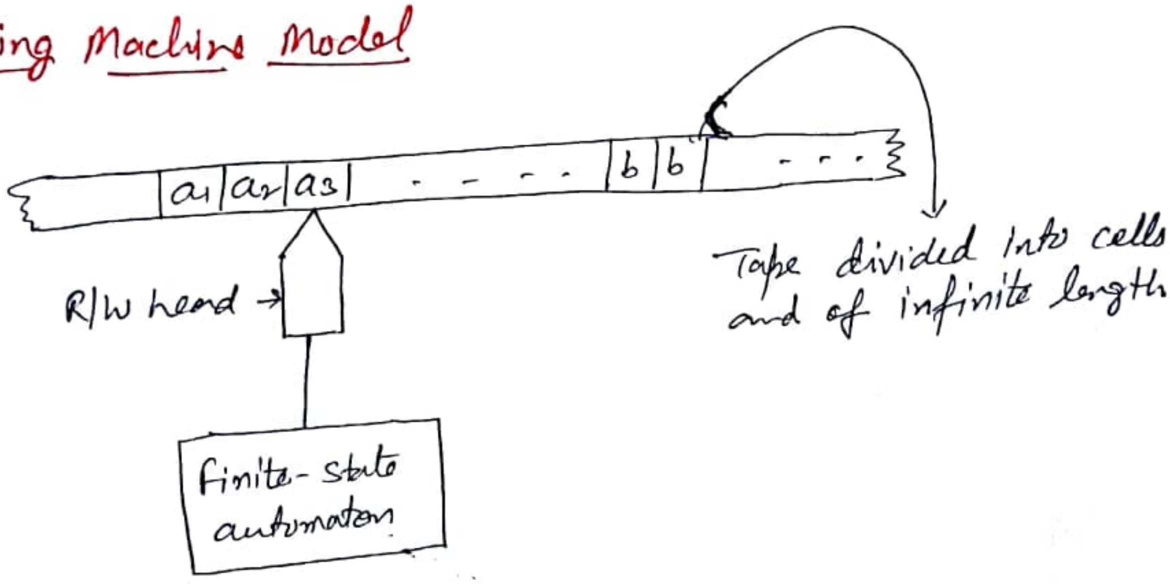
Turing Machine

The turing machine (TM) is a simple mathematical model of a general-purpose computer.

Turing Machine Model



Definition

A Turing machine M is a 7-tuple, viz. $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$, where

1. Q is a finite nonempty set of states.
2. Γ is a finite nonempty set of tape symbols.
3. $b \in \Gamma$ is the blank.
4. Σ is a nonempty set of input symbols and is a subset of Γ and $b \notin \Sigma$.
5. δ is transition function
$$Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\},$$
6. $q_0 \in Q$ is the initial state, and
7. $F \subseteq Q$ is the set of final states.

Representation of Turing Machine

(i) Instantaneous descriptions (ID) using move-relations $\vdash$
(ii) Transition table
(iii) Transition diagram

(i) ID: An ID of a Turing Machine M is a string $\alpha \beta \gamma$, where β is the present state of M, the entire input string is split as $\alpha\gamma$, the first symbol of γ is the current symbol a under R/W head and γ has all the subsequent symbols of the input string, and the string α is the substring of the input string formed by all the symbols to the left of a.

(ii) Representation by Transition Table

Present State	Tape symbol		
	b	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3		bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
q_5	$0Lq_2$		

eg. Consider the given TM description in table. Draw the computation sequence of the input string 00.

