TMs as language accepters, and understand recursive and recursively enumerable languages.
- Explain the idea of a Universal Turing Machine and the encoding of a TM.

Prerequisites

Topic	What you should already know
Finite Automata	States, transitions, acceptance of a string
Pushdown Automata	Stack, instantaneous description, acceptance (Chapter 5)
Context Free Languages	Limits of CFLs, examples such as $a^n b^n c^n$ that are not context free

Notation Used in this Chapter

Symbol	Meaning
Q	Finite set of states
Σ	Input alphabet (does not contain the blank)
Γ	Tape alphabet; $\Sigma \subset \Gamma$
B	Blank symbol; $B \in \Gamma$ and $B \notin \Sigma$
δ	Transition function
q_0	Start state
F	Set of final (accepting) states
L, R	Head movement: one cell left, one cell right
$\alpha q \beta$	Instantaneous description: tape contents $\alpha\beta$, state q , head on the first symbol of β
$\vdash, \vdash^*$	One move, zero or more moves
$L(M)$	Language accepted by the machine M

6.1 Structure and Working of a Simple Turing Machine

6.1.1 Structure (Model) of a Turing Machine

A Turing machine consists of a finite control, a tape that is infinite in both directions, and a read/write head that moves over the tape (Figure 6.1).

Tape (unbounded in both directions)

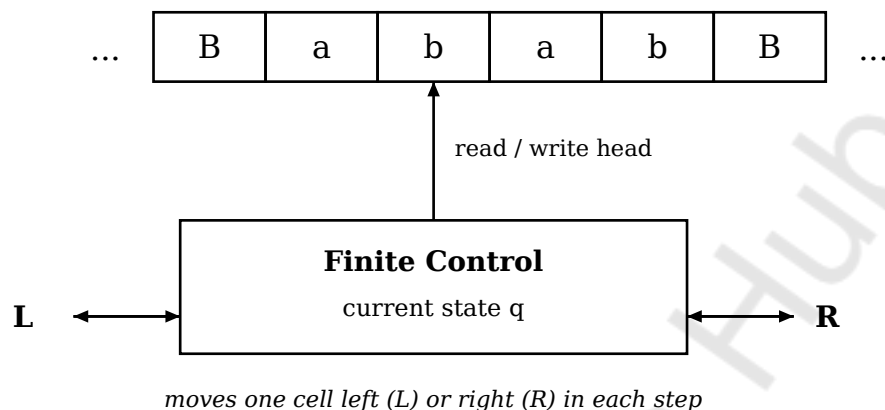


Figure 6.1: Model of a Turing Machine

- **Tape:** divided into cells, each holding one symbol. It is unbounded, so the machine never runs out of memory. All cells that do not hold input contain the blank symbol B.
- **Read/write head:** scans exactly one cell at a time. It can read the symbol, overwrite it, and move one cell to the left or to the right.
- **Finite control:** holds the current state. Based on the current state and the scanned symbol it decides what to write, in which direction to move, and which state to enter next.

Initial set-up. The input string w is written on the tape, one symbol per cell. All other cells are blank. The head is placed on the leftmost symbol of w and the machine is in the start state q_0 .

Comparison: what is new in a TM?

- The head can **write** on the tape (FA and PDA only read the input).
- The head can move in **both directions** and can re-read symbols (FA and PDA move only forward).
- The tape is **unbounded** and serves as both input and working memory.
- The machine can accept or reject **immediately**: it does not have to read the whole input.

6.1.2 Formal Definition of a Turing Machine

A (deterministic) Turing machine is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Component	Meaning	Remarks
Q	Finite set of states	Includes q_0 and the final states
Σ	Input alphabet	Symbols in the input string; $B \notin \Sigma$
Γ	Tape alphabet	$\Sigma \subset \Gamma$; contains B and any marker symbols such as X, Y
δ	Transition function	$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
$q_0 \in Q$	Start state	Computation begins here
$B \in \Gamma$	Blank symbol	Fills every cell that is not part of the input
$F \subseteq Q$	Set of final states	Reaching a final state means acceptance

Meaning of the transition function. The statement

$$\delta(q, a) = (p, b, D)$$

means: when the machine is in state q and the head scans the symbol a , it **(1)** replaces a by b , **(2)** moves the head one cell in the direction $D \in \{L, R\}$, and **(3)** enters state p .

