Turing Machine :-

⇒ Turing machine is an abstract machine developed by an englich mathematican Alan turing in 1936. The model of computation provides a theorifical foundation for modern computers.

⇒ A turing machine will have ;

- A finite set of alphabet.
- A finite set of states.
- A linear tape which is potentially infinite to both end.

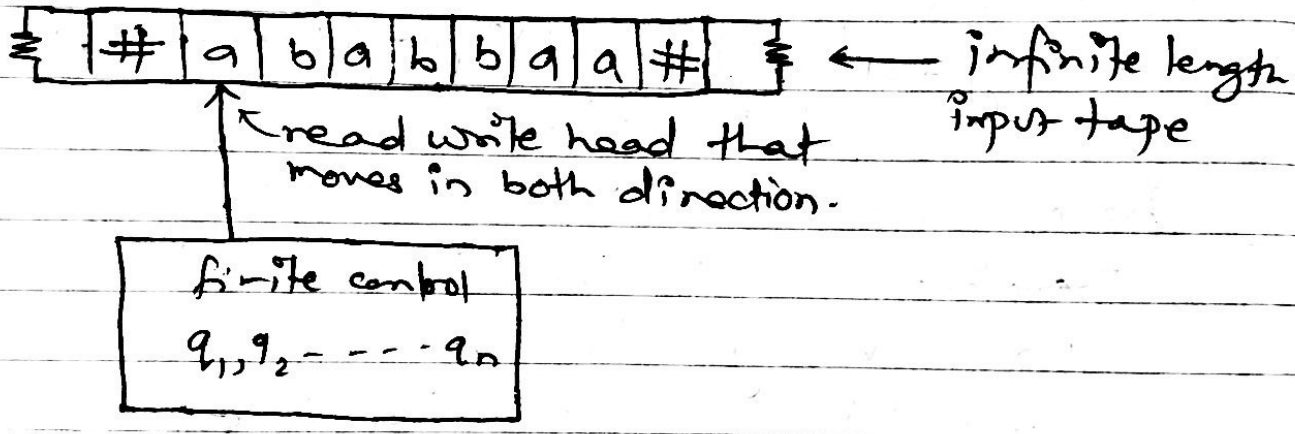


fig:- Block diagram of TM.

⇒ Turing machine consist of infinite length i/p tape, read/write head and finite control.

⇒ Initially all cells of input tape are labelled with special symbol called blank symbol (#).

Scanned with CamScanner

⇒ The head of TM can read from the i/p tape & can write to the input tape as well.

⇒ Further, the read/write head of TM can move in both direction either, from left to right or from right to left.

⇒ Since, TM can write the symbols to the input tape, it supports the feature of storage & bi-directional read/write head can again read the stored symbol so that it can be used in further processing.

⇒ Due to these features, TM is said to be functionally stronger than FA & PDA.

⇒ A single move of turing machine is function of the state of TM and current tape symbol and it consists of 3 things;

- Replacing the symbol in the current square by another, possibly different symbol.
- Moving the tape head one square right or left or leaving it where it is.
- Moving from current state to another, possibly different state.

Formal Description of TM:-

A turing machine TM can be defined by 7-tuples,

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

Where,

Q = finite set of states of the finite control.

Σ = finite set of input symbols.

Scanned with CamScanner

Date: Page:

Γ = the complete set of tape symbols,
Σ is always subset of Γ.
q_0 = Initial state, $q_0 \in Q$.
B(#) = blank symbol ; $B \in \Gamma$ but $B \notin \Sigma$.
F = Set of final states.
δ = transition function defined by,

$$Q \times \Gamma \longrightarrow Q \times \Gamma \times (R, L, \bullet)$$

where, R, L, • is the direction of movement of head. (Right or left ~~or stationary~~)

i.e. $\delta(q, X) = \delta(P, Y, D)$

TM state, current tape symbol, Next state, movement, Replacing tape symbol X with Y.

eg:- $(q_1, 0) \longrightarrow (q_2, 1, R)$

This moves tells us that, whenever we are in state q_1 and read '0' from input tape then the state is changed to q_2, '0' is replaced by '1' and next symbol to read is the symbol in the right of '0' in the input tape.

