

## UNIT-V

Subject-Theory of Computation

**Prof. Shweta Tiwaskar**

[Shweta.tiwaskar@viit.ac.in](mailto:Shweta.tiwaskar@viit.ac.in)

**Department of Computer Engineering**



**BRACT'S, Vishwakarma Institute of Information Technology, Pune-48**

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

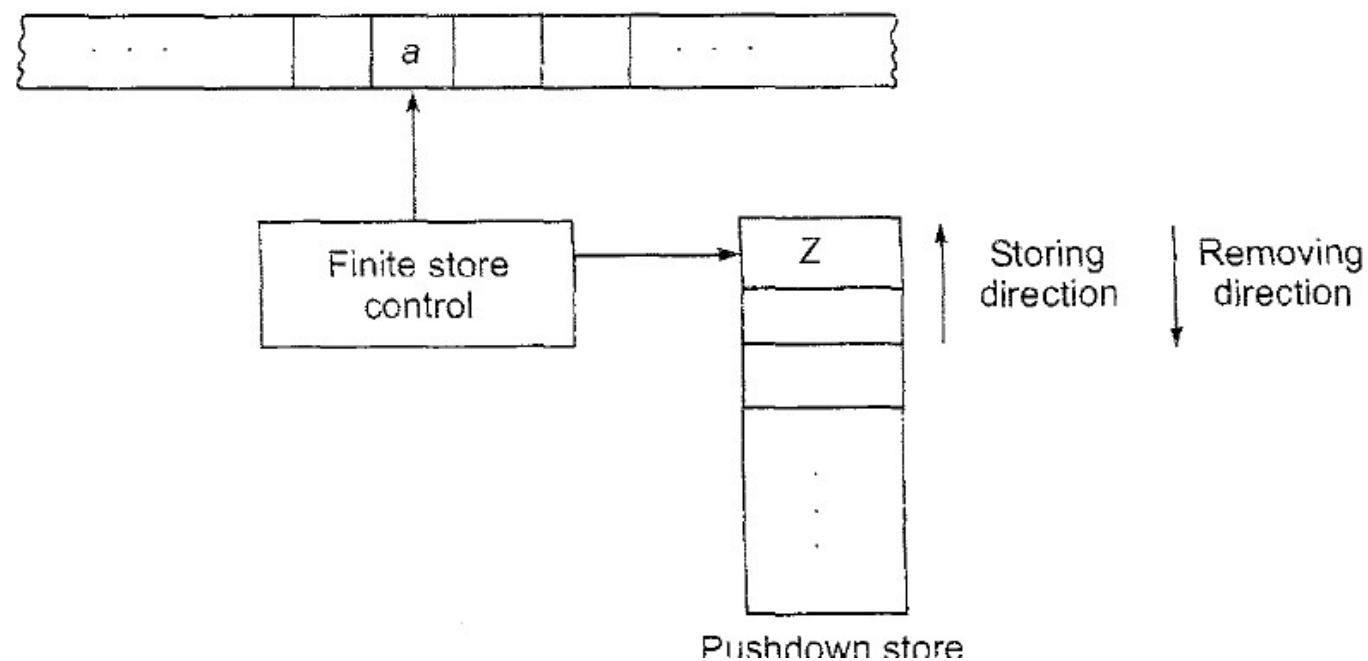
(NBA and NAAC accredited, ISO 9001:2015 certified)

## Turing Machine(TM)

- In the early 1930s. mathematicians were trying to define effective computation
- Alan Turing in 1936, Alano Church in 1933, S.C. Kleene in 1935
- Gave various models using the concept of Turing machines, lambda-calculus, combinatory logic
- were formulated much before the computers were devised
- Among these formalisms, the Turing's formulation is accepted as a model of algorithm or computation
- The Church-Turing thesis states that any algorithmic procedure that can be carried out by human beings/computer can be carried out by a TM

- TM are useful in several ways.
- As an automaton, the TM is the most general model. It accepts type-0 languages
- It can also be used for computing functions

# Model of a Pushdown automaton



- (i) a new symbol to be written on the tape in the cell under the R/W head,
- (ii) a motion of the R/W head along the tape: either the head moves one cell left (L), or one cell right (R),
- (iii) the next state of the automaton, and
- (iv) whether to halt or not.

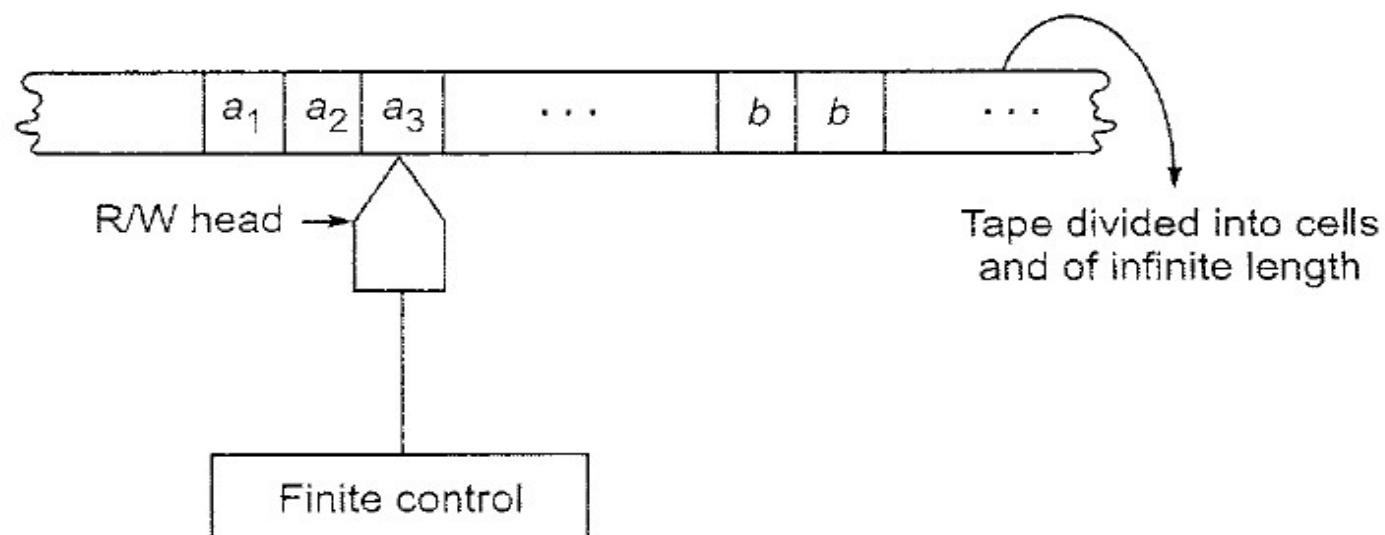


Fig. 9.1 Turing machine model

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

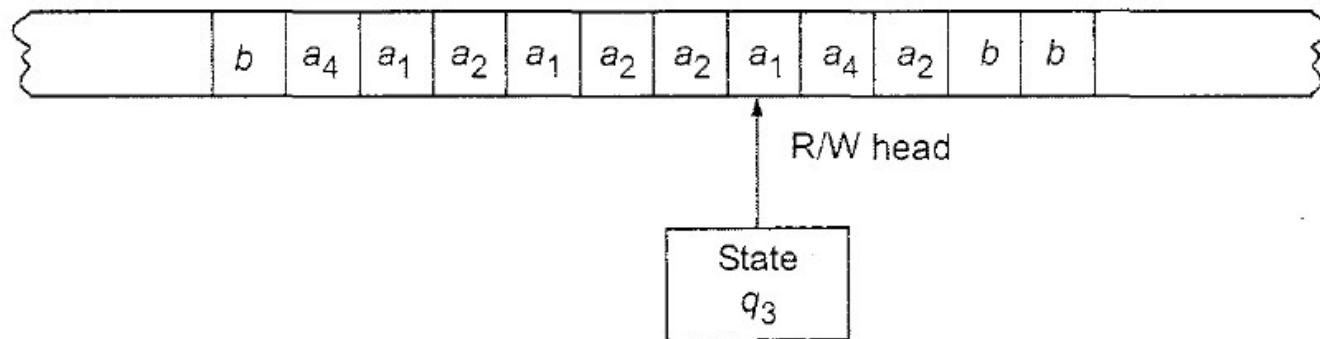
1.  $Q$  is a finite nonempty set of states,
2.  $\Gamma$  is a finite nonempty set of tape symbols,
3.  $b \in \Gamma$  is the blank,
4.  $\Sigma$  is a nonempty set of input symbols and is a subset of  $\Gamma$  and  $b \notin \Sigma$ .
5.  $\delta$  is the transition function mapping  $(q, x)$  onto  $(q', y, D)$  where  $D$  denotes the direction of movement of R/W head;  $D = L$  or  $R$  according as the movement is to the left or right.
6.  $q_0 \in Q$  is the initial state, and
7.  $F \subseteq Q$  is the set of final states.