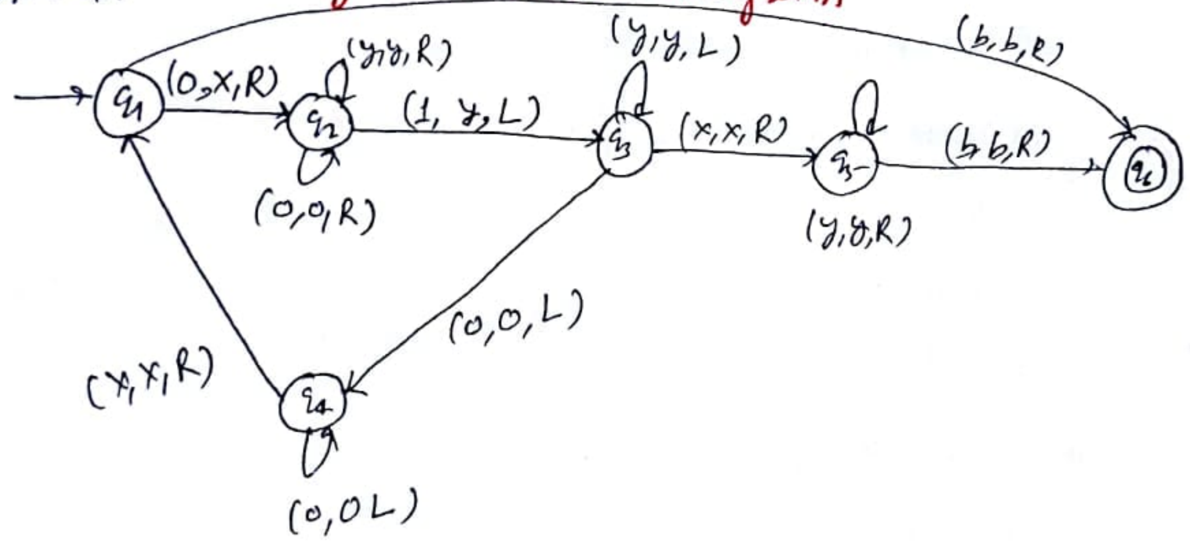
$q_1 00b \vdash 0q_1 0b \vdash 00q_1 b \vdash 0q_2 01 \vdash q_2 001 \vdash q_2 b001$

$\vdash bq_3 001 \vdash bbq_4 01 \vdash bb0q_4 1 \vdash bb01q_4 b \vdash bb010q_5 1$

$\vdash bb01q_2 00 \vdash bb0q_2 100 \vdash bbq_2 0100 \vdash bq_2 b0100$

$\vdash bbq_3 0100 \vdash bbbq_4 100 \vdash bbb1q_4 00 \vdash bbb10q_4 0$
$\vdash bbb100q_4 b \vdash bbb100q_5 b \vdash bbb100q_2 00 \vdash$
$bbb10q_2 000 \vdash bbb1q_2 0000 \vdash bbbq_2 10000 \vdash bbq_2 b10000$
$\vdash bbbq_3 1 0000 \vdash bbbbq_5 0000$

Representation by transition diagram
for 0011



Eg Consider the Turing Machine M described by the transition table given in Table. Describe the processing of (a) 011, (b) 0011 (c) 001 using IDs. Which of the above strings are accepted by M?

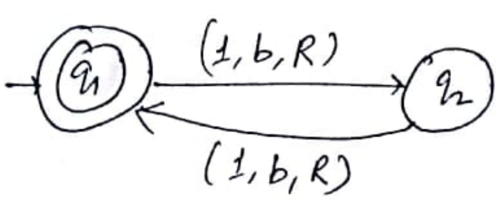
Present state	Tape symbol 0	1	x	y	b
$\to q_1$	xRq_2				bRq_5
q_2	$0Rq_2$	yLq_3		yRq_2	
q_3	$0Lq_4$		xRq_5	yLq_3	
q_4	$0Lq_4$		xRq_1		
q_5					
(q_6)				yRq_5	bRq_6

(a) $q_1 011 \vdash x q_2 11 \vdash q_3 x y 1 \vdash x q_5 y 1 \vdash x y q_5 1$.
As $\delta(q_5, 1)$ is not defined, M halts; so the input string 011 is not accepted.

(b) $q_1 0011 \vdash x q_2 011 \vdash x 0 q_2 11 \vdash x q_3 0 y 1$
$\vdash q_4 x 0 y 1 \vdash x q_1 0 y 1 \vdash x x q_2 y 1 \vdash x x y q_2 1$
$\vdash x x q_3 y y \vdash x q_3 x y y \vdash x x q_5 y y \vdash x x y q_5 y$
$\vdash x x y y q_5 b \vdash x x y y b q_6$
M halts. As q_6 is accepting state so it is accepted by TM.