Representation of TM:-

1. Transition diagram
2. Transition table
3. Instantaneous Description (ID).

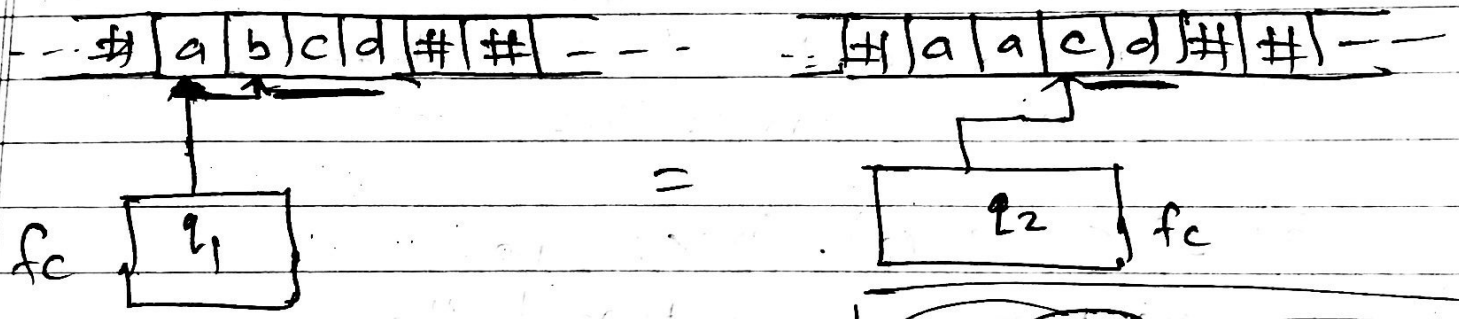
Scanned with CamScanner

Instantaneous description (ID) of TM :-

⇒ The configuration of a TM is described by instantaneous description of TM.

⇒ A string $X_1 X_2 ---- X_{i-1}\, q\, X_i\, X_{i+1} ---- X_n$ represents the ID of TM in which,

- q is the state of TM.
- The tape head scanning the ith symbol from the left.
- $X_1 X_2 ---- X_n$ is the portion of tape betn the leftmost or rightmost non-blank.



$\delta(q_1, b) = (q_2, a, R)$

- present state
- current symbol under r/w head
- next state
- replace b by a
- move head one position right

what if L ? ⊢ $q q_2 acd$ (L)

from q_1 state changes to q_2

$I_1 \vdash^{*}_{M} I_2$

$a q_1 bcd \vdash_M a a q_2 cd$

- remaining i/p tape symbol
- replace 'b' by 'a'
- move one position right

Scanned with CamScanner

Moves of TM:-

The moves of TM, $M = (Q, \Sigma, \Gamma, \delta, q_0, BF)$ is described by the notation $\vdash$, "yield", for single move and by $\vdash^*$ for zero or more moves.

a) for $\delta(q, X_i) = (p, Y, L)$

go 1 step left replace X_i by Y

q change to p

i.e. next move is leftward then,

$X_1 X_2 \ldots X_{i-1}\, q\, X_i\, X_{i+1} \ldots X_n \vdash$

$X_1 X_2 \ldots X_{i-2}\, p\, X_{i-1}\, Y\, X_{i+1} \ldots X_n$

replaces the change of state from q to p and the replacement of symbol X_i with Y and then the head is positioned at $i-1$.