- The acceptability of a string is decided by the reachability from the initial state to some final state. So the final states are also called the accepting states

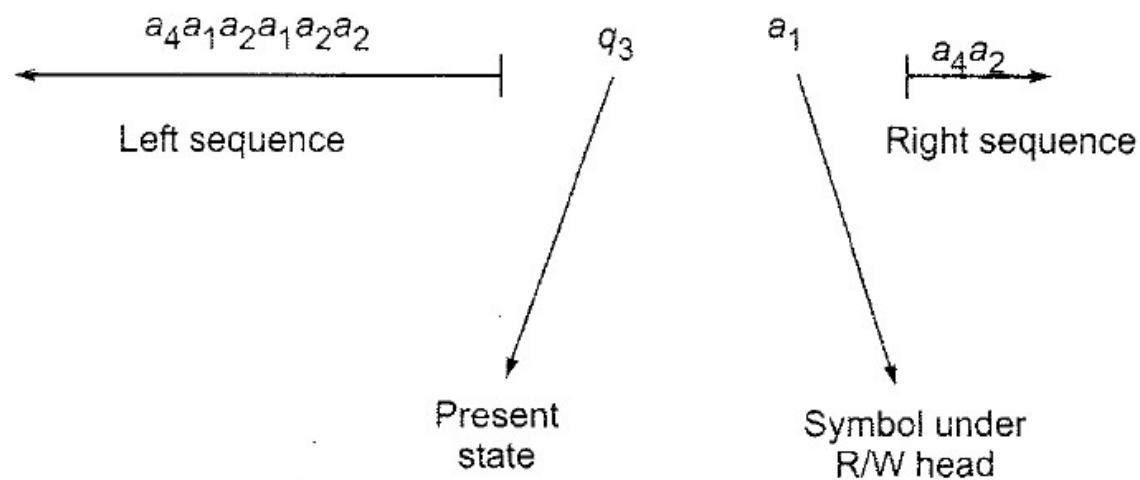
$\delta$  may not be defined for some elements of  $Q \times r$ .

## REPRESENTATION BY INSTANTANEOUS DESCRIPTIONS

**Definition 9.2** An ID of a Turing machine  $M$  is a string  $a\beta\gamma$ , where  $\beta$  is the present state of  $M$ , the entire input string is split as  $\alpha\gamma$ , the first symbol of  $\gamma$  is the current symbol  $a$  under the R/W head and  $\gamma$  has all the subsequent symbols of the input string, and the string  $\alpha$  is the substring of the input string formed by all the symbols to the left of  $a$ .



## Representation of ID



## Moves in a TM

Suppose  $\delta(q, x_i) = (p, y, L)$ .

$x_1 x_2 \dots x_{i-1} q x_i \dots x_n$

After processing  $x_i$ , the resulting ID is

$x_1 \dots x_{i-2} p x_{i-1} y x_{i+1} \dots x_n$

This change of ID is represented by

$x_1 x_2 \dots x_{i-1} q x_i \dots x_n \xrightarrow{\quad} x_1 \dots x_{i-2} p x_{i-1} y x_{i+1} \dots x_n$

If  $\delta(q, x_i) = (p, \bar{y}, R)$ , then the change of ID is represented by

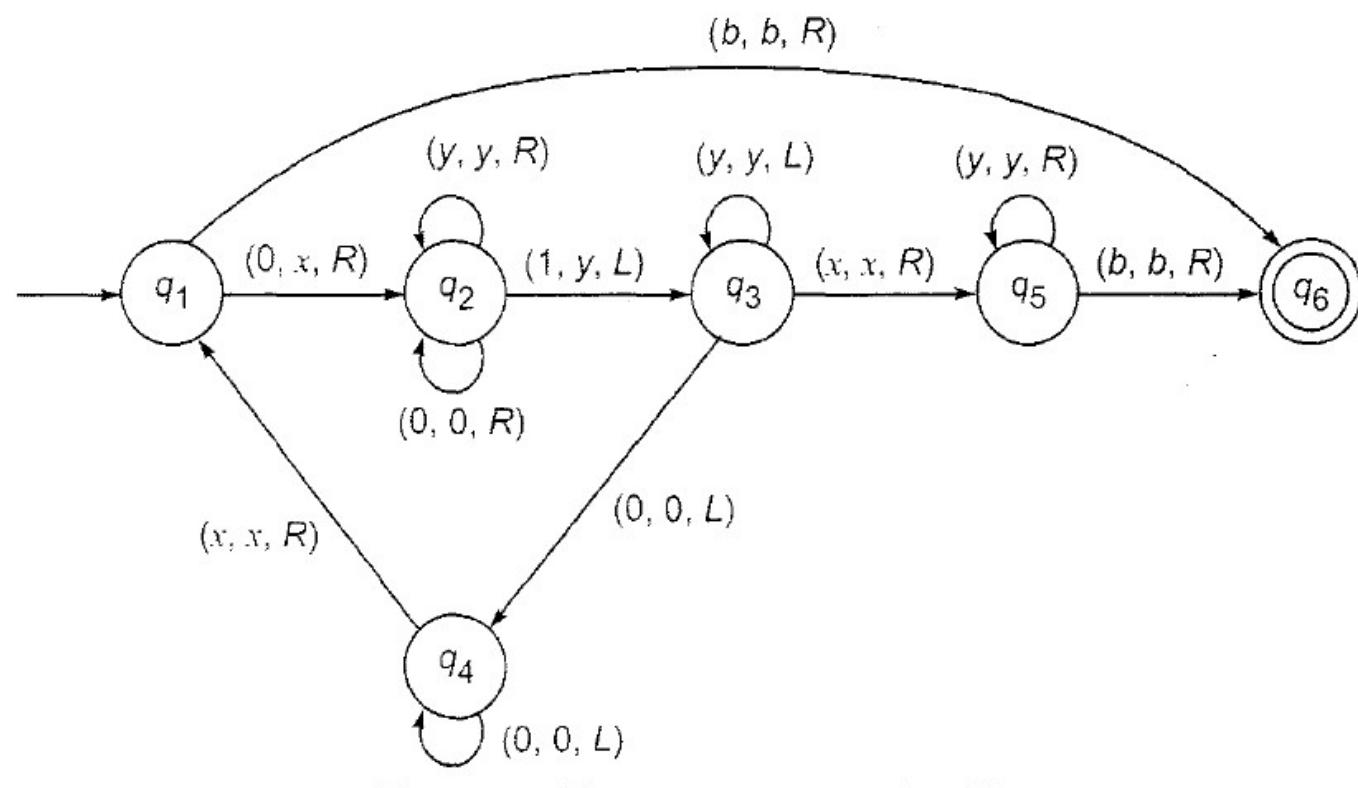
$x_1 x_2 \dots x_{i-1} q x_i \dots x_n \xrightarrow{\quad} x_1 x_2 \dots x_{i-1} \bar{y} p x_{i+1} \dots x_n$

## REPRESENTATION BY TRANSITION TABLE

Transition Table of a Turing Machine

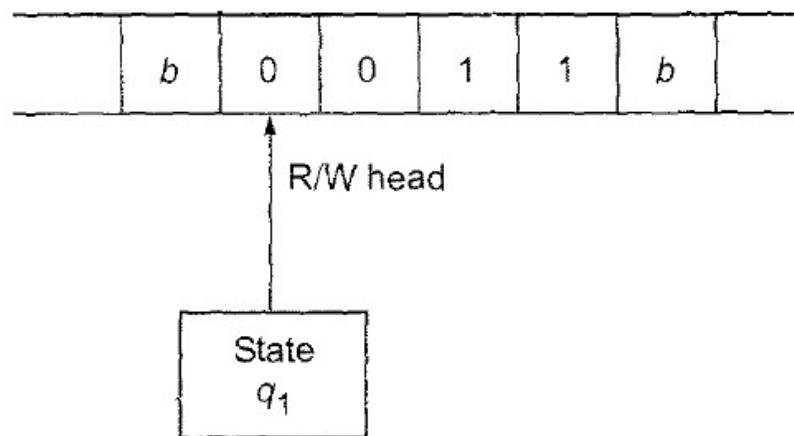
Present state	Tape symbol		
	b	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	
$q_2$	$bRq_3$	$0Lq_2$	$1Lq_2$
$q_3$		$bRq_4$	$bRq_5$
$q_4$	$0Rq_5$	$0Rq_4$	$1Rq_4$
$(q_5)$	$0Lq_2$		

