CHAPTER

TURING MACHINE

INSIDE THIS CHAPTER

10.1. Introduction 10.2. Turing Machine as Physical Computing Device 10.3. Formal Definition of Turing Machine 10.4. Instantaneous Description 10.5. Transition Diagram 10.6. Turing Thesis 10.7. Turing Machine for Computing Functions 10.8. Turing Machine as Language Acceptors 10.9. Combining Turing Machine

0.1. INTRODUCTION

We have seen that neither deterministic finite automata nor pushdown automata n be regarded as truly general models for computers, since they are not capable of cognising even such simple languages as $\{a^n b^n c^n / n >= 0\}$. Turing machine is the next odel of computation under machine based approach and context-sensitive or phrase-ructure grammar in the corresponding model under grammatical approach.

In the 1930's (before advent of digital computers) several mathematicians began think about what it means to be able to compute a function. Alenzo-Church and Alan ring independently arrived at equivalent conclusion. As we might phrase their common finition now. **"A function is computable if it can be computed by a Turing lachine".**

TIPS

A Turing machine is a very simple machine, but logically speaking, has all the power of any digital computer. The Turing Machine (TM) is an abstract model created by Alan Turing (1912-1954), who was mathematician by trade and the inspiration behind the computers as we know them today.

The impact of the turing machine and the theories around them were revolutionary his time. Turing machine were used to show significant results since the foundation f mathematics and are also highly relevant to computer science. Turing introduced uring Machine to demonstrate the indecision of the halting problem.

0.2. TURING MACHINE AS PHYSICAL COMPUTING DEVICE

In order to help our understanding of the subject-matter of TMs, we can visualize TM as a physical computing device that can be represented as a diagram as shown in g. 10.1 below :

Scanned with CamScanner

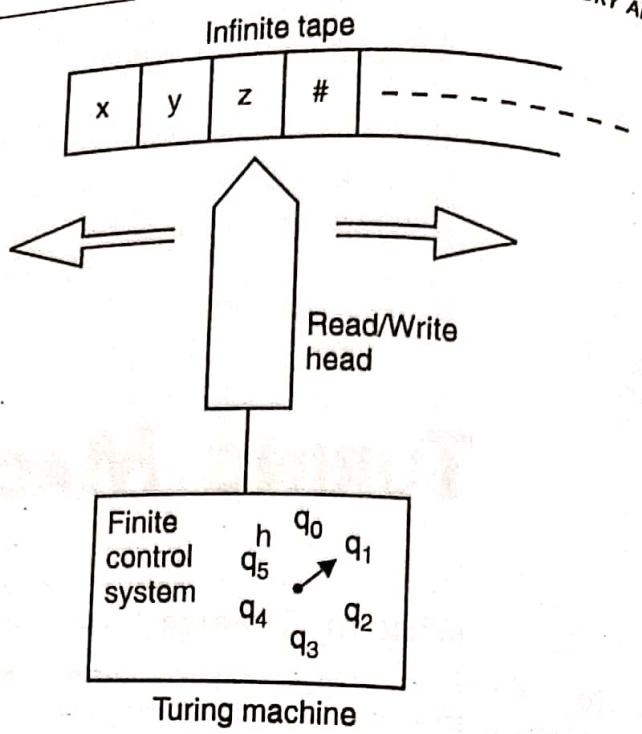


Fig. 10.1.

According to this view of TM, it consists of

(*i*) *A Tape,* with an end on the left but infinite on the right side. The tape is divided into squares or cells, with each cell capable of holding one of the tape symbols *including the blank symbol #.* At any time, there can be only finitely many cells of the tape that can contain non-blank symbols. The set of tape symbols is denoted by Γ.

As the very first step in the sequence of operations of a TM, *the input, as a finite sequence of input symbols is placed in the left-most cells of the tape. The set of the* ***input symbols*** **denoted by Σ, does not contain the blank symbol #. However, during operation of a TM, a cell may contain a *tape symbol* which is not necessarily an input symbol.**

(*ii*) *A Finite Control,* which can be in any one of the finite number of states. The states in TM can be divided in three categories that is :

(*a*) The initial state, the state of the control just at the time when TM starts its operations. The initial state is denoted by q_0.

(*b*) The *Halt state,* which is the state in which TM stops all further operations. The halt sate is *generally denoted by 'h'*. The halt state is distinct from the initial state. Thus a *TM has atleast two states.*

(*c*) Other intermediate states.

(*iii*) A tape head is always stationed at one of the tape cells and provides communication for interaction between the tape and the finite control. The head can read or scan the symbol in the cell under it. The symbol is communicated to the finite control. The control taking into consideration the symbol and its current state decides for further course of action including :

(*a*) The change of the symbol in the cell being scanned and/or

(*b*) Change of its state and/or

(*c*) Moving the head to the left or to the right. The control may decide not to move the head.

Scanned with CamScanner

(iv) The course of action is called a move of the Turing Machine. In other words, the move is a function of current state of the control and the tape symbol being scanned. The change of symbol in the cell being scanned is called writing of the cell by the head. Initially, *the head scans the right-most cell of the tape.*

10.3. FORMAL DEFINITION OF TURING MACHINE

DEFINITION

A Turing machine is a sixtape of the form

$$TM = (Q, \Sigma, \Gamma, \delta, q_0, h)$$

where (i) Q is the finite set of states.

(ii) Σ is the finite set of non-blank information symbols,

(iii) Γ is the set of tape symbols, including the blank symbol #.

(iv) δ is the next-move partial function from

$$(Q \times \Gamma) \longrightarrow (Q \times \Gamma \times \{L, R, N\})$$

where $\Gamma = \Sigma \cup \#$, 'L' denotes the tape Head moves to the left adjacent cell, 'R' denotes tape Head moves to the right adjacent cell and 'N' denotes Head does not move, that is continuous scanning the same cell.

(v) $q_0 \in Q$, is the initial state.

(vi) $h \in Q$ is the 'Halt State', in which the machine stops any further activity.

Let us discuss some move of the turing machine;

(i) if $q \in Q$, $c \in \Sigma$ and $\delta(q, c) = (q_1, b, R)$, then Turing machine (Tm) when in state q and currently scanning symbol c, will enter in state q_1 and move towards right adjacent cell after replacing 'c' with 'b'.

