UNIT V: Turing Machine

Objective:

To understand and design Turing Machines for the given recursively enumerable languages.

Syllabus:

Model, design of TM, types of Turing machines (Proofs not required), recursively enumerable languages and recursive languages. Computability Theory: Decidability of problems, undecidability of posts correspondence problem

Learning Outcomes:

Students will be able to:

- understand turing machine and its model.
- design Turing Machine's for Recursively Enumerable languages.
- define decidability and undecidability of problems.

Learning Material

5.1 Turing Machine:

A Turing machine (TM) is denoted by

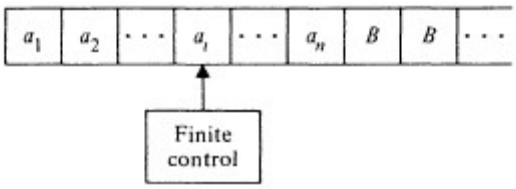
$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F),$$

where

Q is the finite set of *states*,
Γ is the finite set of allowable *tape symbols*,
B, a symbol of Γ, is the *blank*,
Σ, a subset of Γ not including B, is the set of *input symbols*,
δ is the *next move function*, a mapping from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$ (δ may, however, be undefined for some arguments),
q_0 in Q is the *start state*,
$F \subseteq Q$ is the set of *final states*.

5.2 The Turing Machine Model:

- The basic model has a finite control, an input tape that is divided into cells, and a tape head that scans one cell of the tape at a time.
- The tape has a leftmost cell but is infinite to the right. Each cell of the tape may hold exactly one of a finite number of tape symbols.
- Initially, the n leftmost cells, for some finite $n \geq 0$, hold the input, which is a string of symbols chosen from a subset of the tape symbols called the input symbols.
- The remaining infinity of cells each hold the blank, which is a special tape symbol that is not an input symbol.



Moves of Turning Machine

In one move the Turing machine, depending upon the symbol scanned by the tape head and the state of the finite control,
1) changes state,
2) prints a symbol on the tape cell scanned, replacing what was written there, and
3) moves its head left or right one cell.

Note : The difference between a Turing machine and a two-way finite automaton lies in the former's ability to change symbols on its tape.

Instantaneous description (ID):

- Instantaneous description of the Turing machine M is denoted by α1qα2.
- Here q, the current state of M, is in Q; α1α2 is the string in Γ* that is the contents of the tape up to the rightmost nonblank symbol or the symbol to the left of the head, whichever is rightmost. (Observe that the blank B may occur in α1α2.).
- The tape head is assumed to be scanning the leftmost symbol of α2, or if α2 = **ε**, the head is scanning a blank.

Acceptance by Turning Machine

The language accepted by M, denoted L(M), is the set of those words in ∑* that cause M to enter a final state when placed, justified at the left, on the tape of M, , with M in state q0, and the tape head of M at the leftmost cell.
Formally, the language accepted by M = (Q,∑,Γ,δ,q0,B,F) is

$$\{w \mid w \text{ in } \Sigma^* \text{ and } q_0 w \vdash^* \alpha_1 p \alpha_2 \text{ for some } p \text{ in } F, \text{ and } \alpha_1 \text{ and } \alpha_2 \text{ in } \Gamma^*\}.$$

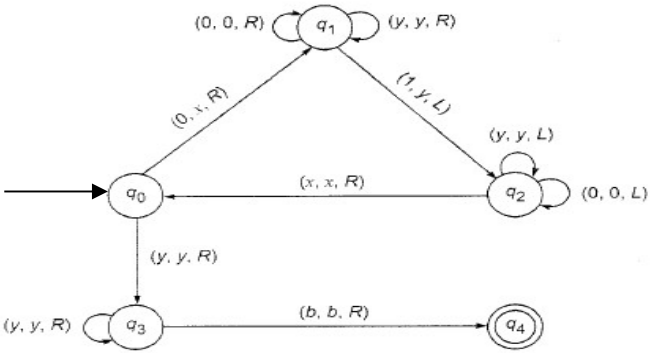
Example:

Design a TM to accept the language L = $\{0^n1^n \mid n \geq 1\}$.