(next scan is X_{i-1})

b) If $\delta(q, X_i) = (p, Y, R)$ i.e.

go one step right

q change to p

replace X_i by Y

next move is rightward then,

$X_1 X_2 \ldots X_{i-1}\, q\, X_i\, X_{i+1} \ldots X_n \vdash$

$X_1 X_2 \ldots X_{i-1}\, Y\, p\, X_{i+1} \ldots X_n$,

which represent that the symbol X_i is replaced with Y and head has moved to cell $i+1$ with change in state from q to P.

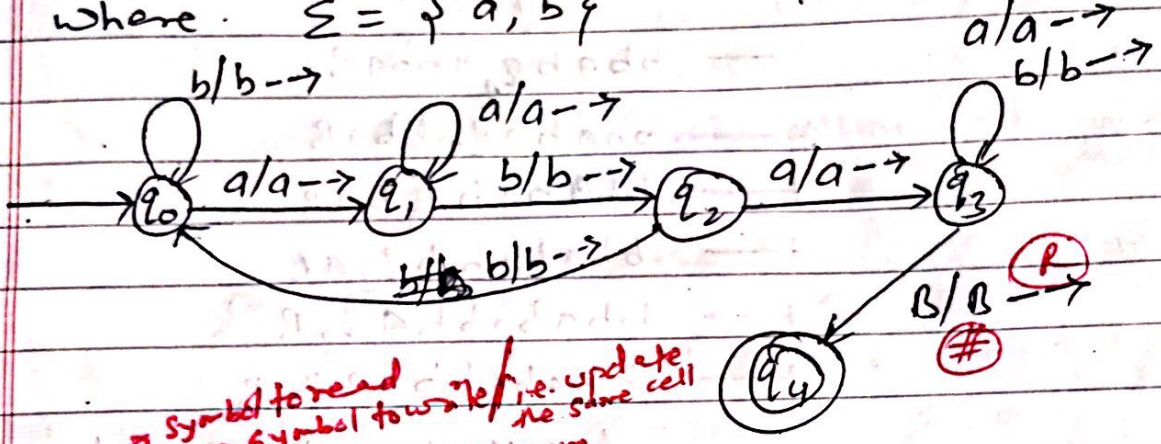
Scanned with CamScanner

Q. Design a TM that accepts the language of all string which contain aba as a substring.

Soln:- Let, the TM be

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

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



a/a→ : symbol to read / symbol to write / i.e. update the same cell / direction to move.

fig:- Transition diagram.

- If we don't want to update the cell then just write the same symbol // eg 1/1→

Then,

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$q_0 = q_0$

$F = q_4$

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

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

δ can be

State	a	b	B
q_0	(q_1, a, R)	(q_0, b, R)	–
q_1	(q_1, a, R)	(q_2, b, R)	–
q_2	(q_3, a, R)	(q_0, b, R)	–
q_3	(q_3, a, R)	(q_3, b, R)	(q_4, B, R)
q_4	–	–	–

fig:- Transition table

Scanned with CamScanner

Check for $w = bbaabbba$

$q_0 bbabbabbaB \vdash bq_0babbabbaB$

$\vdash bbq_0abbabbaB$

$\vdash bbaq_1bbabbaB$

$\vdash bbabq_2abbaB$

$\vdash bbabaq_3bbaB$

$\vdash bbabaq_3bbaB$

$\vdash bbababq_3aB$

$\vdash bbababbaq_3B$

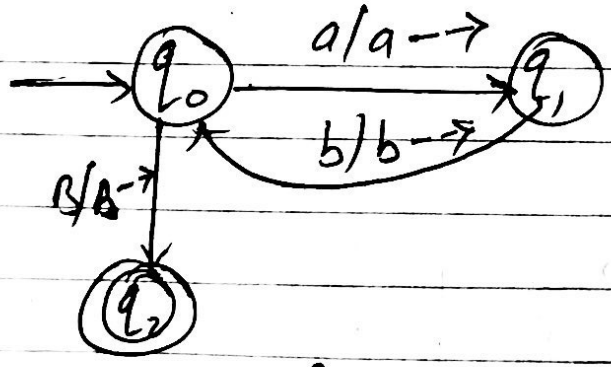
$\vdash bbababbaBq_4B$

Halt and accepted

Q Design a TM for the language $L = \{(ab)^n : n > 0\}$.