(ii) $\delta(q_1, a) = (q_2, b, L)$ means that at present Tm is in state q_1 and scanning the symbol a, will enter in state q_2 and write 'b' in place of 'a' and move towards left adjacent cell.

(iii) $\delta(q_1, x) = (q_2, y, N)$ means that at present Tm is in state q_1, scanning the symbol x, will enter state q_2 and write 'y' in the place of 'x' and remain stick with current position that it does't move towards left or right.

Let us see some example for the better understanding.

Example 10.1. Consider the Turing Machine $(Q, \Sigma, \Gamma, \delta, q_0, h)$ defined below that erases all non-blank symbols on the tape, where the sequence of non-blank symbols does not contain any blank symbol # in-between :

$$Q = \{q_0, h\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{a, b, \#\}$$

and the next move function is defined as follows :

q : state	σ : input symbol	$\delta(q, \sigma)$
q_0	a	$\{q_0, \#, L\}$
q_0	b	$\{q_0, \#, L\}$
q_0	#	$\{h, \#, N\}$
h	#	ACCEPT

Scanned with CamScanner

Here it is assumed that tape head is scanning right most non-blank symbol first, for example *#ababba#*.

10.4. INSTANTANEOUS DESCRIPTION

There are following differences in Tape Head of FA/PDA and Turing machine :

(*i*) The cells of the tape of an FA or a PDA are only read/scanned but are never changed/*written* into, whereas the cells of the tape of TM may be written also.

(*ii*) The tape head of an FA or a PDA always moves from left to right. However, the tape head of TM can move in both the directions.

As a consequence of facts mentioned in (*i*) and (*ii*) above, we conclude that in the case of FA and PDA the *information in the tape cells already scanned do not play any role in deciding future* moves of the automation, but in the case of a TM, the information contents of all the cells, *including the ones earlier scanned* also play a role in deciding future moves. This leads to the slightly different definitions of configuration or Instantaneous Description (ID) in the case of a TM.

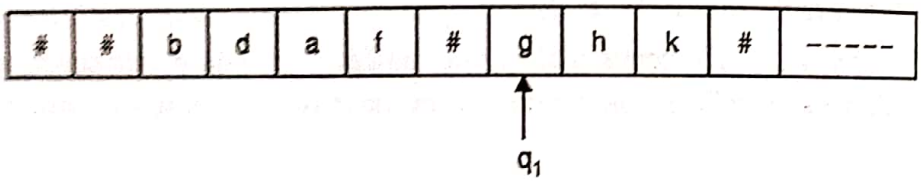
The ID of a Turing machine is the information in respect of :

(*i*) Contents of all the cells of the tape, starting from the right most cell upto atleast the last cell, containing a non-blank symbol and containing all cells upto the cell being scanned.

(*ii*) The cell currently being scanned by the machine and

(*iii*) The state of the machine.

For example let us assume that Turing machine TM is at state q_1 scanning the symbol *g* with the symbols on the tape as follows :



We can show the situation as follows :

$$(q_1, \#\#bdaf\#\underline{g}hk\#)$$

10.5. TRANSITION DIAGRAM

In some situations, graphical representation of the next-move partial function δ of Turing machine may give better idea of the behaviour of TM in comparison to the tabular representation of δ.

A Transition Diagram of the next-move functions δ of a TM is a graphical representation consisting of a finite number of nodes and directed-labelled arcs between the nodes. Each node represents a state of the TM and a label on an arc from one state (say q_1) to a state (q_2) represents the information about the required input symbol say '*a*' for the transition from q_1 to q_2 to take place and the action on the part of the control of TM. The action part consists of

(*i*) The symbol say '*b*' to be written in the current cell and

(*ii*) The movement of the tape head.

The label of an arc is generally written as $a/(b, m)$, where *m* is *L*, *R* or *N*.

Scanned with CamScanner

During the designing procedure of turning machine Tm, any string/input is assumed to be written on input taple in following form :

$$\#abbabb\underline{\#}$$

that is Head is on extreme right-blank in the starting and when machine starts processing the string at q_0 (starting state) then it is supposed to read first input symbol of the string from right to left; as follows :

$$\#abbabb\underline{\#}$$
$$\uparrow$$
q_0 (initial state of the machine)

Example 10.2. Let TM = $\{Q, \Sigma, \Gamma, \delta, q_0, h\}$

Solution.

where

$$Q = \{q_0, q_1, q_2, h\}$$
$$\Sigma = \{0, 1\}$$
$$\Gamma = \{0, 1, \#\}$$

and δ be given by the following table

	0	1	#
q_0	—	—	$(q_2, \#, R)$
q_1	$(q_2, 0, R)$	$(q_1, \#, R)$	$(h, \#, N)$
q_2	$(q_2, 0, L)$	$(q_1, 1, R)$	$(h, \#, N)$
h	—	—	—

The transition diagram for above TM will be as shown below, where we assume that 'q_0' the initial state and 'h' is a final state/half state.

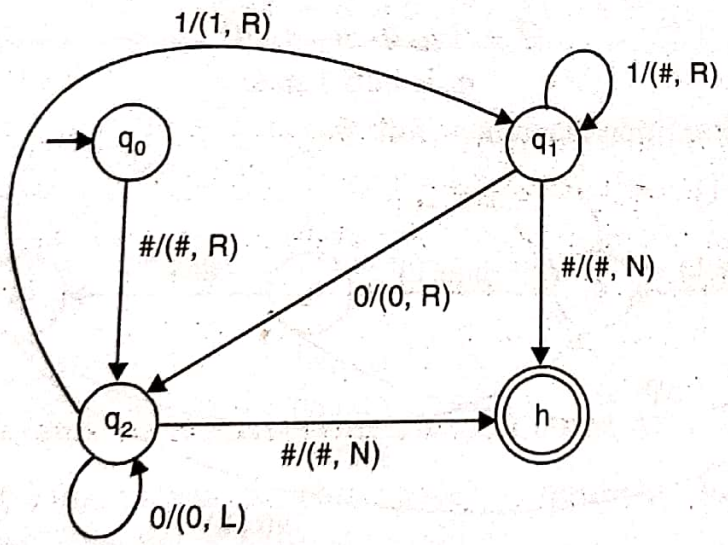


Fig. 10.2.