## REPRESENTATION BY TRANSITION DIAGRAM

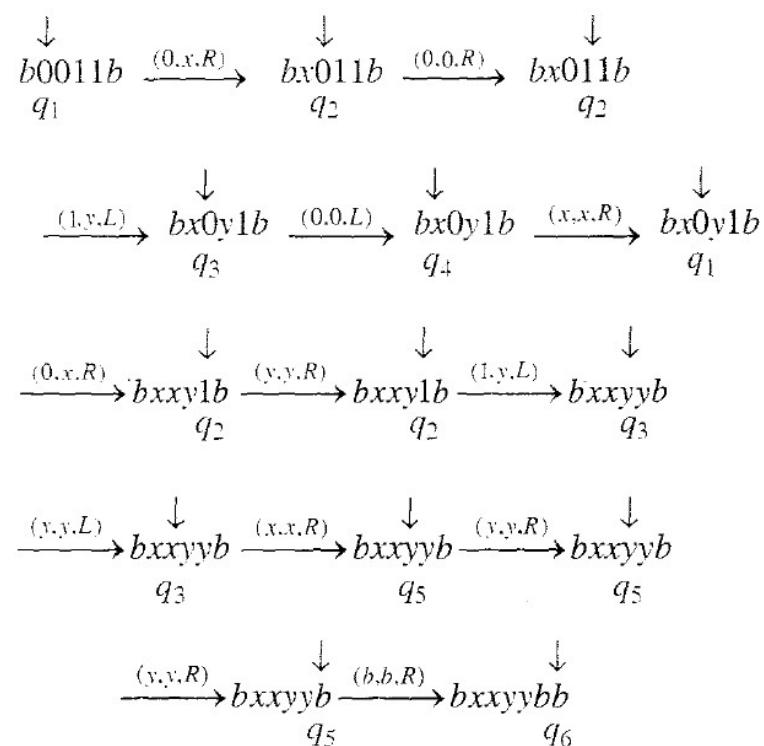


The change brought about by processing the symbol 0 can be represented as

$$\begin{array}{c} \downarrow \\ b0011b \\ q_1 \end{array} \xrightarrow{(0,x,R)} \begin{array}{c} \downarrow \\ bx011b \\ q_2 \end{array}$$



# Entire Computation Sequence of 0011



## 9.3 LANGUAGE ACCEPTABILITY BY TURING MACHINES

Let us consider the Turing machine  $M = (Q, \Sigma, \Gamma, \delta, q_0, b, F)$ . A string  $w$  in  $\Sigma^*$  is said to be accepted by  $M$  if  $q_0w \xrightarrow{*} \alpha_1 p \alpha_2$  for some  $p \in F$  and  $\alpha_1, \alpha_2 \in \Gamma^*$ .

$M$  does not accept  $w$  if the machine  $M$  either halts in a nonaccepting state or does not halt.

Consider the Turing machine  $M$  described by the transition table given in Table 9.2. 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
$\rightarrow q_1$	$xRq_2$				$bRq_5$
$q_2$	$0Rq_2$	$yLq_3$		$yRq_2$	
$q_3$	$0Lq_4$		$xRq_5$	$yLq_3$	
$q_4$	$0Lq_4$		$xRq_1$		
$q_5$			$yxRq_5$		$bRq_6$
( $q_6$ )					

$$(a) \ q_1 011 \xrightarrow{} xq_2 11 \xrightarrow{} q_3 xy1 \xrightarrow{} xq_5 y1 \xrightarrow{} xyq_5 1$$

As  $\delta(q_5, 1)$  is not defined,  $M$  halts; so the input string 011 is not accepted.

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

(b)  $q_10011 \vdash xq_2011 \vdash x0q_211 \vdash xq_30y1 \vdash q_4x0yl \vdash xq_10y1.$

$\vdash xxq_2y1 \vdash xxyq_21 \vdash xxq_3yy \vdash xq_3xxy \vdash xxq_5yy$

$\vdash xxyq_5y \vdash xxyyq_5b \vdash xxyybq_6$

$M$  halts. As  $q_6$  is an accepting state, the input string 0011 is accepted by  $M$ .

Present state	Tape symbol				
	0	1	x	y	b
$\rightarrow q_1$	$xRq_2$				$bRq_5$
$q_2$	$0Rq_2$	$yLq_3$		$yRq_2$	
$q_3$	$0Lq_4$		$xRq_5$	$yLq_3$	
$q_4$	$0Lq_4$		$xRq_1$		
$q_5$			<del><math>yRq_5</math></del>		$bRq_6$
$\circled{q_6}$					

(c)  $q_1001 \xrightarrow{} xq_201 \xrightarrow{} x0q_21 \xrightarrow{} xq_30y \xrightarrow{} q_4x0y$   
 $\xrightarrow{} xq_10y \xrightarrow{} xxq_2y \xrightarrow{} xxyq_2$

$M$  halts. As  $q_2$  is not an accepting state, 001 is not accepted by  $M$ .

Design a Turing machine over  $\{1, b\}$  which can compute a concatenation function over  $\Sigma = \{1\}$ . If a pair of words  $(w_1, w_2)$  is the input, the output has to be  $w_1w_2$ .

if  $w_1 = 11$ ,  $w_2 = 111$ ,

b	1	1	b	1	1	1	b
---	---	---	---	---	---	---	---

b	1	1	1	1	1	b	b
---	---	---	---	---	---	---	---

Input and output tapes.

The separating symbol  $b$  is found and replaced by 1.

The rightmost 1 is found and replaced by a blank  $b$ .

The R/W head returns to the starting position.

$$\begin{aligned}
 & q_011b111 \xrightarrow{} 1q_01b111 \xrightarrow{} 11q_0b111 \xrightarrow{} 111q_1111 \\
 \xrightarrow{} & 1111q_111 \xrightarrow{} 11111q_11 \xrightarrow{} 111111q_1b \xrightarrow{} 111111q_21b \\
 \xrightarrow{} & 1111q_31bb \xrightarrow{} 111q_311bb \xrightarrow{} 11q_3111bb \xrightarrow{} 1q_31111bb \\
 \xrightarrow{} & q_311111bb \xrightarrow{} q_3b11111bb \xrightarrow{} bq_f11111bb
 \end{aligned}$$

Present States	1	b
q0	1Rq0	1Rq1
q1	1Rq1	bLq2
q2	bLq3	
q3	1Lq3	bRqf
qf		

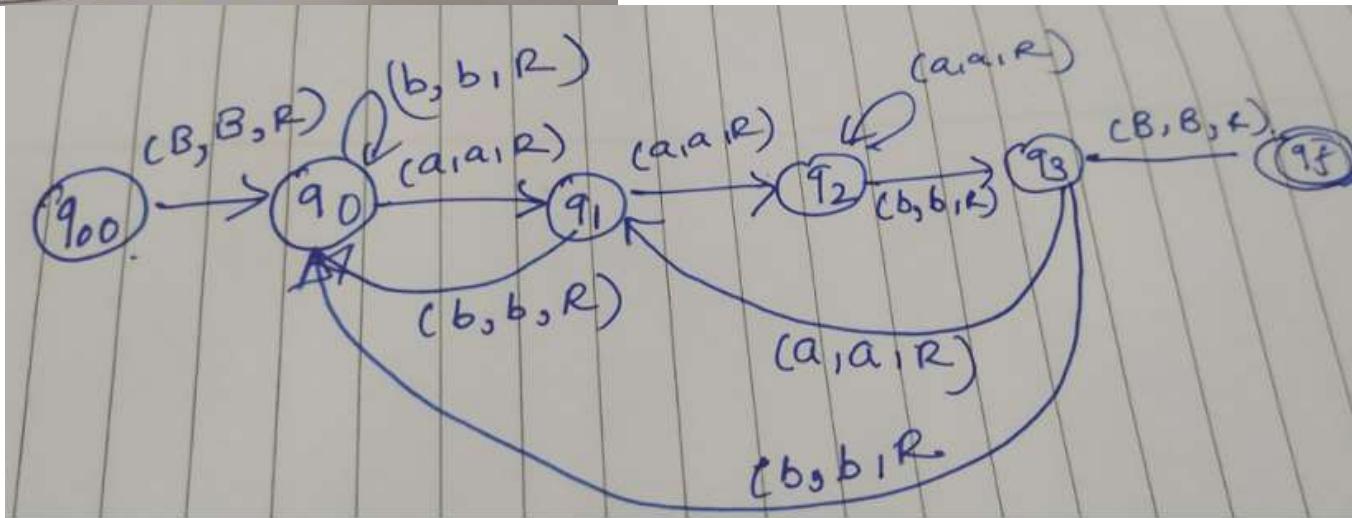
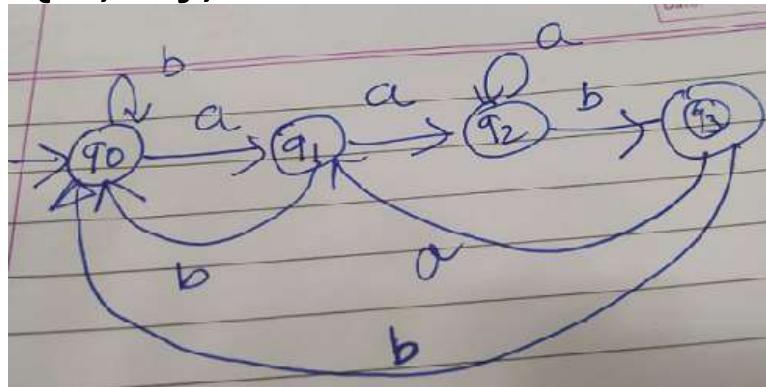
What is the computation Sequence for 1b1