Soln: Let TM be

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



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

check for string $w = BabaBB$

⇒ The acceptance of abab by the TM, can be described by the following sequence of moves.

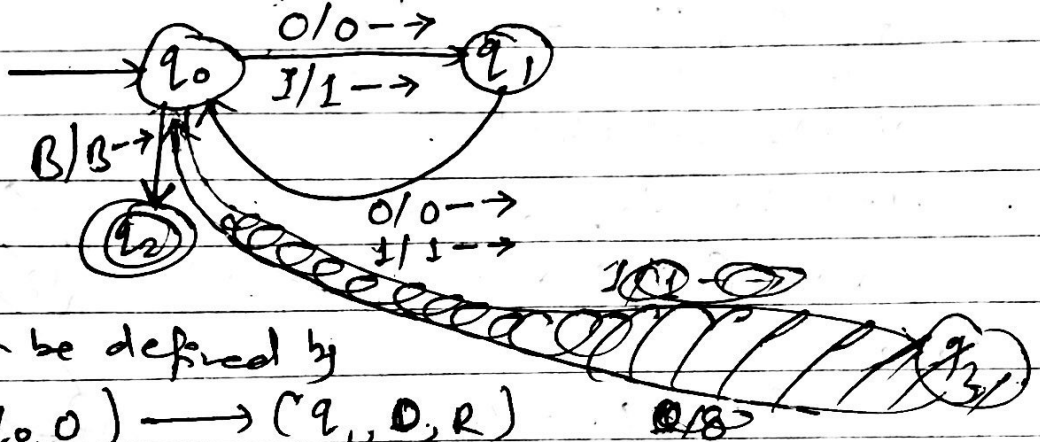
Scanned with CamScanner

Date: _____ Page: _____

$q_0 abab B \vdash a q_1 bab B$
$\vdash ab q_0 ab B$
$\vdash aba q_1 b B$
$\vdash abab q_0 B$
$\vdash abab B q_2 B$ Halt & accepted.

Q Design a TM that accepts all the strings of even length over $\Sigma = \{0, 1\}$

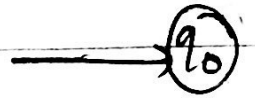
Soln.



δ can be defined by

$\delta(q_0, 0) \longrightarrow (q_1, 0, R)$
$\delta(q_0, 1) \longrightarrow (q_1, 1, R)$
$\delta(q_1, 0) \longrightarrow (q_0, 0, R)$
$\delta(q_1, 1) \longrightarrow (q_0, 1, R)$
$\delta(q_0, B) \longrightarrow (q_2, B, R)$ → \$ or N in such cases

Check $w = 11010|$



Scanned with CamScanner

Date: Page:

Q Write the TM that erases all the symbols of given string over $\Sigma = \{a, b\}$

Soln:

δ consists of

$\delta(q_0, a) \longrightarrow (q_1, B, R)$

$\delta(q_0, b) \longrightarrow (q_1, B, R)$

$\delta(q_1, a) \longrightarrow (q_1, B, R)$

$\delta(q_1, b) \longrightarrow (q_1, B, R)$

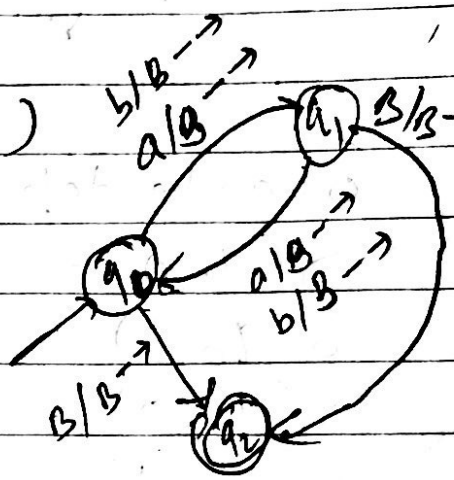
$\delta(q_1, B) \longrightarrow (q_2, B, R)$

$\delta(q_0, B) \longrightarrow (q_2, B, R)$

Let $w = ababab$

TM be defined as

$TM = \{Q, \Sigma, \delta, q_0, F, \Gamma, B\}$



where,

$Q = \{q_0, F\}$

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

q_0, is initial state.