Example 10.3. Design a TM that recognizes the language of all strings of even length over alphabet $\{a, b\}$.

Solution. Let Turing machine is

$$Tm = (Q, \Sigma, \Gamma, \delta, q_0, h)$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{a, b, \#\}$$
$$Q = \{q_0, q_1, h\}$$

Scanned with CamScanner

Here h is half state when machine halts after accepting the language q_0 is initial state. Here next move partial function δ is given as follows :

	a	b	#
q_0	(q_1, a, L)	(q_1, b, L)	$(h, \#, N)$
q_1	(q_0, a, L)	(q_0, b, L)	*
h	*	*	Accept

Here '*' is for undefined move. We can construct the transition diagram as follows :

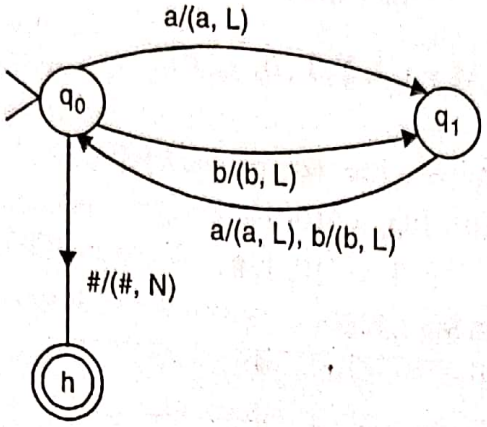


Fig. 10.3.

Example 10.4. Design a Turing Machine that accepts the language of all strings which contain *aba* as a substring.

Solution. Let Turing Machine is TM = $(Q, \Sigma, \Gamma, \delta, q_0, h)$

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

$$\Gamma = \{a, b, \#\}$$

$$Q = \{q_0, q_1, q_2, q_3, h\}$$

q_0 is initial state

and δ is defined by transition diagram as follows :

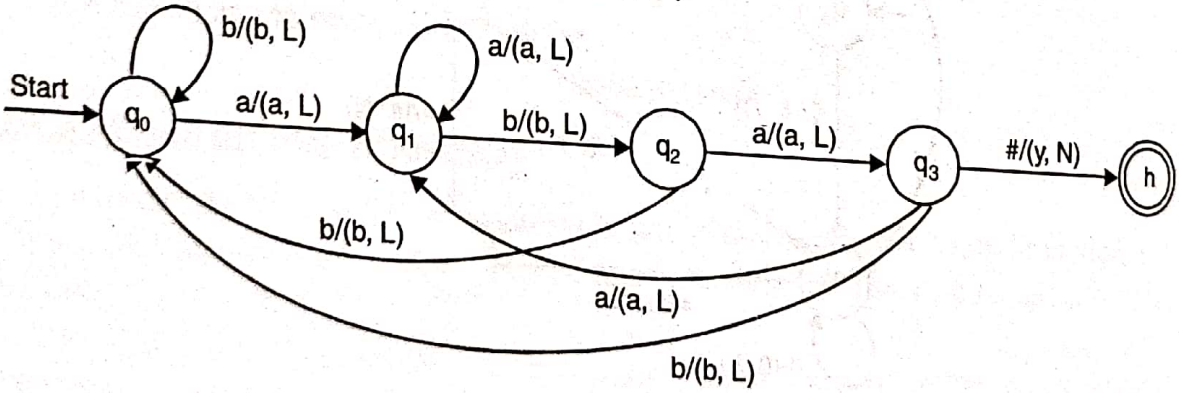


Fig. 10.4.

10.6. TURING THESIS

"The power of any computational process is captured within the class of Turing Machines."

It may be noted that Turing thesis is just a conjecture and not a theorem, hence, Turing thesis cannot be logically deduced from more elementary facts. However, the conjecture can be shown to be false, if a more powerful computational model is proposed that can recognize all the languages which are recognized by the TM model and also recognizes at least one more language that is not recognized by the TM.

Scanned with CamScanner

The Turing Machines are designed to play atleast the following three different roles :

(*i*) *As accepting devices for languages,* similar to the role played by FAs and PDAs.

(*ii*) *As a computer of functions.* In this role, a TM represents a particular function. Initial input is treated as representing an *argument* of the function. And the (final) string on the tape when the TM enters the Halt state treated as representative of the value obtained by an application of the function to the argument represented by the initial string.

(*iii*) *As an enumerator of strings of a language,* that outputs the strings of a language, one at a time, in some systematic order that is as a list.

10.7. TURING MACHINE FOR COMPUTING FUNCTIONS

Turing machine can be used to compute functions. How is it possible ? First we see how to input a string to a turing machine. We adopt the following policy to input any string to a turing machine.

The string W is presented in to the form of # W #, that is string, W is surrounded by blanks symbols from both sides and is placed on the left most square of tape, the head of turing machine is positioned at the right most blank symbol which immediately follows the string W.

This is shown by an underscore that is we use an underscore to show the current position of machine head in the tape. A turing machine is said to halt on input 'W' if we can reach to a halting state 'h', after performing some operations, that is if,

T_m = $(Q, \Sigma, \Gamma, \delta, q_0, h)$ is a turing machine then Tm is said to halt to on input W if and only if $(q, \# W \underline{\#})$ yields to $(h, \# \mu \underline{\#})$.

DEFINITION

A function $f(x) = y$ is said to be computable by a turing machine Tm $(Q, \Sigma, \Gamma, \delta, q_0, s)$, if $(s, \#\underline{x}\#) \vdash^* (h, \#\underline{y}\#)$ where x may be in some alphabet Σ_1^* and y may be some alphabet Σ_2^* and $\Sigma_1, \Sigma_2 \subseteq \Sigma$.

Symbol $\vdash^*$ has its usual meaning as in automata, means string x yields to y after a finite number of steps.

It means that if we give input x to the turing machine Tm, turing machine gives output as a string if it computes the function $f(x) = y$.

Example 10.5. Design a turing machine which compute the function $f(m) = m + 1$ for each m that belongs to the set of natural numbers.

Solution. Given function is $f(m) = m + 1$. Here we represents input m on the tape by a number of I's on the tape.

For example $m = 1$; input will be #I#,

For $n = 2$;

input will be #II# and so on.