For the input string  $1b1$ , the computation sequence is given as

$$\begin{aligned}
 q_01b1 &\xrightarrow{} 1q_0b1 \xrightarrow{} 11q_11 \xrightarrow{} 111q_1b \xrightarrow{} 11q_2b \xrightarrow{} 1q_31bb \\
 &\xrightarrow{} q_311bb \xrightarrow{} q_3b11bb \xrightarrow{} bq_f11bb.
 \end{aligned}$$

Present States	<b>1</b>	<b>b</b>
$q_0$	$1Rq_0$	$1Rq_1$
$q_1$	$1Rq_1$	$bLq_2$
$q_2$	$bLq_3$	
$q_3$	$1Lq_3$	$bRq_f$
$q_f$		

Example: Design a TM for a language L over  $\{a,b\}$ , which is ending with 'aab'.



- Validity testing: for the String ‘baab’
- $q_{00}BbaabB \vdash Bq_0baabB \vdash Bbq_0aabB \vdash Bbaq_1abB \vdash Bbaaq_2bB \vdash Bbaabq_3B \vdash BbaabBqf$

TM will halt at qf state and as qf is the final state, String ‘baab’ is accepted by the TM.

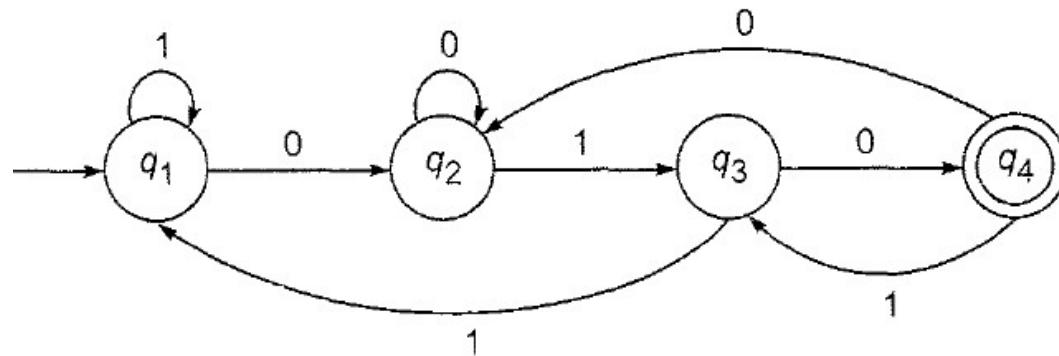
Validity testing: for the String ‘baba’

$q_{00}BbabaB \vdash Bq_0babaB \vdash Bbq_0abaB \vdash Bbaq_1baB \vdash Bbabq_0aB \vdash Bbabaq_1B$

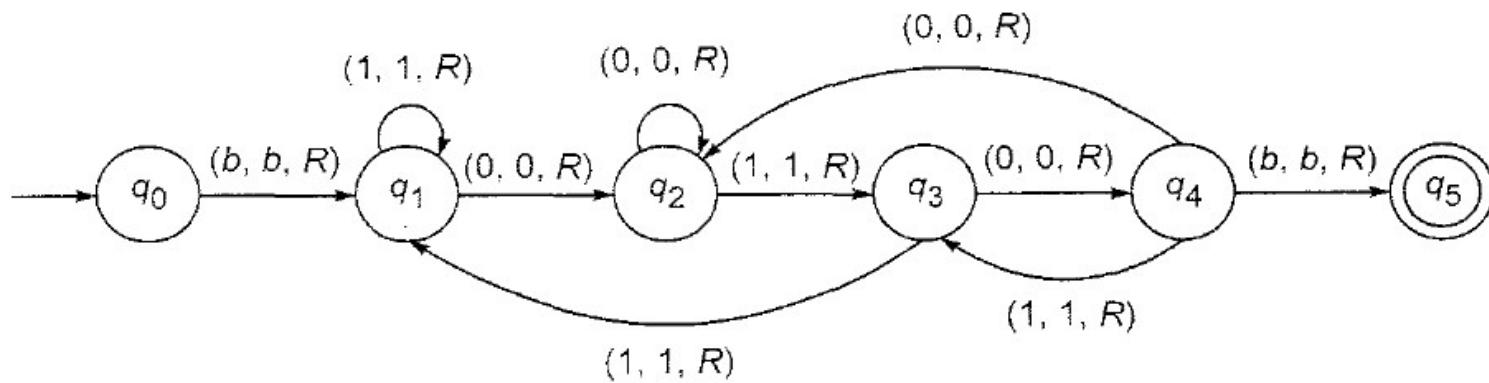
TM will halt at q1 state & as q1 is not final state String ‘baba’ is not accepted by TM.

Construct a TM to accept the set  $L$  of all strings over  $\{0,1\}$  ending with 010.

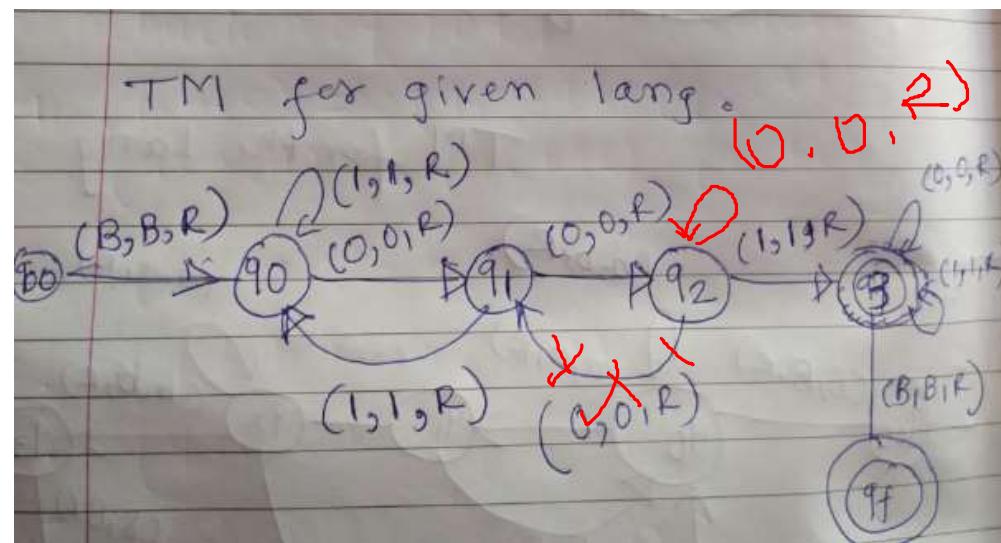
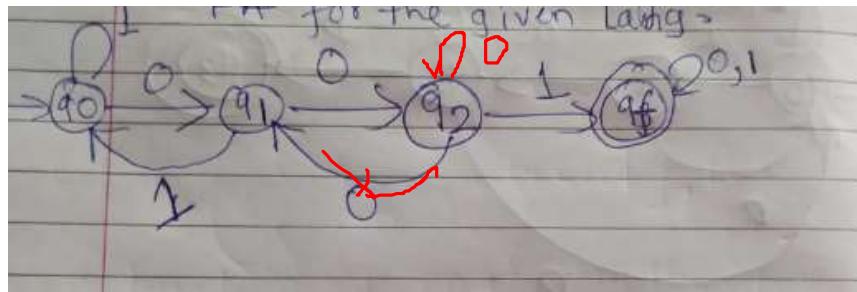
DFA for given language