$F = q_2$

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

B = blank symbol.

TM operation as,

$q_0 ababab \vdash B q_1 babab$

$\vdash BB q_1 abab$

$\vdash BBB q_1 bab$

$\vdash BBBB q_1 ab$

$\vdash BBBBB q_1 b$

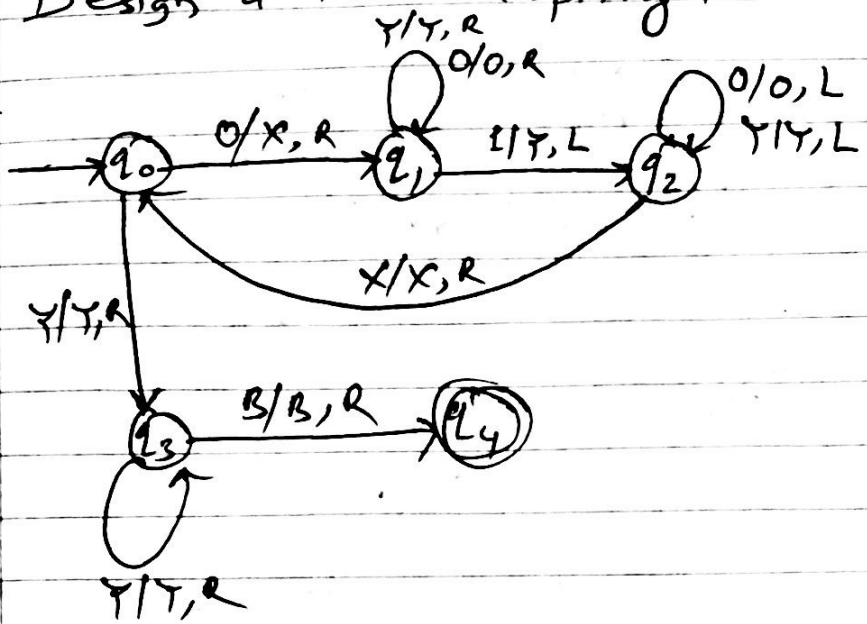
$\vdash BBBBBB q_1 B$

$\vdash BBBBBBB q_2 B$

Scanned with CamScanner

Q Design a TM accepting $\{0^n 1^n \mid n \geq 1\}$

Soln:-



n=1
01
n=2, 0011
n=3, 000111
Let n=3.
B0OOYIIB
BXOOYIIB
BXXOYYIB
BXXXYYYB
Accept

States	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					

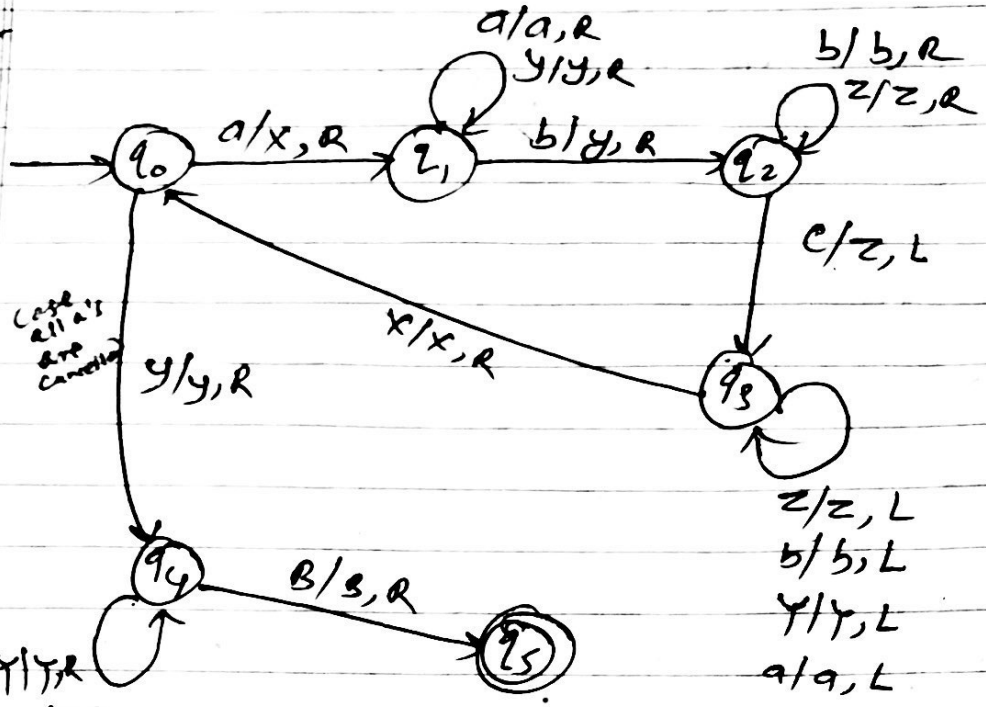
Now acceptance of 0011 by the TM can be described by

$q_0 0011 \vdash X q_1 011 B$
$\vdash X 0 q_1 11 B$
$\vdash X 0 Y q_1 1 B$
$\vdash X q_2 0 Y 1 B$
$\vdash q_2 X 0 Y 1 B$
$\vdash X q_0 0 Y 1 B$
$\vdash X X q_1 Y 1 B$
$\vdash X X Y q_1 1 B$
$\vdash X X Y Y q_2 B$
$\vdash X X Y q_2 Y B$
$\vdash X X q_2 Y Y B$
$\vdash X X q_0 Y Y B$
$\vdash X X X Y q_3 Y B$
$\vdash X X Y Y B q_4 B$

Scanned with CamScanner

Q Design a TM accepting $\{a^n b^n c^n / n \geq 1\}$

Soln:



$\delta(q_4, b) = (q_{reject}, b, R)$
$\delta(q_4, c) = (q_{reject}, c, R)$
$\delta(q_1, z) = (\quad,\quad,\quad)$
$\delta(q_1, c) = (\quad,\quad,\quad)$
$\delta(q_2, B) = (\quad,\quad,\quad)$

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

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$\Sigma = \{a, b, c\}$ → i/p alphabets.