Similarly output can be seen by the number of I's on the tape.

Let Turing machine be Tm $= (Q, \Sigma, \Gamma, \delta, q_0, h)$

where $Q = \{q_0, h\}$

$\Gamma = \{I, \#\}$

Scanned with CamScanner

q_0 is initial state.

	$\sigma = I$	$\sigma = \#$
q_0	(q_1, I, R)	$(q_1, \#, R)$
q_1	*	(h, I, R)
h	*	*

Here σ is input at any time.

Let input be $n = 3$ that is tape situation is $\#III\#$

Now let us process the given string.

(**Note :** We are processing string from right to left).

$$\#III\, q_0 \vdash \#III\#q_1$$
$$\vdash \#IIII\#h$$

Example 10.6. Prove that following function is computable

$$f(n) = n+2$$

Solution. We know that if any function is computable then there exist a turing machine for it. "so it will be sufficient to construct a turing machine to prove any function computable".

Let Turing machine for given function be

$$\text{Tm} = (Q, \Sigma, \Gamma, \delta, q_0, h)$$
$$Q = \{q_0, q_1, q_2, h\}$$
$$\Gamma = \{I, \#\}$$

q_0 is initial state and halting state is represented by 'h'.

Transition relations are defined as follows :

δ	I	$\#$
q_0	(q_1, I, R)	$(q_1, \#, R)$
q_1	*	(q_2, I, R)
q_2	*	(h, I, R)
h	*	*

Initially turing machine is in q_0 state, when it reads I or # then it changes it's state to q_1 and moves towards right (right symbol is only blank). At q_1 it encountered with # and replace it by I, change its state to q_2 and moves towards right. At q_2, # is replaced by I and machine halts after moving right.

Let $n = 3$ that is $\#III\#$

then Tm, behaves as follows L

$$\#IIIq_0 \vdash \#III\#\, q_1$$
$$\vdash \#IIII\#q_2$$
$$\vdash \#IIIII\#\, h$$

Example 10.7. Design a turing machine that replace every 0 and 1 with every 1 with 0 in a binary string.

Or

Prove that function $f(w) = \overline{w}$ is computable where $w \in \{0, 1\}^*$ and $\overline{w}$ the one's complement of w.

Scanned with CamScanner

Solution. We want to design a turing machine that takes input as #101# then its output would be #010#. This machines can be designed by the following idea; scan backwards through the input, replace every 0 with 1 and every 1 with 0, until the blank marking the left most sequence is found. In order to leave the tape in its proper format, the machine must then return its head to the blank marking the right end of the input.

Let Turing Machine is $\quad Tm = (Q, \Sigma, \Gamma, \delta, q_0, h)$

$$Q = \{q_0, q_1, h\}$$
$$\Sigma = \{0, 1\}$$
$$\Gamma = \{0, 1, \#\}$$

q_0 is initial state.

δ can be defined by following transition diagram.

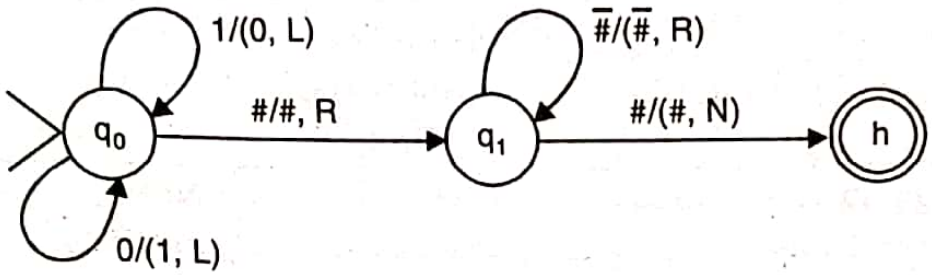


Fig. 10.5.

Here $\overline{\#}$ represent non-blank symbol (in this case either 0 or 1).

Let input string be w = #101100#

$$\begin{aligned}
\#101100\, q_0 &\vdash \#10110q_01 \\
&\vdash \#1011q_011 \\
&\vdash \#101q_011 \\
&\vdash \#10q_00011 \\
&\vdash \#1q_010011 \\
&\vdash \#q_0010011 \\
&\vdash \#q_1010011 \\
&\vdash \#0q_110011 \\
&\vdash \#01q_10011 \\
&\vdash \#010q_1011 \vdash \#0100q_111 \\
&\vdash \#01001q_11 \vdash \#010011q_1\# \vdash \#010011\#h
\end{aligned}$$

Scanned with CamScanner

Now at this level we can compare, the language accepted by the regular expression, context free grammar and type-0 grammar (*I* will introduced it letter); as follows in Fig. 10.6.

Language defined by	*Corresponding Acceptor*	*Non-determinism = Determinson*	*Language closed under*	*What can be decided*	*Example of application*
Regular Expression	Finite Automata	Yes kleene star, intersection complement	Under union, Product, finiteness, membership	Equivalence, emptiness, circuit	Text editors, sequential
Context-free grammar	Pushdown automata	No	Union, product, kleene star	Emptiness finiteness membership	Programming language compilers statements
Type-0 grammar	Turing machine, post machine DPDA, *n*PDA	Yes	Union, product, Intersection, Kleene star.	Not much	Computers

Fig. 10.6.

Example 10.8. Design a turing machine that recognises the set of all strings of 0's and 1's containing at least one 1.

Solution. Designing such a turing machine is very simple. For the designing this turing machine we will move towards left until 1 is found, and then halt.

Let Turing machine be $Tm = (Q, \Sigma, \Gamma, \delta, q_0, h)$

where $Q = \{q_0, q_1, h\}$

$\Gamma = \{0, 1, \#\}$

$\Sigma = \{0, 1\}$

q_0 is initial state

and transition relation δ can be represented by following transition diagram :

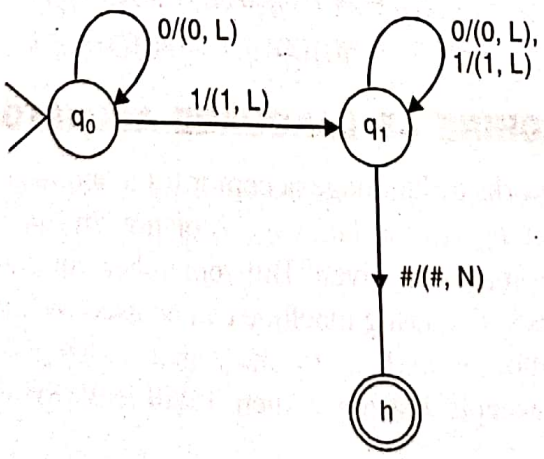


Fig. 10.7.