Important points about δ

- δ is a **partial function**: for some pair (q, a) it may be undefined. Then the machine has no move and **halts**.
- In every move a TM **must** move the head left or right (no "stay" option in this basic model).
- A TM can write the same symbol that it read, which leaves the cell unchanged.
- The machine is deterministic: there is at most one move for each (state, symbol) pair.
- Final states usually have no outgoing transitions, so the machine halts as soon as it enters one.

6.1.3 Working of a Turing Machine

A TM works in discrete steps. In each step it carries out the following actions.

1. Read the symbol a in the cell under the head; let q be the current state.
2. Look up $\delta(q, a)$. If it is undefined, the machine **halts**.
3. If $\delta(q, a) = (p, b, D)$, write b in the current cell.
4. Move the head one cell to the left ($D = L$) or to the right ($D = R$).
5. Change the state to p and repeat from step 1.

Possible outcomes. For a given input exactly one of the following happens:

Outcome	Description
Halts and accepts	The machine reaches a final state (it halts there because no move is defined)
Halts and rejects	The machine reaches a non-final state with no move for the scanned symbol
Never halts (loops)	The machine keeps moving forever; the input is not accepted

In many textbooks the tape is taken to be infinite only to the right, with the input at the left end. This one-way infinite tape model has exactly the same computing power as the two-way infinite tape used here.

6.1.4 Ways of Representing a Turing Machine

Method	Description
Transition table	Rows are states, columns are tape symbols; each entry is (new state, symbol written, direction)
Transition diagram	Circles for states; an arrow from q to p labelled $a/b, D$ stands for $\delta(q, a) = (p, b, D)$; double circle for a final state
Instantaneous description	A snapshot $\alpha q \beta$ of the tape and the state (Section 6.2)

6.1.5 Solved Example 1: TM for $L = \{0^n 1^n \mid n \geq 1\}$

Idea. Repeatedly match the leftmost unmarked 0 with the leftmost unmarked 1. Replace the 0 by X and the matching 1 by Y. When no 0 is left, check that no 1 is left either. Here $\Gamma = \{0, 1, X, Y, B\}$.

1. Mark the leftmost 0 by X (state q_0 to q_1).
2. Move right over 0's and Y's to the first 1 and change it to Y (state q_1 to q_2).
3. Move left over 0's and Y's until X is seen; step right to the next unmarked symbol (state q_2 back to q_0).
4. If q_0 sees Y, then all 0's are marked; go to q_3 and verify that only Y's remain up to the blank.
5. On reaching B in q_3 , enter the final state q_4 .

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$ with the following transitions.

State	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1, 0, R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4 (final)	-	-	-	-	-

A dash (-) means that the transition is undefined, so the machine halts.

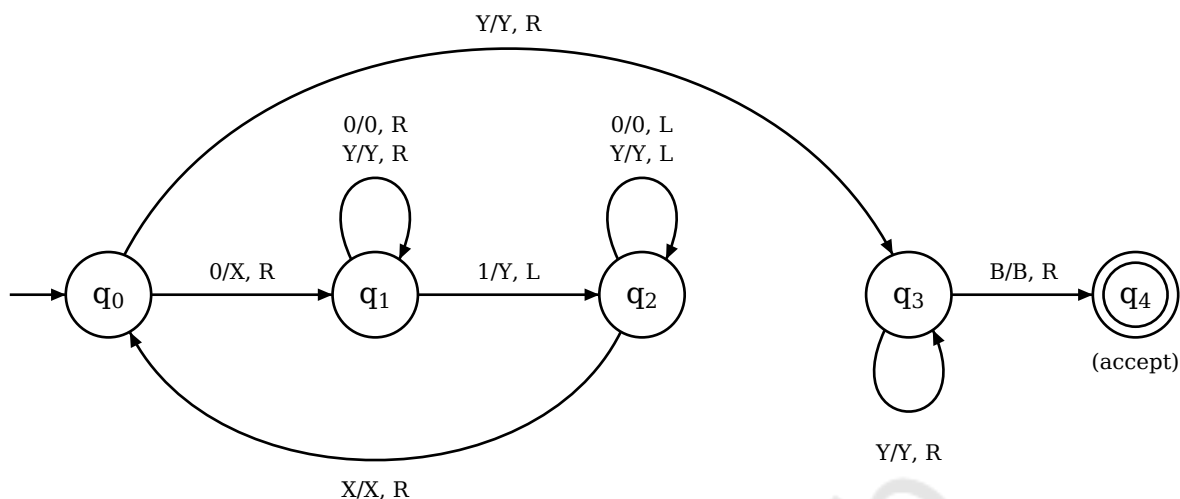


Figure 6.2: Transition diagram of the TM for $0^n 1^n$. An edge label " $a/b, D$ " means: read a , write b , move in direction D .