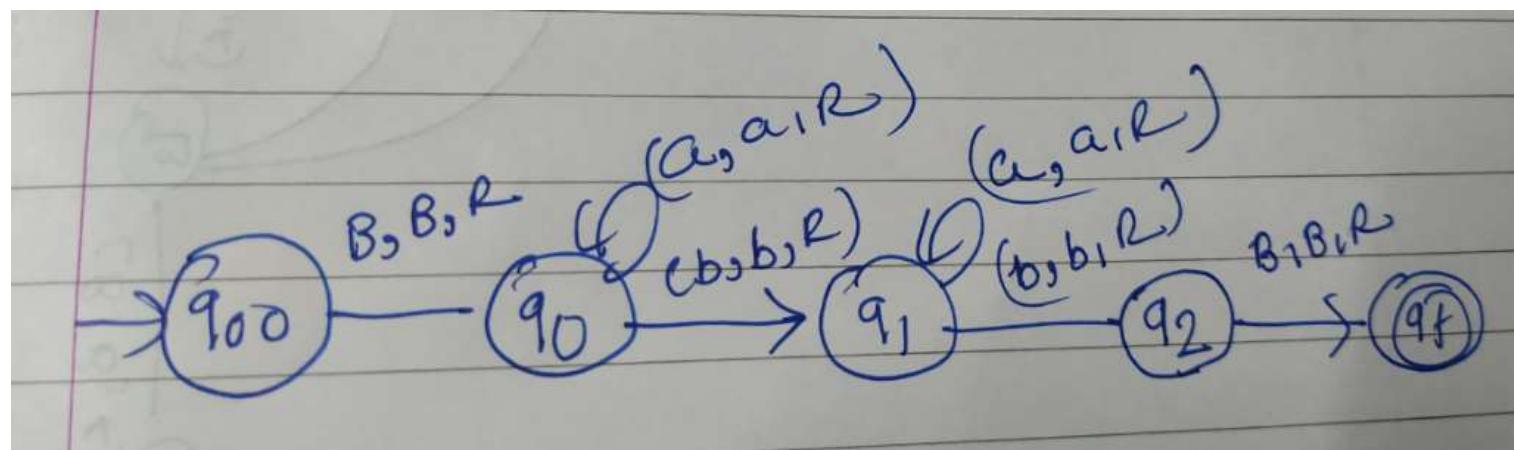
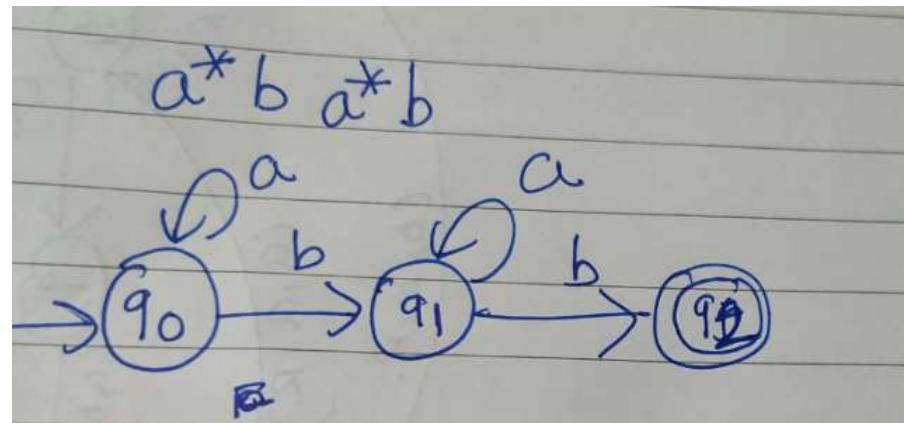
TM for given language



Example: Design a TM for a language  $L$  over  $\{0,1\}$ , containing the substring '001'.



Design a TM for language L over {a, b}



- Validity testing: for the String ‘1001’
- $q_{00}B1001 \xrightarrow{} Bq_0 1001 \xrightarrow{} B1q_0 001B \xrightarrow{} B10q_1 01B \xrightarrow{} B100q_2 1B \xrightarrow{} B1001q_3 B \xrightarrow{} B1001Bqf$

TM will halt at qf state and as qf is the final state, String ‘1001’ is accepted by the TM.

Validity testing: for the String ‘1010’

$q_{00}B1010B \xrightarrow{} Bq_0 1010B \xrightarrow{} B1q_0 010B \xrightarrow{} B10q_1 10B \xrightarrow{} B101q_0 0B \xrightarrow{} B1010q_1 B$

TM will halt at q1 state & as q1 is not final state String ‘1010’ is not accepted by TM.

Design a TM that accepts

$$\{0^n 1^n \mid n \geq 1\}.$$

- (a) If the leftmost symbol in the given input string  $w$  is 0, replace it by  $x$  and move right till we encounter a leftmost 1 in  $w$ . Change it to  $y$  and move backwards.
- (b) Repeat (a) with the leftmost 0. If we move back and forth and no 0 or 1 remains, move to a final state.
- (c) For strings not in the form  $0^n 1^n$ , the resulting state has to be nonfinal.

Design a TM for  $\{a^n b^n \mid n \geq 1\}$

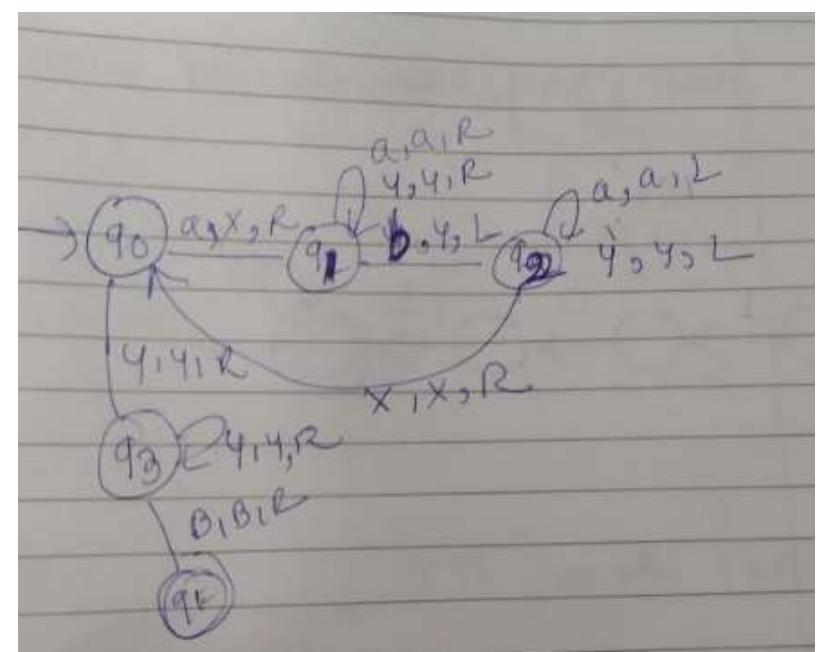
Design Strategy:

Replace leftmost 0 by 'x' TM changes state from  $q_0$  to  $q_1$  go to Right.

x	x	x	y	y	y
---	---	---	---	---	---

States	a	b	x	y	b
q0	xRq1			yRq3	
q1	aRq1	yLq2		yRq1	
q2	aLq2		xRq0	yLq2	
q3				yRq3	bRqf
qf	-	-	-	-	-

Transition table of TM



Transition diagram of TM

# Validity testing ‘ab’

State s	a	b	x	y	B
q0	xRq1			yRq3	
q1	aRq1	yLq2		yRq1	
q2	aLq2		xRq0	yLq2	
q3				yRq3	BRqf
qf	-	-	-	-	-

$q_0 a b \xrightarrow{a} q_1 b \xrightarrow{b} q_2 x y$   
 $\xrightarrow{x} q_0 y \xrightarrow{y} q_3 B \xrightarrow{B} q_f$

# Validity Testing ‘aabb’

State s	a	b	x	y	B
q0	$xRq1$			$yRq3$	
q1	$aRq1$	$yLq2$		$yRq1$	
q2	$aLq2$		$xRq0$	$yLq2$	
q3				$yRq3$	$BRqf$
qf	-	-	-	-	-