Here halt state *h* is reached only when at least one 1 is encountered and *h* works as accepted state.

Scanned with CamScanner

Example 10.17. Design a Turing machine which accepts the language $L = \{w \in (a, b)^*/w$ has equal number of a's and b's$\}$.

Solution. The language L is the set of strings (included null string) in which every strings either start by a or b but number of a's and b's in the string are always equal. That is turing machine should accept strings like ab, $aaabbb$, $baba$, $abab$, and so on and it must reject strings like a, b, abb, baa, $abbb$, $abbab$ and so on.

The Turing machine that accepts L is shown below :

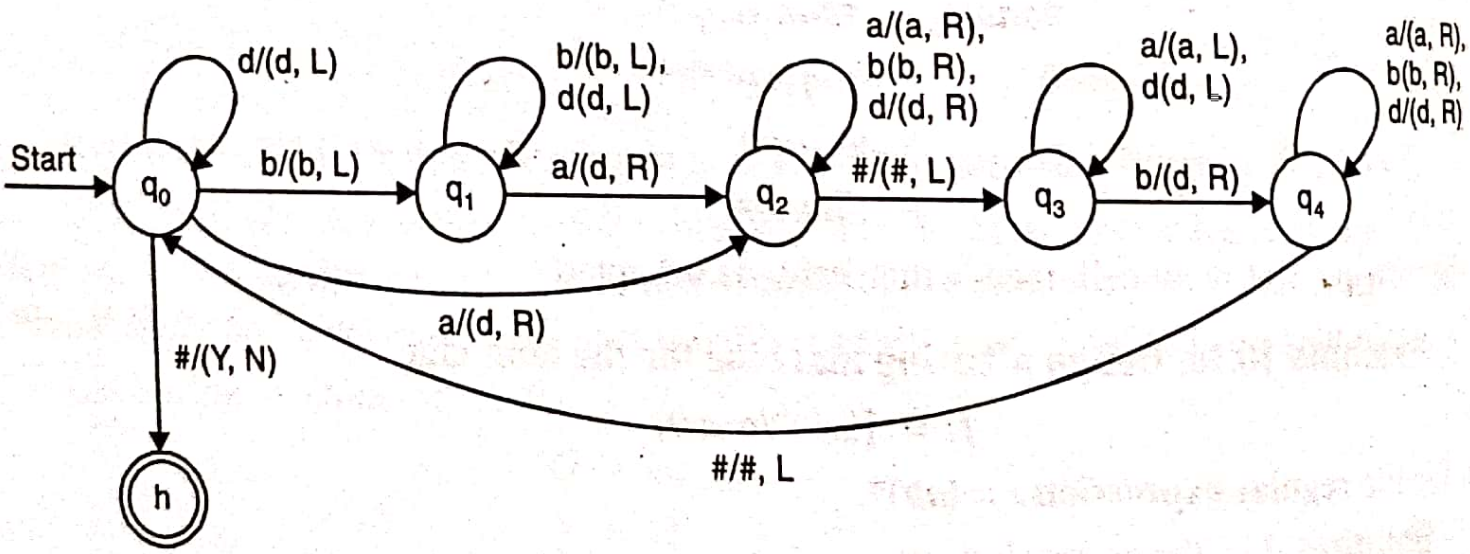


Fig. 10.24.

Y : Stands for Accept

In one main cycle, this machine changes one a and one b in two d's. It does so by scanning the the non-blank part of take two times from right to left, once for an 'a' and once for 'b', returning each time to the right end of the tape before starting the next scan. This sequence is repeated until the supply of a or b is exhausted.

Scanned with CamScanner

Now if machine sees all d's on the non-blank part then it halts that is, it accept the string. The move which are not available in the machine leads towards not accepting state.

Following is the sequence of operation on string abba.

$$\#abbaq_0 \vdash \#abbdq_2$$
$$\vdash \#abbd\#q_2$$
$$\vdash \#abbdq_3$$
$$\vdash \#abbq_3d$$
$$\vdash \#abddq_4$$
$$\vdash \#abdd\#q_4$$
$$\vdash \#abddq_0 \quad \text{(one cycle is completed)}$$
$$\vdash \#abdq_0d$$
$$\vdash \#abq_0dd$$
$$\vdash \#aq_1bdd$$
$$\vdash \#dbq_2dd$$
$$\vdash \#dbdq_2d$$
$$\vdash \#dbddq_2$$
$$\vdash \#dbdd\#q_2$$
$$\vdash \#dbddq_3$$
$$\vdash \#dbdq_3d$$
$$\vdash \#dbq_3dd$$
$$\vdash \#dddq_4d$$
$$\vdash \#ddddq_4$$
$$\vdash \#dddd\#q_4$$
$$\vdash \#ddddq_0$$
$$\vdash \#dddq_0d$$
$$\vdash \#ddq_0dd$$
$$\vdash \#dq_0ddd$$
$$\vdash \#q_0dddd$$
$$\vdash Yhdddd$$

Scanned with CamScanner

11 CHAPTER

EXTENSION OF TURING MACHINE

INSIDE THIS CHAPTER

11.1. Introduction 11.2. Recursively Enumerable and Recursive Languages 11.3. Church's Thesis 11.4. Universal Turing Machines 11.5. The Halting Problem 11.6. Undecidability/ Decidability 11.7. Pushdown Automata with Two Stacks

11.1. INTRODUCTION

Now it becomes clear that turing machine can perform fairly powerful computation. In order to better understand their surprising power, we shall consider the effect of extending the turing machine model in various directions. The "new, improved models" of the turing machine can in each instance be simulated by the standard model. Such results increase our confidence that the turing machine is indeed the ultimate computational device, the end of our progression to move and move powerful automata. A side benefit by these results is that we shall feel free subsequently to use the additional features when designing turing machines to solve particular problems.