Computation on $w = 0011$ (accepted).

Step	Instantaneous description	Move used
1	$q_0 0011$	start
2	$X q_1 011$	$\delta(q_0, 0) = (q_1, X, R)$
3	$X0 q_1 11$	$\delta(q_1, 0) = (q_1, 0, R)$
4	$X q_2 0Y1$	$\delta(q_1, 1) = (q_2, Y, L)$
5	$q_2 X0Y1$	$\delta(q_2, 0) = (q_2, 0, L)$
6	$X q_0 0Y1$	$\delta(q_2, X) = (q_0, X, R)$
7	$XX q_1 Y1$	$\delta(q_0, 0) = (q_1, X, R)$
8	$XXY q_1 1$	$\delta(q_1, Y) = (q_1, Y, R)$
9	$XX q_2 YY$	$\delta(q_1, 1) = (q_2, Y, L)$
10	$X q_2 XYY$	$\delta(q_2, Y) = (q_2, Y, L)$
11	$XX q_0 YY$	$\delta(q_2, X) = (q_0, X, R)$
12	$XXY q_3 Y$	$\delta(q_0, Y) = (q_3, Y, R)$
13	$XXYY q_3 B$	$\delta(q_3, Y) = (q_3, Y, R)$
14	$XXYYB q_4$	$\delta(q_3, B) = (q_4, B, R)$; final state, ACCEPT

Rejected strings. For $w = 001$, after marking one pair the machine finds q_0 on the second 0 and later reaches a blank in q_1 or q_2 with no matching move, so it halts in a non-final state. For $w = 011$ the second 1 is met in state q_3 where no move for the symbol 1 exists, so the machine halts and rejects.

6.1.6 Solved Example 2: TM for $L = \{a^n b^n c^n \mid n \geq 1\}$

This language is **not context free**, so no PDA can accept it, but a Turing machine can. **Idea.** Mark one a (by X), then the first unmarked b (by Y), then the first unmarked c (by Z). Return to the left end and repeat. When all a's are marked, verify that only Y's and Z's remain. Here $\Gamma = \{a, b, c, X, Y, Z, B\}$.

$M = (\{q_0, \dots, q_5\}, \{a, b, c\}, \Gamma, \delta, q_0, B, \{q_5\})$ with the transition table:

State	a	b	c	X	Y	Z	B
q_0	(q_1, X, R)	-	-	-	(q_4, Y, R)	-	-
q_1	(q_1, a, R)	(q_2, Y, R)	-	-	(q_1, Y, R)	-	-
q_2	-	(q_2, b, R)	(q_3, Z, L)	-	-	(q_2, Z, R)	-
q_3	(q_3, a, L)	(q_3, b, L)	-	(q_0, X, R)	(q_3, Y, L)	(q_3, Z, L)	-
q_4	-	-	-	-	(q_4, Y, R)	(q_4, Z, R)	(q_5, B, R)
q_5 (final)	-	-	-	-	-	-	-

- q_0 : looks for the leftmost unmarked a; if it sees Y, all a's are done.
- q_1 : moves right to the first unmarked b. q_2 : moves right to the first unmarked c.
- q_3 : returns to the left end (stops at X, then re-enters q_0).
- q_4 : final check that only Y's and Z's remain up to the blank.

Computation on $w = abc$ (accepted).

Step	Instantaneous description	Move used
1	$q_0 abc$	start
2	$X q_1 bc$	$\delta(q_0, a) = (q_1, X, R)$
3	$XY q_2 c$	$\delta(q_1, b) = (q_2, Y, R)$
4	$X q_3 YZ$	$\delta(q_2, c) = (q_3, Z, L)$
5	$q_3 XYZ$	$\delta(q_3, Y) = (q_3, Y, L)$
6	$X q_0 YZ$	$\delta(q_3, X) = (q_0, X, R)$
7	$XY q_4 Z$	$\delta(q_0, Y) = (q_4, Y, R)$
8	$XYZ q_4 B$	$\delta(q_4, Z) = (q_4, Z, R)$
9	$XYZB q_5$	$\delta(q_4, B) = (q_5, B, R)$; ACCEPT

6.1.7 Variations of the Turing Machine (Concept Only)

Many modified versions of the basic TM have been studied. Surprisingly, none of them accepts more languages than the basic model; they only make programming easier or change the speed.