(c) $q_1 001 \vdash x q_2 01 \vdash x 0 q_2 1 \vdash x q_3 0 y \vdash q_4 x 0 y$
$\vdash x q_1 0 y \vdash x x q_2 y \vdash x x y q_2 b \vdash$
M halts. As q_2 is not an accepting state, 001 is not accepted by M.

Eg: Design a TM to recognise all strings consisting of even number of 1's.



Languages Accepted by TM :→

Context Sensitive & Type 0

Variants of Turing Machine :→

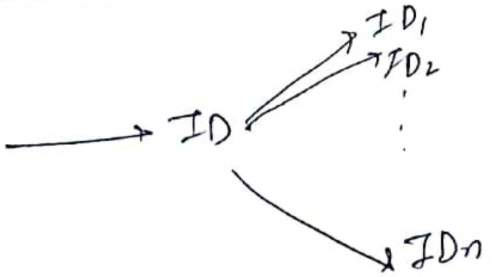
1. More No. of Heads (R/W)
2. Tape is two or three dimensional
3. Adding special purpose memory, such as stack or special purpose registers

TM as Computer of Integer function:

Universal Turing Machine :-

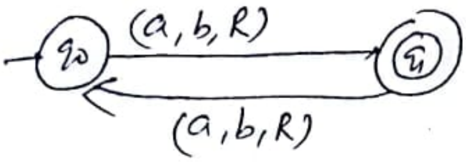
We must design a machine that accept two inputs i.e. (i) the input data and (ii) a description of computation (Algorithm).

Nondeterministic Turing Machine :→

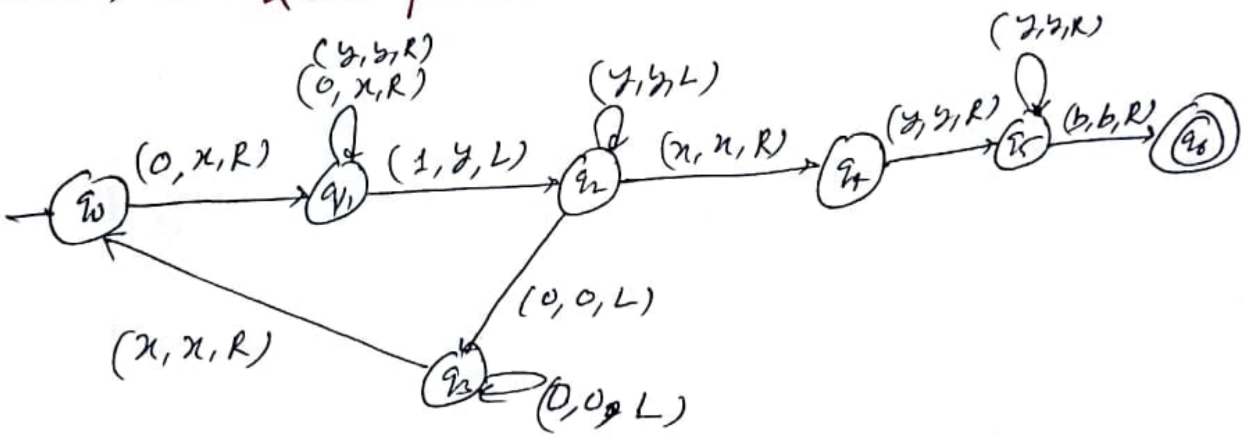


Transition in nondeterministic TM.

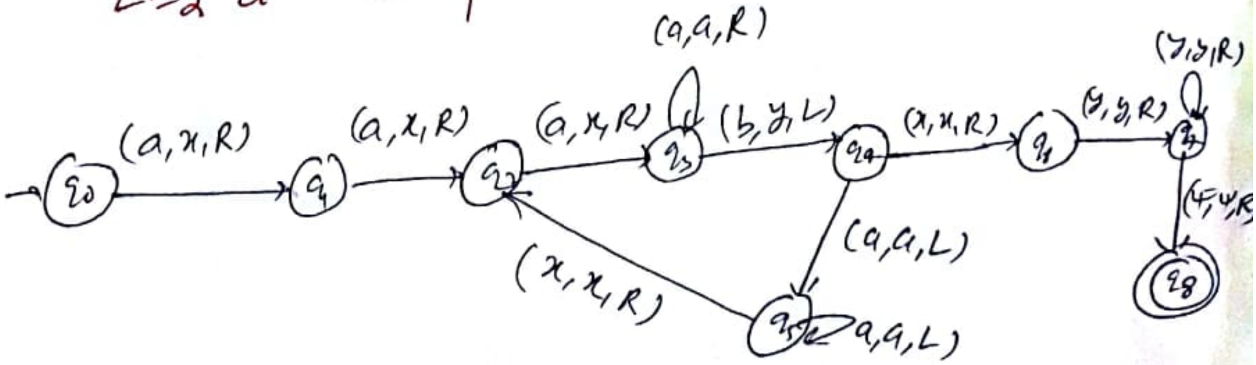
Design a TM to recognize all strings consisting of an odd number of a's.