The extensions of Turing machine considered are :

(*i*) There may be several tapes instead of one only, each tape having its own independent head.

(*ii*) The tape may be allowed to be infinite in both the directions.

(*iii*) There may be more than one head scanning various cells of the tape. Two or more heads simultaneously read the same cell or may attempt to write in the same cell.

(*iv*) The tape may be K-dimensional, $K \geq 2$, instead of only one-dimensional.

(*v*) For a given pair of current state and symbol under the Head, instead of atmost one possible move, there may be any finite, possible zero, number, of next moves. This is called **Non-deterministic Turing Machine.**

11.1.1. Multiple Tapes Turing Machine

One can think of turing machines that have several tapes (see Fig. 11.1). Each tape is connected to the finite control by means of a read/write head (one on each tape).

The machine can in one step read the symbols scanned by all its heads and then, depending on these symbols in current state, rewrite some of those scanned squares and move some of the heads to the left or right, in addition to changing the state.

Scanned with CamScanner

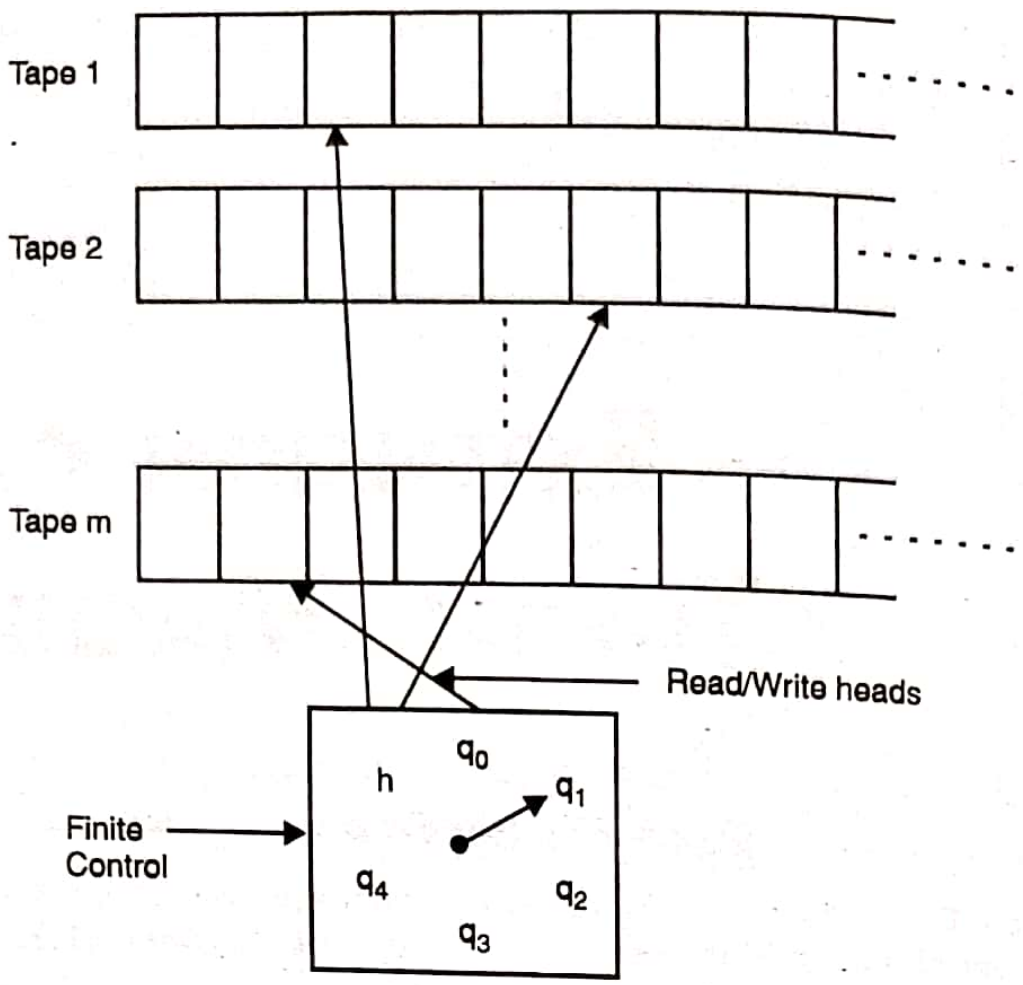


Fig. 11.1.

For any fixed integer $m \geq 1$, an m-tape turing machine is a turing machine equipped with m tapes and corresponding heads. Thus standard turing machine studied so far in this chapter is just an m-tape turing machine, with $m = 1$.

Scanned with CamScanner

11.1.2. Two-way Infinite Tape

The extensions are distinguished from each other and from the standard Tm through different definitions of next-move relation δ and of configurations for each of the extension. So here we discuss the extension only in terms of definitions of δ and of configuration.

Like standard Tm, in this case also, the next-move is given by δ as a partial function from

$$(Q \times \Gamma) \text{ to } (Q \times \Gamma \times \{L, R, N\})$$

The following three points need to be noted in respect of configurations of Two-way Turing Machine.

(*i*) **Configuration/Instantaneous Description.** In standard Tm, if there are a number of left-most positions which contain blanks, then those are included in the configuration that is if the one-way configuration tape is of the form

Scanned with CamScanner

$$\#ab\#c\underset{\uparrow}{d}ef\#\#\ldots\#$$
$$q_2$$

then the configuration in the standard Tm is written as :

$$(q_2, \#ab\#c\underline{d}ef\#\#\ldots\#)$$

However, in the two-way infinite tape Tm, both left-hand and right-ha parts of the tape are symmetrical in the sense that there is an infinite continuo sequence of blanks on each of the right hand and left hand of the sequence of no blanks. Therefore, in case of two-way infinite tape, if the above string is on t tape then it will be in the form.

$$(q_2, \#\ldots\#\#ab\#c\underline{d}ef\#\#\ldots\#)$$

(*ii*) **No Ceasing of operation without Halting.** In this case, as there is no le end of the tape, therefore, there is no possibility of jumping off the left-end of the tape

(*iii*) **The empty Tape configuration.** When at some point of time all the cells c the Tape are #'s and the state is say q, then configuration in Two-way tape may b denoted as :

$$(q, \#)$$

where only the current cell containing # is shown in configuration.