Variant	Change from the basic model	Power
Multi-tape TM	Several tapes, each with its own head	Same as the basic TM
Nondeterministic TM	δ gives a set of possible moves	Same as the basic TM
Multi-track TM	Each cell holds several symbols at once	Same as the basic TM
TM with "stay" option	Head may also stay (S) in a move	Same as the basic TM
One-way infinite tape	Tape is unbounded on the right only	Same as the basic TM

Key Points: Section 6.1

- TM = finite control + unbounded tape + read/write head that moves left or right.
- $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$; $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$.
- One move: read a , write b , move L or R, change state.
- A TM halts when no move is defined; it accepts if it halts in a final state.
- A TM can accept $a^n b^n c^n$, which is beyond the power of a PDA.

6.2 Instantaneous Description of a Turing Machine

As in a PDA, we need a short way to describe the complete state of the machine at any instant. A TM is fully described at a given moment by three things: the contents of the tape, the position of the head and the current state. Since the tape is infinite but only a finite part of it is non-blank, we write only the non-blank part (plus any blanks that the head is scanning).

Definition: Instantaneous Description (ID)

An ID of a TM is a string of the form $\alpha q \beta$, where q is the current state, $\alpha \beta$ is the non-blank portion of the tape, and the head scans the **first symbol of β** . The state q is written immediately to the left of the scanned symbol.

Part	Meaning
α	Tape contents to the left of the head
q	Current state (written just before the scanned symbol)
β	Tape contents from the scanned cell onwards; its first symbol is under the head

Example. The ID $XXY q_1 1$ means: the tape holds $XXY1$, the state is q_1 , and the head scans the symbol 1 that immediately follows q_1 . The initial ID on input w is $q_0 w$.

6.2.1 The Move Relation $\vdash$

Let $X_1X_2\dots X_{i-1} q X_i\dots X_n$ be an ID. The next ID depends on $\delta(q, X_i)$.

If $\delta(q, X_i) =$	Then	Effect
(p, Y, R)	$X_1\dots X_{i-1} q X_i\dots X_n \vdash X_1\dots X_{i-1} Y p X_{i+1}\dots X_n$	Write Y , move right, enter p
(p, Y, L)	$X_1\dots X_{i-1} q X_i\dots X_n \vdash X_1\dots X_{i-2} p X_{i-1} Y X_{i+1}\dots X_n$	Write Y , move left, enter p

- If the head moves right beyond the written part, a blank B is added on the right, so the head then scans B .
- If the head moves left from the first written cell, a blank B is added on the left in the two-way infinite tape model.
- As usual, $\vdash^*$ denotes zero or more moves: $q_0 w \vdash^* \alpha p \beta$.

6.2.2 Types of Instantaneous Descriptions

Type	Description
Initial ID	$q_0 w$, where w is the input; the head is on the leftmost symbol of w
Intermediate ID	Any ID reached after some moves
Accepting ID	$\alpha p \beta$ where $p \in F$ (the machine halts and accepts)
Halting (rejecting) ID	$\alpha q \beta$ where $q \notin F$ and δ is undefined for the symbol under the head

Worked illustration. Consider again the machine of Example 1 on $w = 0011$. Its first few IDs were $q_0 0011 \vdash X q_1 011 \vdash X_0 q_1 11 \vdash X q_2 0Y1 \vdash q_2 X_0 Y1 \vdash X q_0 0Y1$. Observe how the state symbol travels along the tape together with the head, and how the tape contents change as symbols are overwritten by X and Y.

Key Points: Section 6.2

- ID = $\alpha q \beta$: tape contents $\alpha\beta$, state q , head on the first symbol of β .
- Initial ID is $q_0 w$. Right move: $\alpha q X\beta \vdash \alpha Y p \beta$. Left move: $\alpha Z q X\beta \vdash \alpha p Z Y \beta$.
- An ID with a final state is accepting; an ID with no applicable move and a non-final state is rejecting.

Poly Notes Hub

6.3 Turing Machine as Language Acceptor

A Turing machine can be used to decide which strings belong to a language. We give it the string as input, run it, and observe whether it ends up in a final state.

Definition: Language accepted by a TM

For a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, the language accepted by M is

$$L(M) = \{ w \in \Sigma^* \mid q_0 w \vdash^* \alpha p \beta \text{ for some } p \in F \text{ and some } \alpha, \beta \in \Gamma^* \}$$

So a string w belongs to $L(M)$ exactly when, started on w , the machine reaches a final state. If the machine halts in a non-final state, or never halts, the string is not accepted.