Construct a Turing Machine which accepts the regular expression, $L = \{0^n 1^n \mid n \geq 1\}$.



Construct a Turing Machine for the language $L = \{a^{n+2} b^n \mid n > 0\}$



Halting Problem of Turing Machine:

halting problem of Turing machines is unsolvable.

Definition:-
A class of problems with two outputs/(yes/no) is said to be solvable (decidable) if there exists some definite algorithm which always terminates (halts) with two outputs (yes/no). Otherwise, the class of problem is said to be unsolvable (undecidable)

Theorem (Turing Theorem) The halting problem (HP) of Turing machine over $\Sigma = \{0,1\}$ is unsolvable (i.e. the problem of determining whether or not an arbitrary Turing machine 'M' over $\{0,1\}$ halts for an arbitrary inputs x in Σ^* is unsolvable)

Proof:- We proof the theorem by contradiction.
Let M be an arbitrary turning machine. Let $d(M)$ be the encoded binary string representing M. Then the machine-string pair (M, x) will have $d(M)*x$ as its encoded description.
According to our assumption, HP is solvable. Hence there exists an algorithm P ~~as followers~~ which decides the HP. i.e. (a) if M halts for input x, then P reaches an accept halt. (b) if M does not halt for input x, then P reaches a reject halt.

Let us construct a new algorithm Q, based on P as follows: (c) It takes d(M) as input and copies it to obtain d(M)*d(M) and then applies algorithm P to this input (i.e. d(M)*d(M)), (d) Q loops for ~~ever~~ ever if P reaches an accept halt and Q halt if P reaches a reject halt.

By Church's thesis, there exists a Turing machine, say M', which can execute the algorithm Q. Since the algorithm P, as also Q, works for an arbitrary machine M, Q also works for the Turing machine M'. So we take M = M'. From (d) and (a) we can conclude that M' loops for ever if M' halts. From (d) and (b) we conclude that M' halts if M' loops for ever. Thus, we obtain the conclusion "M' halts if and only if M' loops". This is a contradiction and, therefore, HP is unsolvable.

The Post Correspondence Problem

Emil Post in 1946.

The problem over an alphabet Σ belongs to a class of yes/no problems and is stated as follows:

Consider two lists $x = (x_1, \ldots x_n)$, $y = (y_1 \ldots y_n)$ of nonempty strings over an alphabet $\Sigma = \{0, 1\}$. The PCP is to determine whether or not there exist $i_1, \ldots, i_m$ where $1 \le i_j \le n$, such that

$$x_{i_1} \ldots x_{i_m} = y_{i_1} \ldots y_{i_m}$$

Lecture. 58

Eg. Does PCP with two lists $x = (b, bab^3, ba)$ and $y = (b^3, ba, a)$ have a solution.

Solution The required sequence is given by $i_1 = 2$, $i_2 = 1$, $i_3 = 1$, $i_4 = 3$ i.e. $(2, 1, 1, 3)$, and $m = 4$.

Thus corresponding strings are

$$\underset{x_2}{\boxed{bab^3}}\ \underset{x_1}{\boxed{b}}\ \underset{x_1}{\boxed{b}}\ \underset{x_3}{\boxed{ba}} = \underset{y_2}{\boxed{ba}}\ \underset{y_1}{\boxed{b^3}}\ \underset{y_1}{\boxed{b^3}}\ \underset{y_3}{\boxed{a}}$$

Thus, PCP has a solution.

Church's Thesis

The Church Turing thesis says that any real-world computation can be translated into an equivalent computation involving a Turing machine.