Initially, the tape of M contains 0^n1^n followed by infinity of blanks.
Repeatedly, M replaces the leftmost 0 by X, moves right to the leftmost 1, replacing it by Y,, moves left to find the rightmost X, then moves one cell right to the leftmost 0 and repeats the cycle.
If, however, when searching for a 1, M finds a blank instead, , then M halts without accepting.
If, after changing a 1 to a Y, M finds no more 0's, then M checks that no more 1's remain, accepting if there are none.

State	Symbol 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	—	—	—	—	—

The function δ



Transition Diagram

String Verfication by Turning Machine

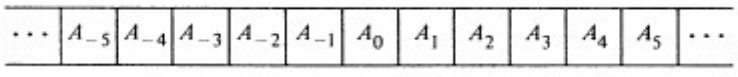
$$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 0Y1 \vdash$$
$$q_2 X0Y1 \vdash Xq_0 0Y1 \vdash XXq_1 Y1 \vdash XXYq_1 1 \vdash$$
$$XXq_2 YY \vdash Xq_2 XYY \vdash XXq_0 YY \vdash XXYq_3 Y \vdash$$
$$XXYYq_3 \vdash XXYYBq_4$$

A computation of M

5.3 Types of Turing Machines:

i) Two-way infinite tape:

A Turing machine with a two-way infinite tape is denoted by M = (Q,∑,Γ,δ,q0,B,F). As its name implies, the tape is infinite to the left as well as to the right. We denote an ID of such a device as for the one-way infinite TM. We imagine, however, that there is an infinity of blank cells both to the left and right of the current nonblank portion of the tape.

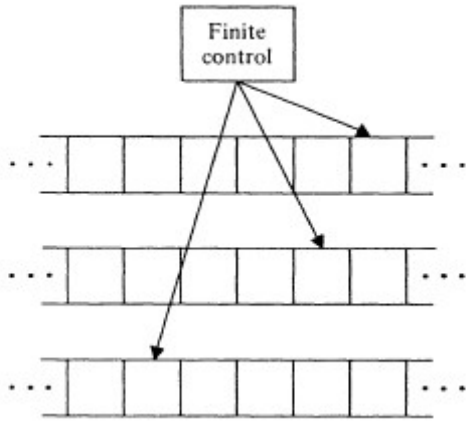


ii) Multitape Turing machines:

A multitape Turing machine consists of a finite control with k tape heads and k tapes; each tape is infinite in both directions.. On a single move, depending on the state of the finite control and the symbol scanned by each of the tape heads, the machine can:

1) change state;
2) print a new symbol on each of the cells scanned by its tape heads;
3) move each of its tape heads,, independently, one cell to the left or right, or keep it stationary.

Initially, the input appears on the first tape, and the other tapes are blank.



iii) Nondeterministic Turing machines:

A nondeterministic Turing machine is a device with a finite control and a single, one-way infinite tape. For a given state and tape symbol scanned by the tape head, the machine has a finite number of choices for the next move. Each choice consists of a new state, a tape symbol to print, and a direction of head motion. Note that the nondeterministic TM is not permitted to make a move in which the next state is selected from one choice, and the symbol printed and/or direction of head motion are selected from other choices. The nondeterministic TM accepts its input if any sequence of choices of moves leads to an accepting state.

iv) Multidimensional Turing machines:

The device has the usual finite control, but the tape consists of a k-dimensional array of cells infinite in all 2k directions, for some fixed k. Depending on the state and symbol scanned, the device changes state, prints a new symbol, and moves its tape head in one of 2k directions, either positively or negatively, along one of the k axes. Initially, the input is along one axis, and the head is at the left end of the input. At any time, only a finite number of rows in any dimension contain nonblank symbols, and these rows each have only a finite number of nonblank symbols.