6.3.1 Recursively Enumerable and Recursive Languages

Basis	Recursively Enumerable (RE) language	Recursive language
Machine	Accepted by some TM	Accepted by some TM that halts on every input
If $w \in L$	TM halts and accepts	TM halts and accepts
If $w \notin L$	TM may halt and reject, or loop forever	TM always halts and rejects
Also called	Turing-acceptable	Turing-decidable (decidable)
Relation	Every recursive language is RE	A proper subset of RE

The machine that always halts is called a **decider** (or an algorithm for the language). A language is recursive exactly when such a decider exists. Two useful facts, stated without proof:

- The complement of a recursive language is recursive.
- If both a language L and its complement are RE, then L is recursive.

A machine that loops: let M have only $\delta(q_0, a) = (q_0, a, R)$ and $\delta(q_0, B) = (q_0, B, R)$. On every input it moves right forever, so it never halts. Here $L(M) = \emptyset$, and M is not a decider.

6.3.2 Solved Example 3: TM accepting strings over $\{a, b\}$ that end with ab

Idea. Move right to the end of the string, step back and check that the last two symbols are b and a . $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, \{q_3\})$.

State	a	b	B
q_0	(q_0, a, R)	(q_0, b, R)	(q_1, B, L)
q_1	-	(q_2, b, L)	-
q_2	(q_3, a, R)	-	-
q_3 (final)	-	-	-

Computation on $w = bab$ (accepted).

Step	Instantaneous description	Move used
1	$q_0 bab$	start
2	$b q_0 ab$	$\delta(q_0, b) = (q_0, b, R)$
3	$ba q_0 b$	$\delta(q_0, a) = (q_0, a, R)$
4	$bab q_0 B$	$\delta(q_0, b) = (q_0, b, R)$
5	$ba q_1 b$	$\delta(q_0, B) = (q_1, B, L)$
6	$b q_2 ab$	$\delta(q_1, b) = (q_2, b, L)$
7	$ba q_3 b$	$\delta(q_2, a) = (q_3, a, R)$; final state, ACCEPT

Rejected: $w = ba$. The machine reaches q_1 scanning a . There is no move for (q_1, a) , so it halts in the non-final state q_1 and the string is rejected.

6.3.3 Power of Machines: The Hierarchy of Languages

Language class	Accepted by	Example
Regular	Finite automaton	a^*b^*
Context free	Pushdown automaton	$a^n b^n, ww^R$
Context sensitive	Linear bounded automaton	$a^n b^n c^n$
Recursive	TM that always halts	Every context sensitive language
Recursively enumerable	Turing machine	The language of the Universal TM (Section 6.4)

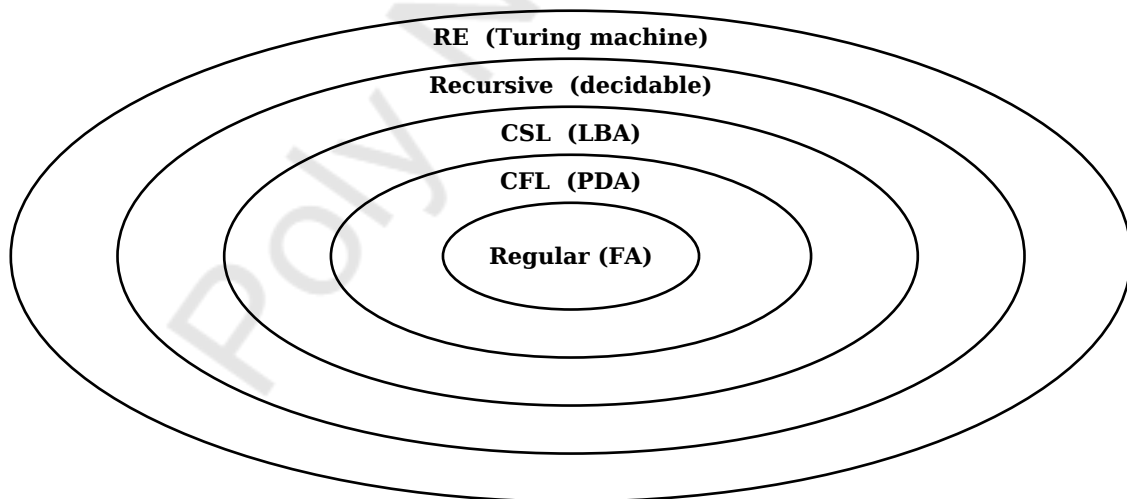


Figure 6.3: Hierarchy of languages: $\text{Regular} \subset \text{CFL} \subset \text{CSL} \subset \text{Recursive} \subset \text{RE}$

Church-Turing Thesis

Every function that can be computed by an effective procedure (an algorithm) can be computed by a Turing machine.