TIPS

Despite the fact that, it is possible in the new model of computer to move left as far as required. The model does not provide any additional computational capability.

11.1.3. Multiple Heads Turing Machines

In order to simplify the discussion, we assume that there are only two Heads on the tape. The tape is assumed to be one-way infinite. We explain the involved concepts with the help of an example. Let the content of the tape and the position of the two heads that is H_1 and H_2, be as given below :

$$\#\#a\underset{\uparrow}{b}c\#d\underset{\uparrow}{e}f\#\#\ldots$$
$$H_2 \qquad H_1$$

Further, let the state of the Tm be q.

The move function of the two-head one-way turing may be defined as

δ(state, symbol under Head1, symbol under Head2)

$$= (\text{New state}, (S_1, M_1), (S_2, M_2))$$

where S_i is the symbol to be written in the cell under H, (the ith Head) and M_i denotes the movement of H_i, where the movement may be L, R or N and further L denotes movement to the left, R denotes movement to the right of the current cell and N denotes 'no movement of the Head'.

Two special cases of the δ function defined above, need to be considered :

(*i*) What should be written in the current cell when both Heads are scanning the same cell at a particular time and the next moves (S_1, M_1), (S_2, M_2) for the two heads, are such that $S_1 \neq S_2$ that is symbol to be written in current cell by H_1 is not same as symbol to be written in current cell by H_2.

In such a situation, a general rule may be defined, say, as whatever is to be done by H_1 will take precedence over whatever is to be done by H_2.

Scanned with CamScanner

(*ii*) For two-head one way tape, a configuration shall be called hanging if $\delta(q,$ symbol under H_1, symbol under $H_2) = (q_1, (S_1, M_1), (S_2, M_2))$ is such that either

(*a*) Symbol under H_1 is in the left-most cell and M_1 is L, that is movement of H_1 is to be to the left,

or

(*b*) Symbol under H_2 is in the left-most cell and M_2 is L, that is movement of H_2 is to be to the left.

The above discussion can be further extended easily to the case when number of Heads is more than two. The power of Tm is not enhanced by the use of extra heads.

11.1.4. K-dimensional Turing Machine

Again to facilitate the understanding of the basic ideas involved, let us discuss initially only Two-dimensional Turing Machine. Then these ideas can be easily generated to K-dimensional case, where $K > 2$.

In the case of two-dimensional tape as shown below, we assume that the tape is bounded on the left and the bottom.

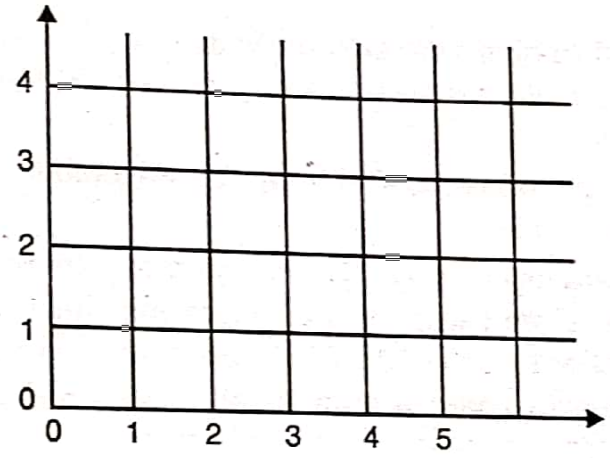


Fig. 11.3.

Each cell is given an address say (i, j) where i is the row-number of the cell and j is the column number of the cell.

A configuration of a two-dimensional Tm at a particular time may be described in terms of finitely many of the triplets of the form, (i_1, i_2, c) where for each such triplets of the form, (i_1, i_2, c) where for each such triplet (i_1, i_2) is the address of a cell and c denotes the contents of the cell. Only these cells are included in an ID, for which, the contents, are non-blank symbols.

In the configuration or ID, order of the cell which are included in an ID, **Row Major Ordering** is to be followed, that is first all the elements in the row with least index are included in the ID, followed by the elements of the row with next least index and so on. Within cells of each row, the cell with non # contents and having least column number is included first followed by the non blank cell with next least column number and so on.

Let $q \in Q$, $C_K \in \Gamma - \{\#\}$, that is C_K is a non-blank tape symbol.

The configuration at a particular instant is denoted by

$$(q, (H_1, H_2) (i_1, j_1, C_{i_1j_1}), (i_2, j_2, C_{i_2j_2}) \ldots (i_k, j_k, C_{i_kj_k}) \ldots))$$

where each of $C_{i_1j_1}$, $C_{i_2j_2}$ ... is non-blank and these are the only non-blank on the tape.

Scanned with CamScanner

Also (H_1, H_2) denotes the location of the cell currently being scanned that is the cell under the Head.

11.1.5. Non-deterministic Turing Machine

We have already seen that when finite automata are allowed to act non-deterministically no increase in the computational power, but that non-determinism in push down automata are more powerful than deterministic ones.

We can also imagine turing machine that acts non-deterministically. In standard Tm, to each pair of current state (except the halt-state) and the symbol being scanned, there is a unique triplet comprising of the next state, unique action in terms of writing a symbol in the cell being scanned and the motion, if any, to the right or left. However, in case of NDTM (non deterministic turing machine), to each pair (q, a) with q as current state and 'a' as symbol being scanned, there may be a **finite set of triplets** $\{(q_i, a_i, m_i); i = 1, 2, ...\}$ of possible moves. The set of triplets may be empty that is for some particular (q, a) the Tm may not have any next move. Or alternatively the set $\{(q_i, a_i, m_i)\}$ may have more than one triplet, meaning thereby that the NDTM in the state q and scanning symbol 'a' and can choose next move from the set $\{(q_i, a_i, m_i)\}$ of next moves.

From the above discussion it can be easily understood that standard Tm is a special case of the NDTM in which for each (q, a) the set $\{(q_i, a_i, m_i)\}$ in next moves is a singleton set or empty.

In order to define the concept of NDTM and a configuration in NDTM, we assume that the tape is one-way infinite.

Proper non-determinism means that at some stage, there are atleast two next possible moves. Now, if we engage two different machines to work out further possible moves according to each of these two moves, the two can work independent of each other. This means non-determinism allows parallel computation.