B	B	B	a_1	B	B	B
B	B	a_2	a_3	a_4	a_5	B
a_6	a_7	a_8	a_9	B	a_{10}	B
B	a_{11}	a_{12}	a_{13}	B	a_{14}	a_{15}
B	B	a_{16}	a_{17}	B	B	B

v) Multihead Turing machines:

A k-head Turing machine has some fixed number,,k, of heads. The heads are numbered 1 through k,, and a move of the TM depends on the state and on the symbol scanned by each head.. In one move, the heads may each move independently left,, right, or remain stationary.

vi) Off-line Turing machines:

An off-line Turing machine is a multitape TM whose input tape is read-only. Usually we surround the input by endmarkers, $\not\subset$ on the left and \$ on the right. The Turing machine is not allowed to move the input tape head off the region between $\not\subset$ and \$.

5.4 Recursive function: a function which calls itself directly or indirectly and terminates after finite number of steps.

Total recursive function

- A function is called total recursive function if it is defined for all its arguments.
- Let f(a1,a2.....,a) be a function and defined on function g(b1,b2,....,bm), then f is total function if every element of f is assigned to some unique element of function g.
- From the definition it is clear that total recursive function is the subset of partial recursive function.
- All those partial functions for which TM halts are called total recursive functions.

Partial recursive function

- A function is called partial recursive function if it is defined for some of its arguments.
- Let f(a1,a2.....,a) be a function and defined on function g(b1,b2,....,bm), then f is partial function if some elements of f is assigned to almost one element of function g.
- Partial recursive function are turing computable.It means that there exist a turing machine for every partial recursive function.

5.5 Recursively enumerable languages

A language that is accepted by a Turing machine is said to be recursively enumerable (r.e.).

- Recursively enumerable languages are equivalent to the class of partial recursive functions.

Recursive Language:
A subclass of the r.e. sets, called the recursive sets, which are those languages accepted by at least one Turing machine that halts on all inputs.

Decidable and undecidable problems:

- A problem whose language is recursive is said to be decidable.
- A problem is undecidable if there is no algorithm that takes as input an instance of the problem and determines whether the answer to that instance is "yes" or "no."

5.6 Post's Correspondence Problem:
An instance of Post's Correspondence Problem (PCP) consists of two lists, A = w1,, ..., wk and B = x1, ... , xk, of strings over some alphabet $\sum$. This instance of PCP has a solution if there is any sequence of integers i1, i2, .. ., im, with m $\geq$1, such that ,wi1, wi2,…, wim = xi1, xi2,… xim
.
The sequence i1,…,im is a solution to this instance of PCP.

Example 1:
Let $\sum$ = {0, 1}. Let A and B be lists of three strings each, as defined

	List A	List B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

In this case PCP has a solution. Let m = 4, i1 = 2,i2 =1, i3 = 1, and i4 = 3. Then W2W1W1W3 = X2X1X1X3 = 101111110.

Example 2: Show that PCP problem with 2 lists
X=(b,bab^3,ba) and y=(b^3,ba,a) has a solution.

Given lists are x=(b,bab^3,ba) y=(b^3,ba,a)

The instances of PCP is as follows

	List X	List Y
i	X_i	Y_i
1	a	b^3
2	bab^3	ba
3	ba	a

In this case PCP is as follows
X2x1x1x3=y2y1y1y3=bab^{3}bbba
The solution sequence is 2113 PCP has a solution.

Example 3: Prove that PCP with two lists X =(01,1,1) Y=(0101,10,11) has no solution.

sol) Instance of PCP is given as

	List X	List Y
i	X_i	Y_i
1	01	0101
2	1	10

3	1	11

Where X1=01 Y1=0101

X2=1 Y2=10

X3=1 Y3=11

For any i $|X_i| < |Y_i|$The last Y is having strings of greater lengths. So to get same string for same sequences of x_1,x_2,x_3 and y_1,y_2,y_3 is difficult.

We cannot get solution sequence. Therefore the given PCP is having no solution.