This is a **thesis**, not a theorem, because "effective procedure" is an informal idea and cannot be proved mathematically. It is accepted because every known model of computation (lambda calculus, recursive functions, real computers) turned out to be equivalent to the Turing machine.

Key Points: Section 6.3

- $L(M) = \{w \mid q_0 w \vdash^* \alpha p \beta, p \in F\}$.
- A TM can accept, reject (halt in a non-final state) or loop forever.
- RE: accepted by some TM. Recursive: accepted by a TM that halts on all inputs.
- Regular $\subset$ CFL $\subset$ CSL $\subset$ Recursive $\subset$ RE.
- Church-Turing thesis: algorithms = Turing machines.

6.4 Universal Turing Machine

6.4.1 Motivation

Each Turing machine we have designed so far solves exactly one problem, like a calculator that has one fixed program built in. Real computers are different: one machine can run any program, because the program is stored in memory as data. Turing showed that the same is possible for TMs. A **Universal Turing Machine (UTM)** is a single TM that can simulate every other TM. It is the theoretical origin of the **stored-program computer**.

Definition: Universal Turing Machine

A Universal Turing Machine U takes as input the **encoding of a Turing machine M together with an input string w** , written $\langle M, w \rangle$, and simulates M on w . U accepts $\langle M, w \rangle$ if and only if M accepts w , and U halts exactly when M halts.

Its language is $L_U = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$.

6.4.2 Encoding of a Turing Machine

To give a machine as input to another machine, it must be written as a string over a fixed alphabet. The standard method uses only 0 and 1. Number the states $q_1, q_2, \dots$ (q_1 is the start state), the tape symbols $X_1, X_2, \dots$ and the directions $D_1 = L, D_2 = R$. Each is encoded by a block of 0's:

Item	Encoding	Example
State q_i	0^i (i zeros)	q_3 is 000
Tape symbol X_j	0^j	X_1 (the symbol 0) is 0, X_2 (the symbol 1) is 00, X_3 (blank) is 000
Direction	$L = 0, R = 00$	R is 00
One transition $\delta(q_i, X_j) = (q_k, X_l, D_m)$	$0^i 1 0^j 1 0^k 1 0^l 1 0^m$	see below
Whole machine	111 code1 11 code2 11 ... coden 111	transitions separated by 11

Example. Let the transition be $\delta(q_1, 1) = (q_3, 0, R)$. With $q_1 = 0$, symbol 1 = $X_2 = 00$, $q_3 = 000$, symbol 0 = $X_1 = 0$ and $R = 00$, the code is

$$0 \ 1 \ 00 \ 1 \ 000 \ 1 \ 0 \ 1 \ 00 = \mathbf{0100100010100}$$

The code of the complete machine M , followed by the separator 111 and the input w , forms the input of the UTM. Since every TM has such a binary code, the set of all Turing machines can be listed.

6.4.3 Structure of the Universal Turing Machine

For ease of understanding a UTM is described with three tapes (a multi-tape machine can always be converted into an ordinary single-tape machine).

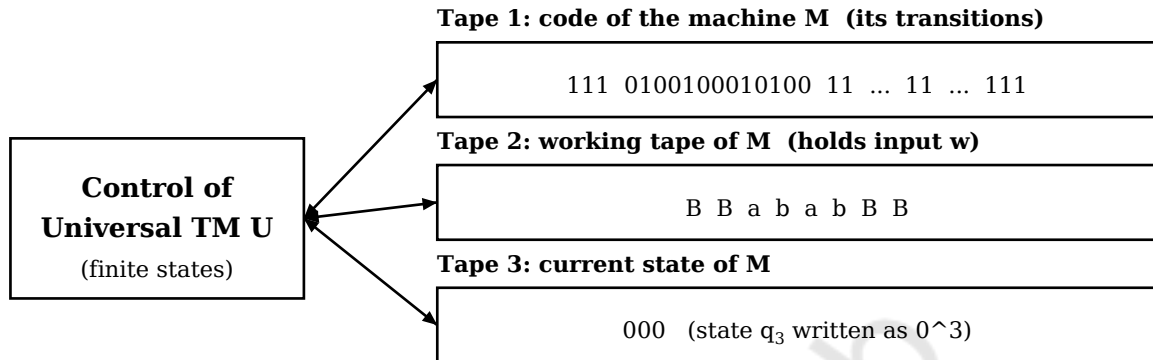


Figure 6.4: Three-tape model of a Universal Turing Machine

Tape	Contents	Purpose
Tape 1	Binary code of M (all its transitions)	The "program"; read-only lookup table
Tape 2	Input w, later the current tape of M	Simulates the tape of M
Tape 3	Current state of M, as 0^i	Remembers the state of M