DEFINITION

A non-deterministc Turing machine is a six tuple $(Q, \Sigma, \Gamma, \delta, q_0, h)$ where

Q : set of states
Σ : set of input symbols
Γ : set of tape symbols
δ : $(Q \times \Gamma) \rightarrow$ power set of $(Q \times \Gamma \times \{L, R, N\})$
q_0 : initial state and
h : hut state, $h \in Q$

The concept of a configuration is same as in case of standard Tm. But the concept of 'yield in one step' denoted by Tm has different meaning. Here one configuration may yield more than one configuration.

Scanned with CamScanner

11.2. RECURSIVELY ENUMERABLE AND RECURSIVE LANGUAGES

When a Turing machine executes an input, there are four possible outcomes of execution. Then Tm

(*i*) Halts and accept the input

(*ii*) Halts and rejects the input

(*iii*) Never halts (fall into loop), or

(*iv*) Crash.

Scanned with CamScanner

We also know that the language accepted by Tm is known as recursively enumerable and the set of recursive languages is subset of recursive enumerable set.

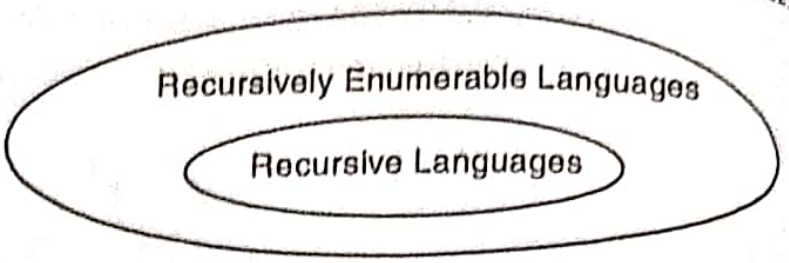


Fig. 11.5.

The set of recursively enumerable languages are precisely those languages that can be listed (enumerated) by a turing machine. Basically, if a Tm can generate all the strings in a language, another Tm can accept all the strings that are not in the given language. (Strings that are not in the language may be rejected or may cause the turing machine to go into an infinite loop).

TIPS

A language is recursive if there exists a Turing machine that accepts every string of the language and rejects every string that is not in the language. So, we are sure about the rejection of the strings with are not in the given recursive language.

Scanned with CamScanner

11.3. CHURCH'S THESIS

What function can not be computed by turing machines ? This is very surprising. It is believed that there are no functions that can be defined by humans, whose calculation can be described by any well-defined mathematical algorithm that people can be taught to perform, that can not be computed by Tm's. The Tm is believed to be the ultimate calculating mechanism.

In other words "*No computational procedure will be considered an algorithm unless it can be represented as a turing machine*". This statement is called Church's thesis because Alonzo Church (1936) gave many sophisticated reasons for believing it. Church's original statement was slightly different because his thesis was presented slightly before the turing invented his machines.

Church actually said that any machine that can do certain list of operations will be able to perform all conceivable algorithms. He tied together what logicians had called recursive functions and computable functions. Tm's can do all that Church asked, so they are one possible model of the universal algorithm machine Church described.

TIPS

Unfortunately, Church's thesis can not be a theorem in mathematics because ideas such as "can ever be defined by humans" and "algorithm that people can taught to perform" are not part of any known mathematics. **There are no axioms that deal with "people".**

11.4. UNIVERSAL TURING MACHINES

We can consider turing machine in both ways : The turing machine is an "unprogrammable" piece of hardware, specialized at solving one particular problem, with instructions that are "*hard-wired at the factory*".

We shall now take the opposite point of view. We shall argue that turing machines are also software. That is, we shall show that there is a certain "generic" turing machine that can be programmed, about the same way that a general purpose computer can, to solve any problem that can be solved by turing machine. The "program" that makes this generic machine behave like a specific machine Tm will have to be a description of Tm. In other words, we shall be thinking of the formalism of turing machines as a programming language, in which we can write programs. Programs written in this language can then be **interpreted** by **universal machine** that is to say another program in same language.

DEFINITION

To begin, we must present a general way of specifying turing machines, so that their descriptions can be used as input to other turing machines.

Universal turing machines '*U*' takes two arguments, a description of a machine Tm, "Tm", and a description of an input string *w*, "*w*". We want *U* to have the following property : *U* halts on input "Tm" "*w*" if and only if Tm halts on input *w*.

$$U(\text{“}m\text{”}\ \text{“}w\text{”}) = \text{“}m\ (w)\text{”}.$$

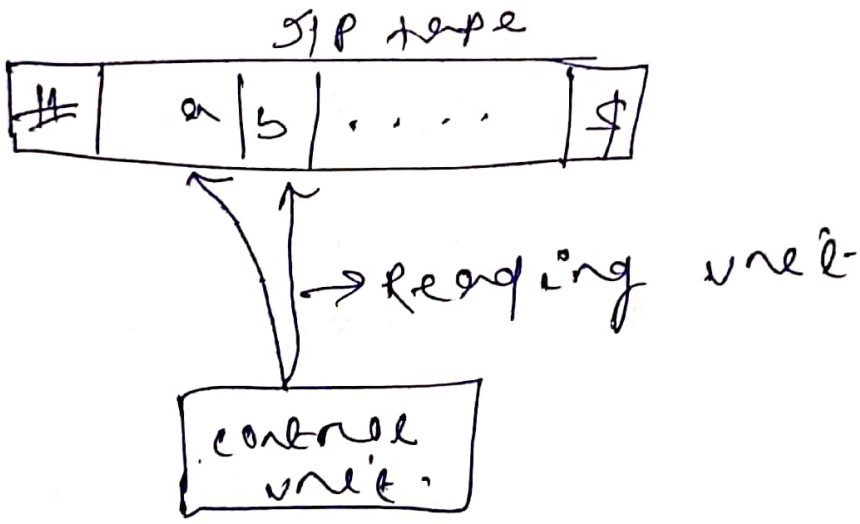
It is the functional notation of universal turing machine.

Scanned with CamScanner

Context sensitive Language:

It's one of the type of TM where

$|LHS| \leq |RHS|$ &



→ left end marker

$ → Right end marker.

Scanned with CamScanner