9<sub>0</sub>aabb  $\xrightarrow{aRq_1}$  a<sub>0</sub>q<sub>1</sub>abb  $\xrightarrow{aRq_1}$  a<sub>0</sub>q<sub>1</sub>bb  
 a<sub>0</sub>q<sub>1</sub>ab  $\xrightarrow{aRq_2}$  a<sub>2</sub>q<sub>1</sub>ab  
 a<sub>2</sub>q<sub>1</sub>ab  $\xrightarrow{aRq_0}$  a<sub>2</sub>q<sub>0</sub>ab — a<sub>2</sub>q<sub>1</sub>ab  
 a<sub>2</sub>q<sub>0</sub>ab  $\xrightarrow{aRq_1}$  a<sub>2</sub>q<sub>1</sub>ab  
 a<sub>2</sub>q<sub>1</sub>ab  $\xrightarrow{aRq_2}$  a<sub>2</sub>q<sub>2</sub>ab  
 a<sub>2</sub>q<sub>2</sub>ab  $\xrightarrow{aRq_0}$  a<sub>2</sub>q<sub>0</sub>ab  
 a<sub>2</sub>q<sub>0</sub>ab  $\xrightarrow{aRq_1}$  a<sub>2</sub>q<sub>1</sub>ab  
 a<sub>2</sub>q<sub>1</sub>ab  $\xrightarrow{aRq_2}$  a<sub>2</sub>q<sub>2</sub>ab  
 a<sub>2</sub>q<sub>2</sub>ab  $\xrightarrow{aRq_0}$  a<sub>2</sub>q<sub>0</sub>ab

# Validity Testing ‘aaabbb’

State s	0	1	x	y	B
q0	$xRq1$			$yRq3$	
q1	$0Rq1$	$yLq2$		$yRq1$	
q2	$0Lq2$		$xRq0$	$yLq2$	
q3				$yRq3$	$BRqf$
qf	-	-	-	-	-

# Validity Testing 'abb'

State s	a	b	x	y	B
q0	xRq1			yRq3	
q1	0Rq1	yLq2		yRq1	
q2	0Lq2		xRq0	yLq2	
q3				yRq3	BRqf
qf	-	-	-	-	-

"abb"  
~~q0abb~~ — ~~xq1bb~~ — ~~q2xyb~~  
~~— xq0yb~~ — ~~000~~ ~~xq3b~~

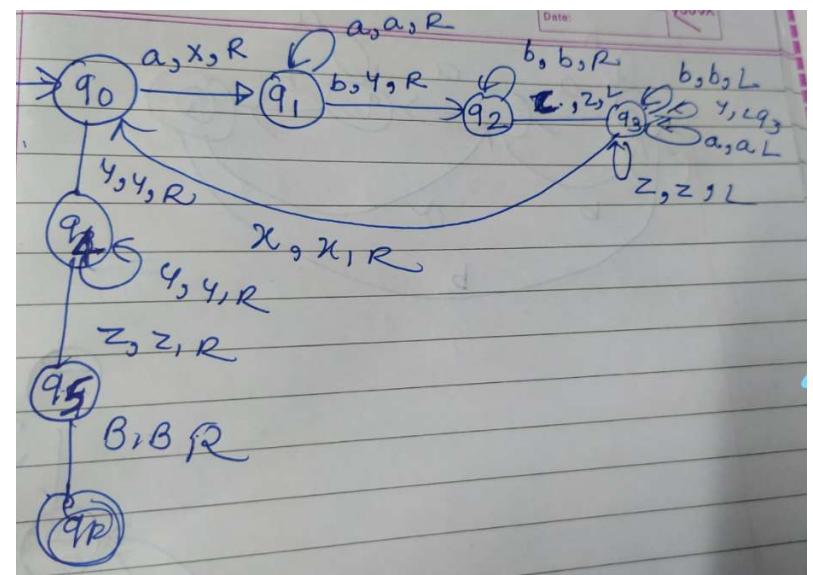
Design a TM for a language  $L = a^n b^n c^n$   
where  $n \geq 1$ .

1. Replace leftmost  $a$  by ' $x$ ' Change the state, replace leftmost  $b$  by ' $y$ ' change the state, replace leftmost  $c$  by ' $z$ ' change the state and move backward(i. e in left direction).
2. Repeat 1 with remaining ' $a$ ', ' $b$ ' and ' $c$ '. TM will move back and forth.
3. TM will move to a final state only when number of ' $a$ ', ' $b$ ' and ' $c$ ' are same.
4. TM will move to a non final state when number of ' $a$ ', ' $b$ ' and ' $c$ ' are not same.

a	a	b	b	c	c
---	---	---	---	---	---

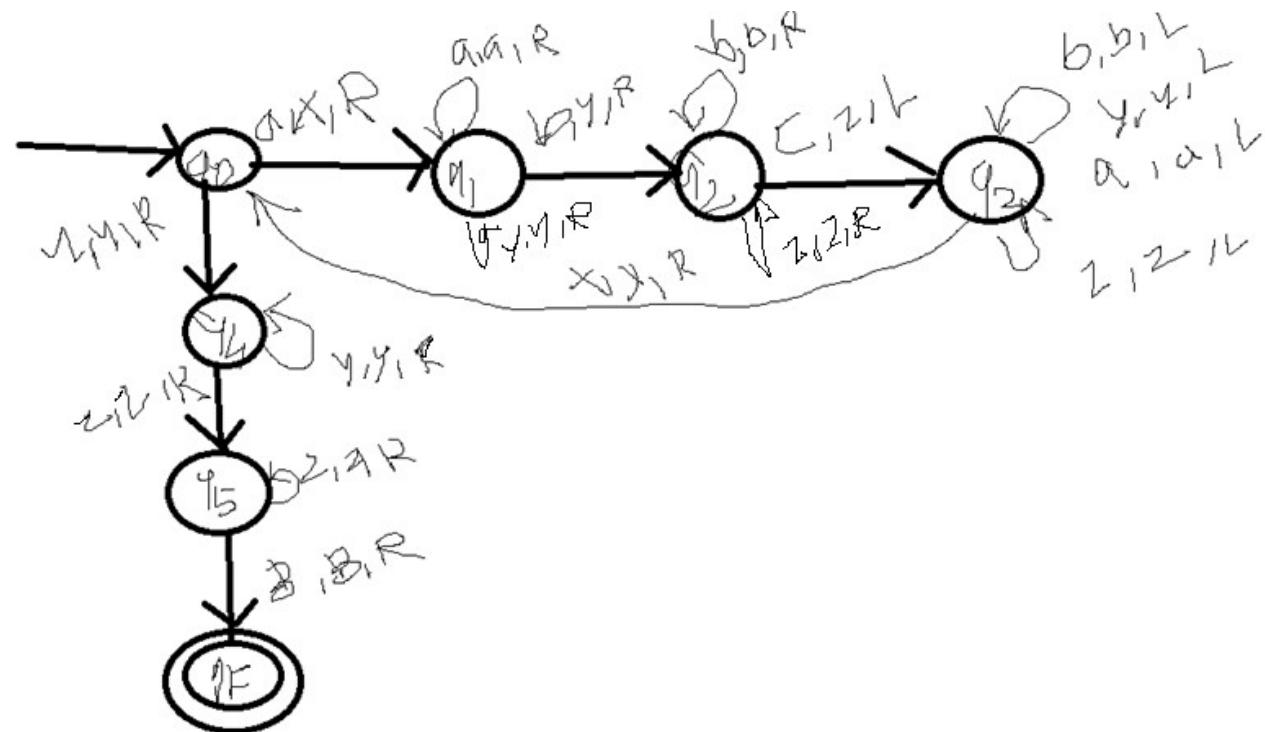
States	a	b	c	x	y	z	B
q0	xRq1				yRq4		
q1	aRq1	yRq2			yRq1		
q2		bRq2	zLq3			zRq2	
q3	aLq3	bLq3		xRq0	yLq3	zLq3	
q4					yRq4	zRq5	
q5						zRq5	BRqf
qf							