6.4.4 Working of the UTM

1. **Check:** verify that tape 1 holds a valid code of a Turing machine; otherwise halt and reject.
2. **Initialise:** copy w to tape 2 and write 0 (the start state q_1) on tape 3; place the heads on the first symbols.
3. **Search:** read the scanned symbol of tape 2 and the state on tape 3. Look on tape 1 for the transition that begins with this state and symbol.
4. **Simulate:** if found, change the symbol on tape 2, move its head in the stated direction and replace the state on tape 3 by the new state.
5. **Halt:** if no transition is found, M halts, so U halts. If the state on tape 3 is a final state of M, U accepts; otherwise it rejects.
6. Otherwise go back to step 3 and repeat.

6.4.5 Importance and Limits

Ordinary TM	Universal TM
Solves one particular problem	Can simulate any TM, so it can run any algorithm
Program is built into the states	Program is given as data (stored-program idea)
Like a special-purpose calculator	Like a general-purpose computer
Input is just the string w	Input is $\langle M, w \rangle$, the code of M and the string w

- The UTM is the model behind the modern stored-program computer and behind interpreters and virtual machines.
- **(Concept only)** L_U is recursively enumerable (the UTM accepts it) but it is **not recursive**: no TM can decide, for every M and w , whether M accepts w . A related result is the **halting problem**: there is no algorithm that tells, for every M and w , whether M halts on w .

Key Points: Section 6.4

- UTM simulates any TM M on any input w ; its input is the code of M followed by w .
- Encoding: $q_i = 0^i$; symbols and directions by blocks of 0's; fields separated by 1, transitions by 11, machine enclosed in 111.
- Three-tape model: code of M , working tape, current state.
- UTM = stored-program computer. L_U is RE but not recursive.

Quick Revision Table

Concept	Definition / Formula
Turing machine	$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$
Transition function	$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
Move	$\delta(q, a) = (p, b, D)$: write b, move D, go to p
Instantaneous description	$\alpha q \beta$; head on the first symbol of β ; initial ID $q_0 w$
Language accepted	$L(M) = \{w \mid q_0 w \vdash^* \alpha p \beta, p \in F\}$
Halting	No move is defined for the current state and symbol
Recursive language	Accepted by a TM that halts on every input
RE language	Accepted by a TM that may loop on strings outside the language
Universal TM	Simulates any TM M on w; input is the code of M followed by w
Encoding of a transition	$0^i 1 0^j 1 0^k 1 0^l 1 0^m$ for $\delta(q_i, X_j) = (q_k, X_l, D_m)$
Hierarchy	Regular $\subset$ CFL $\subset$ CSL $\subset$ Recursive $\subset$ RE

Tips for Examinations

- Always write the 7-tuple, and give the transitions as a table or a diagram.
- Explain the idea in two or three lines before the transitions (mark, move, return, check).
- Show the computation as IDs of the form $\alpha q \beta$ and name the transition used at each step.
- Draw the transition diagram with labels in the form a/b, D.
- For the UTM, draw the three-tape diagram and write the encoding rules with one small example.

Multiple Choice Questions (15)

Choose the correct option. The answer key is given after the questions.

Q1. Who introduced the Turing machine in 1936?

- (a) Noam Chomsky
- (b) Alan Turing
- (c) Stephen Kleene
- (d) John Backus

Q2. A Turing machine is formally defined as a tuple of how many components?

- (a) 5
- (b) 6
- (c) 8
- (d) 7

Q3. The transition function of a deterministic Turing machine has the form

- (a) $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- (b) $\delta : Q \times \Sigma \rightarrow Q$
- (c) $\delta : Q \times \Gamma \rightarrow \text{power set of } (Q \times \Gamma \times \{L, R\})$
- (d) $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$

Q4. The move $\delta(q, a) = (p, b, R)$ means that the machine

- (a) reads b, writes a and moves left
- (b) reads a and pops b from a stack
- (c) reads a, writes b, moves right and enters state p
- (d) reads b and only moves right

Q5. The tape of a Turing machine is

- (a) finite in length
- (b) used like a stack
- (c) unbounded in length
- (d) read-only

Q6. In the instantaneous description $\alpha q \beta$, the head scans

- (a) the last symbol of α
- (b) the state q
- (c) a blank cell
- (d) the first symbol of β

Q7. A Turing machine halts when

- (a) no move is defined for the current state and scanned symbol
- (b) it reads a blank
- (c) the tape ends
- (d) the stack becomes empty

Q8. The class of languages accepted by Turing machines is called

- (a) context free languages
- (b) regular languages
- (c) recursively enumerable languages
- (d) context sensitive languages

- Q9.** A language accepted by a Turing machine that halts on every input is called
- (a) context free
 - (b) regular
 - (c) not recursively enumerable
 - (d) recursive
- Q10.** Which of the following languages can be accepted by a TM but **not** by any PDA?
- (a) $\{a^n b^n c^n \mid n \geq 1\}$
 - (b) $\{ww^R\}$
 - (c) $\{wcw^R\}$
 - (d) $\{a^n b^n \mid n \geq 1\}$
- Q11.** Compared with a finite automaton, the main new ability of a TM is that it can
- (a) read the input only from left to right
 - (b) have a finite set of states
 - (c) write on the tape and move its head in both directions
 - (d) be nondeterministic
- Q12.** The input of a Universal Turing Machine is
- (a) only the string w
 - (b) only the encoding of M
 - (c) a CFG and a string
 - (d) the encoding of a TM M together with a string w
- Q13.** A Universal Turing Machine is analogous to
- (a) a special-purpose calculator
 - (b) a stored-program general-purpose computer
 - (c) a deterministic finite automaton
 - (d) a regular expression
- Q14.** The Church-Turing thesis states that
- (a) every language is recursive
 - (b) every algorithmically computable function can be computed by a Turing machine
 - (c) a PDA is as powerful as a TM
 - (d) every Turing machine halts
- Q15.** Which of the following relations is correct?
- (a) Recursive $\subset$ CFL
 - (b) RE $\subset$ Recursive
 - (c) CFL $\subset$ Recursive $\subset$ RE
 - (d) Regular $\supset$ CFL

Answer Key for MCQs

Q.	Answer	Reason
1	(b)	Alan Turing proposed the machine in 1936.
2	(d)	$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ has seven components.
3	(a)	Read a symbol of Γ , write a symbol of Γ and move L or R.
4	(c)	Read a, write b, move right, go to p.
5	(c)	The tape is unbounded, so memory is not limited.
6	(d)	The scanned symbol is the first symbol of β , just right of q.
7	(a)	The machine stops when δ is undefined for the pair (state, symbol).
8	(c)	TMs accept exactly the recursively enumerable languages.
9	(d)	A decider (always halting TM) accepts a recursive language.
10	(a)	$a^n b^n c^n$ is not context free but a TM accepts it.
11	(c)	A TM writes on its tape and moves in both directions.
12	(d)	The UTM receives the code of M and the input w.
13	(b)	A UTM runs any program given as data, like a modern computer.
14	(b)	Algorithms correspond to Turing machines.
15	(c)	$\text{Regular} \subset \text{CFL} \subset \text{CSL} \subset \text{Recursive} \subset \text{RE}$.

Broad Questions (10)

- Q1.** Define a Turing machine formally. Explain each component of the 7-tuple and draw the model of a Turing machine.
- Q2.** Explain the working of a Turing machine. Compare a finite automaton, a pushdown automaton and a Turing machine in tabular form.
- Q3.** Design a Turing machine that accepts $L = \{0^n 1^n \mid n \geq 1\}$. Give the transition table and diagram and trace the computation on 000111.
- Q4.** Design a Turing machine for $L = \{a^n b^n c^n \mid n \geq 1\}$. Explain the idea and trace the acceptance of the string aabbcc.
- Q5.** What is an instantaneous description of a Turing machine? Explain the move relation for left and right moves with examples.
- Q6.** Define the language accepted by a Turing machine. Distinguish between recursive and recursively enumerable languages. What are the three possible outcomes of a TM on an input?
- Q7.** Design a Turing machine that accepts all strings over $\{a, b\}$ that end with ab. Show the IDs for the strings abab and aba.
- Q8.** State and explain the Church-Turing thesis. List any four variants of the Turing machine and state their computing power.
- Q9.** What is a Universal Turing Machine? Explain how a Turing machine is encoded as a binary string with an example, and describe the working of the UTM with a neat diagram.
- Q10.** Write short notes on: (a) halting of a Turing machine, (b) hierarchy of languages and machines, (c) halting problem (concept only).

— End of Chapter 6 —