Transition table of TM

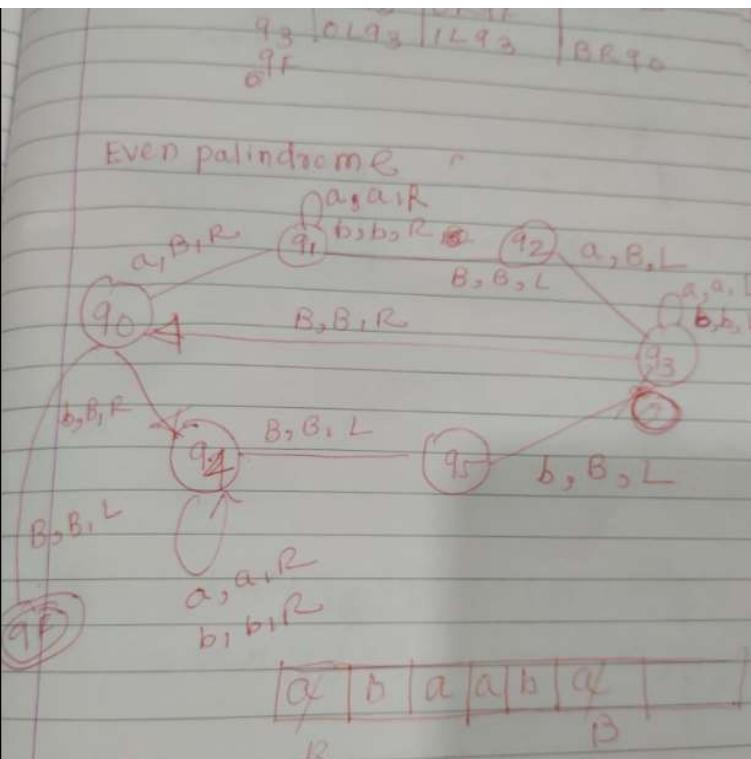


Transition diagram of TM

TM transition diagram for a language  $L = a^n b^n c^n$



# Design a TM for accepting a even palindrome over {a,b}



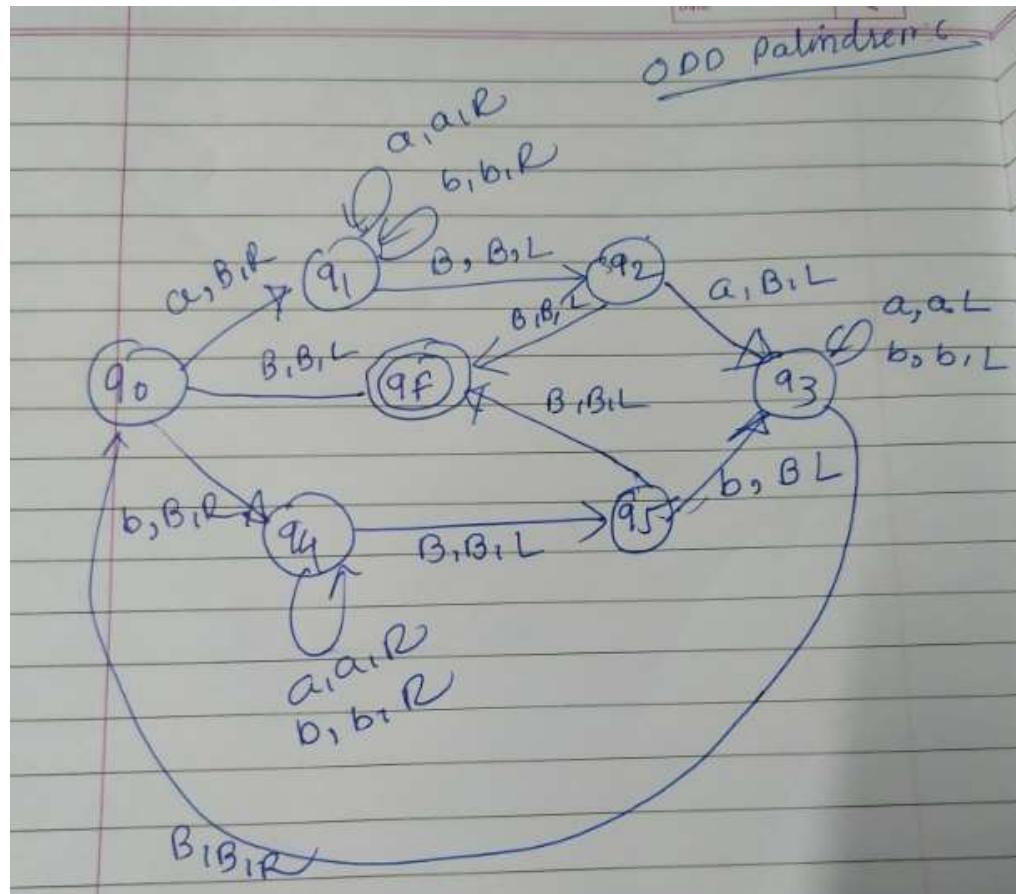
a	b	a	a	b	a
---	---	---	---	---	---

P. State	a	b	B
q0	BRq1		BLqf
q1	aRq1	bRq1	BLq2
q2	BLq3		
q3	aLq3	bLq3	BRq0
qf			

Transition Table of TM

Transition diagram of TM

Design a TM for accepting a ODD length palindrome over {a,b}





# Design a TM for subtraction( $m-n$ )(if $m>n$ )

M = 0000

N=00

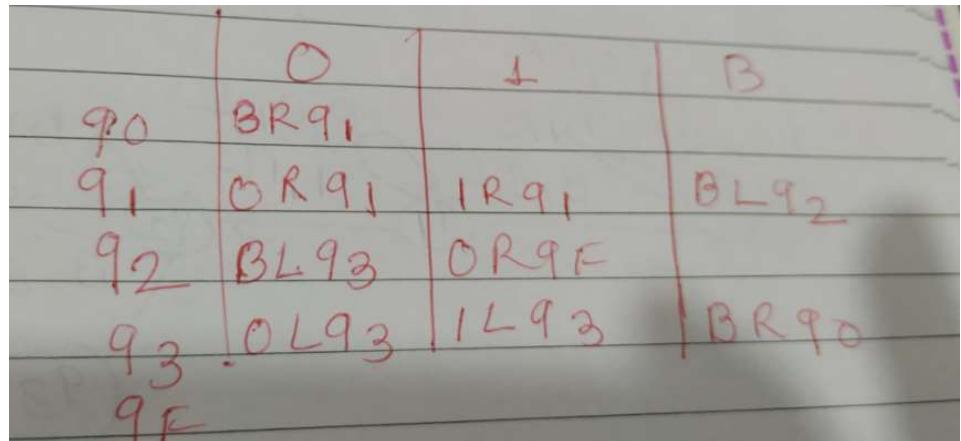
0	0	0	0	1	0	0
---	---	---	---	---	---	---

Design Strategy: 1 : Replace leftmost 0 by 'B' traverse in the right direction until rightmost 0 is found

2: Replace rightmost 0 by by 'B' traverse in the left direction until 'B' is found.

Repeat above steps until 'N' gets over.

And finally replace 1 by '0' & move to final state.



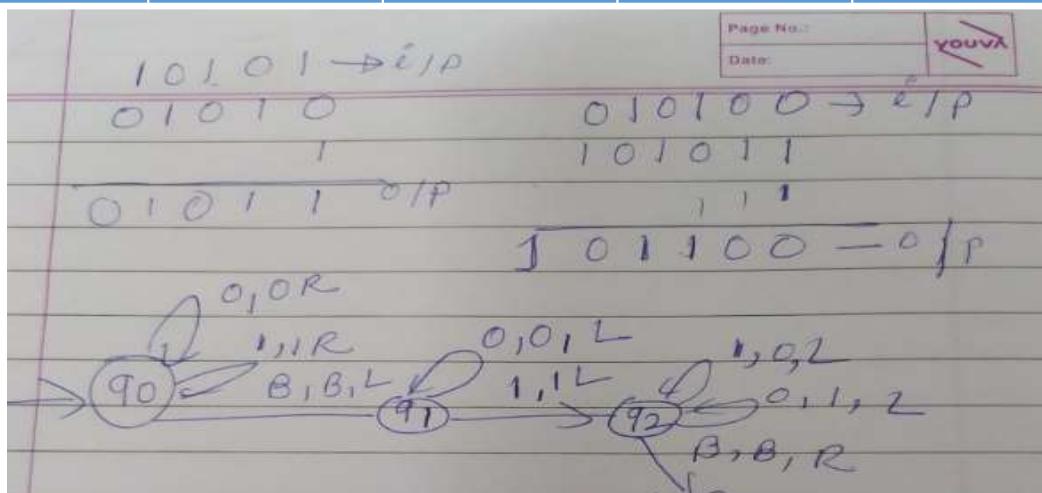
P. State	0	1	B
q0			
q1			
q2			
q3			
qf			

# Design a TM for a 2's complement of a given number.

Design Strategy:

- Traverse the input from left to right.
- From right direction start scanning towards left direction & in the path keep symbol as it is until first '1' is encountered, keep that '1' also as it is after that
- Replace every '0' by 1 and '1' by 0. till leftmost symbol is encountered.

1	0	1	1	0	0
---	---	---	---	---	---



P. State	0	1	B
$q_0$	$0Rq_0$	$1Rq_0$	$BLq_1$
$q_1$	$0Lq_1$	$1Lq_2$	
$q_2$	$1Lq_2$	$0Lq_2$	$BRqf$
$q_f$			

- Types of Turing Machine
- **Multi-dimensional Tape Turing Machine**
- **Multi-head Turing Machine**
- Nondeterministic TM
- Multi Tape TM
- Off line Turing Machine

# Multi-dimensional Tape Turing Machine:

- It has multi-dimensional tape where head can move any direction that is left, right, up or down
- Multi dimensional tape Turing machine can be simulated by one-dimensional Turing machine

## MULTIDIMENSIONAL TURING MACHINE

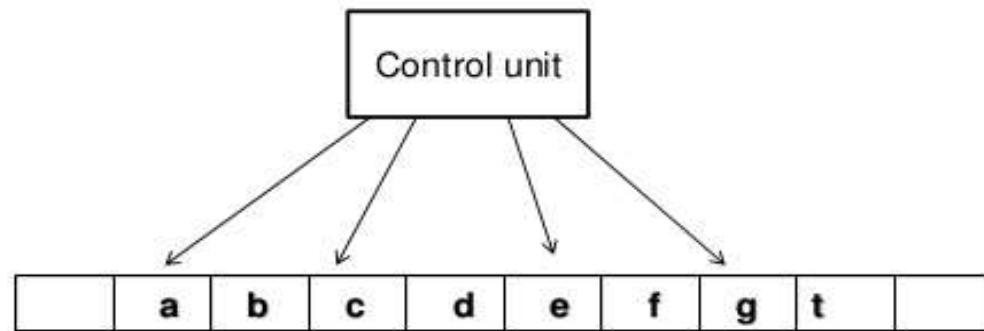
- A Multidimensional TM has a multidimensional tape.  
For example, a two-dimensional Turing machine would read and write on an infinite plane divided into squares, like a checkerboard.
- For a two- Dimensional Turing Machine transaction function define as:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$$

- . **Multi-head Turing Machine:**
- A multi-head Turing machine contain two or more heads to read the symbols on the same tape.
- In one step all the heads sense the scanned symbols and move or write independently.
- Multi-head Turing machine can be simulated by single head Turing machine.

## MULTIHEAD TURING MACHINE

- Multihead TM has a number of heads instead of one.
- Each head independently read/ write symbols and move left / right or keep stationary.



Every language accepted by Multi head TM is accepted by single head TM

# Nondeterministic TM

**Definition 9.5** A nondeterministic Turing machine is a 7-tuple  $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$  where

1.  $Q$  is a finite nonempty set of states
2.  $\Gamma$  is a finite nonempty set of tape symbols
3.  $b \in \Gamma$  is called the blank symbol
4.  $\Sigma$  is a nonempty subset of  $\Gamma$ , called the set of input symbols. We assume that  $b \notin \Sigma$ .
5.  $q_0$  is the initial state
6.  $F \subseteq Q$  is the set of final states
7.  $\delta$  is a partial function from  $Q \times \Gamma$  into the power set of  $Q \times \Gamma \times \{L, R\}$ .

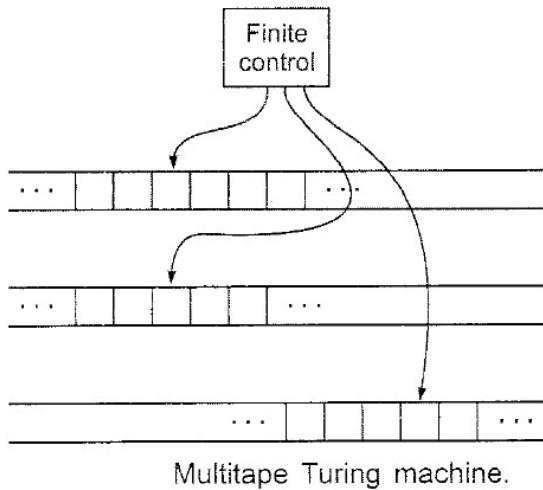
**Note:** If  $q \in Q$  and  $x \in \Gamma$  and  $\delta(q, x) = \{(q_1, y_1, D_1), (q_2, y_2, D_2), \dots, (q_n, y_n, D_n)\}$  then the NTM can chose any one of the actions defined by  $(q_i, y_i, D_i)$  for  $i = 1, 2, \dots, n$ .

**Theorem 9.3** If  $M$  is a nondeterministic TM, there is a deterministic TM  $M_1$  such that  $T(M) = T(M_1)$ .

# Multi Tape TM

In a typical move:

- (i)  $M$  enters a new state.
- (ii) On each tape, a new symbol is written in the cell under the head.
- (iii) Each tape head moves to the left or right or remains stationary. The heads move independently; some move to the left, some to the right and the remaining heads do not move.



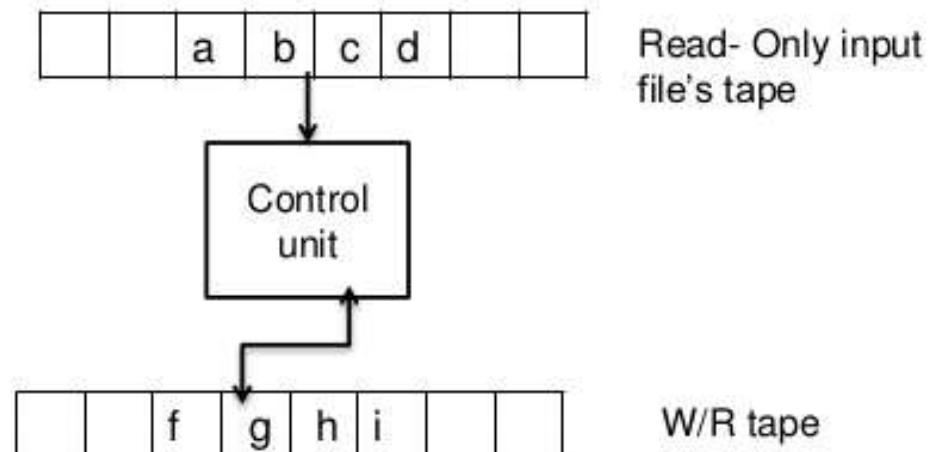
Multitape Turing machine.

**Theorem 9.1** Every language accepted by a multitape TM is acceptable by some single-tape TM (that is, the standard TM).

## OFF- LINE TURING MACHINE

- An *Offline Turing Machine* has two tapes

1. *One tape is read-only* and contains the input
2. The other is read-write and is initially blank.



# Universal TM

- In computer science, a **universal Turing machine (UTM)** is a Turing machine that simulates an arbitrary Turing machine on arbitrary input
- UTM essentially achieves this by reading both the description of the machine to be simulated as well as the input to that machine from its own tape
- Alan Turing introduced the idea of such a machine in 1936–1937

- A Turing machine is said to be universal Turing machine if it can accept:
  - The input data, and
  - An algorithm (description) for computing.
- This is precisely what a general purpose digital computer does.
- A digital computer accepts a program written in high level language.
- Thus, a general purpose Turing machine will be called a universal Turing machine if it is powerful enough to simulate the behavior of any digital computer, including any Turing machine itself.

- More precisely, a UTM can simulate the behavior of an arbitrary Turing machine over any set of input symbols
- Thus, it is possible to create a single machine that can be used to compute any computable sequence
- Designing a general purpose Turing machine is a more complex task
- The model of a UTM is considered to be a theoretical breakthrough that led to the concept of stored programmer computing device
- This principle is considered to be the origin of the idea of a stored-program computer used by John von Neumann in 1946 for the "Electronic Computing Instrument" that now bears von Neumann's name: the von Neumann architecture

- Deterministic and the nondeterministic finite automata behave in the same way in so far as acceptability of languages is concerned
- The same is the case with Turing machines
- But the behaviour of deterministic and nondeterministic pushdown automata is different
- A context-free language is said to be deterministic if it is accepted by a deterministic pushdown automaton

# HW

- 1.  $a^*b\ a^*b$
- 2.  $a^n\ b^n\ c^n$
- 1's complement of a binary number
- ODD palindrome TM