$\Gamma = \{a, b, c, x, y, z, B\}$ → Tape symbols

δ is,

$Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

$q_0 = q_0$

B = blank symbol

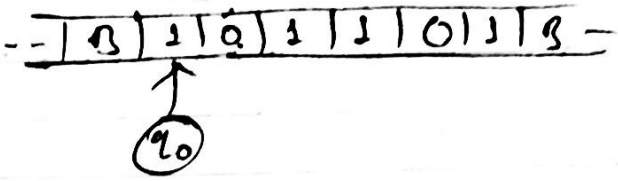
$F = q_5$

Scanned with CamScanner

Q. Design a TM that outputs 1's complement of a binary string.

⇒

1's comp. of 1 0 1 1 0 1 = 0 1 0 0 1 0



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

$\tau = \{0, 1, B\}$

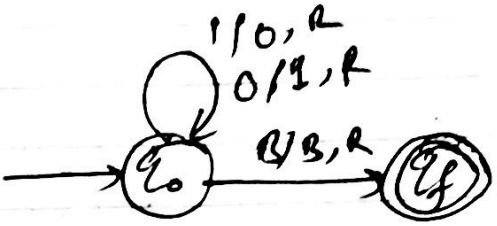
$Q = \{q_0, q_f\}$

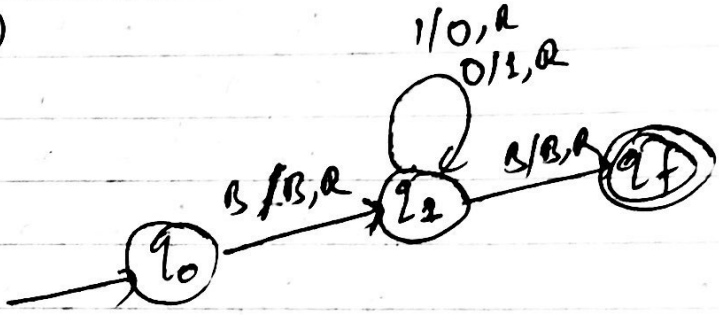
δ consists of

$\delta(q_0, 0) = (q_0, 1, R)$

$\delta(q_0, 1) = (q_0, 0, R)$

$\delta(q_0, B) = (q_f, B, R)$





OR

$\delta(q_0, B) \rightarrow (q_1, B, R)$

$\delta(q_1, 1) \rightarrow (q_1, 0, R)$

$\delta(q_1, 0) \rightarrow (q_1, 1, R)$

$\delta(q_1, B) \rightarrow (q_f, B, R)$

Scanned with CamScanner

Q. Design a TM that checks well formed string of parenthesis.

Soln ⇒ B (( ) ( )) ( ) B

Idea:-

Search for closing brackets if that exists mark it (eg by X) & move left then mark opening bracket.

$\delta(q_0, C) \longrightarrow (q_0, C, R)$

$\delta(q_0,)) \longrightarrow (q_1, X, L)$

$\delta(q_1, C) \longrightarrow (q_0, X, R)$

$\delta(q_0, X) \longrightarrow (q_0, X, R)$

$\delta(q_1, X) \longrightarrow (q_1, X, L)$ ← keep moving left

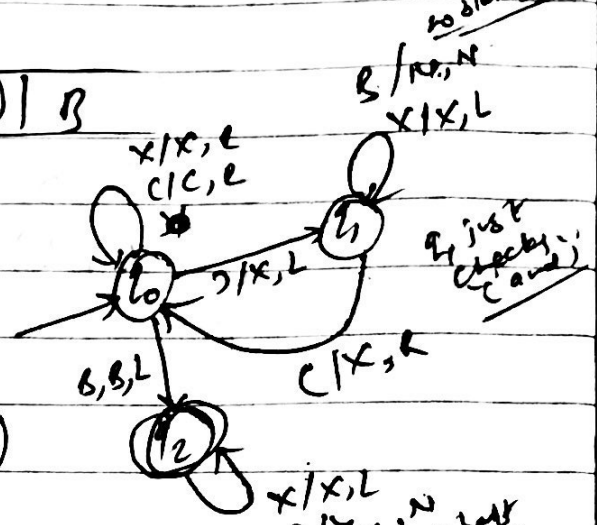
$\delta(q_0, B) \longrightarrow (q_2, B, L)$

$\delta(q_2, X) \longrightarrow (q_2, X, L)$

$\delta(q_2, B) \longrightarrow (q_2, Yes, N)$ → Halt & accept

$\delta(q_1, C) \longrightarrow (q_2, No, N)$ → Halt & reject

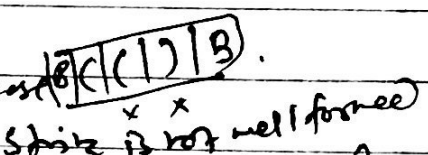
$\delta(q_1, B) \longrightarrow (q_1, No, N)$



Case → $\delta(q_1, B) = (q_1, N, H)$

B |) | (|)) q

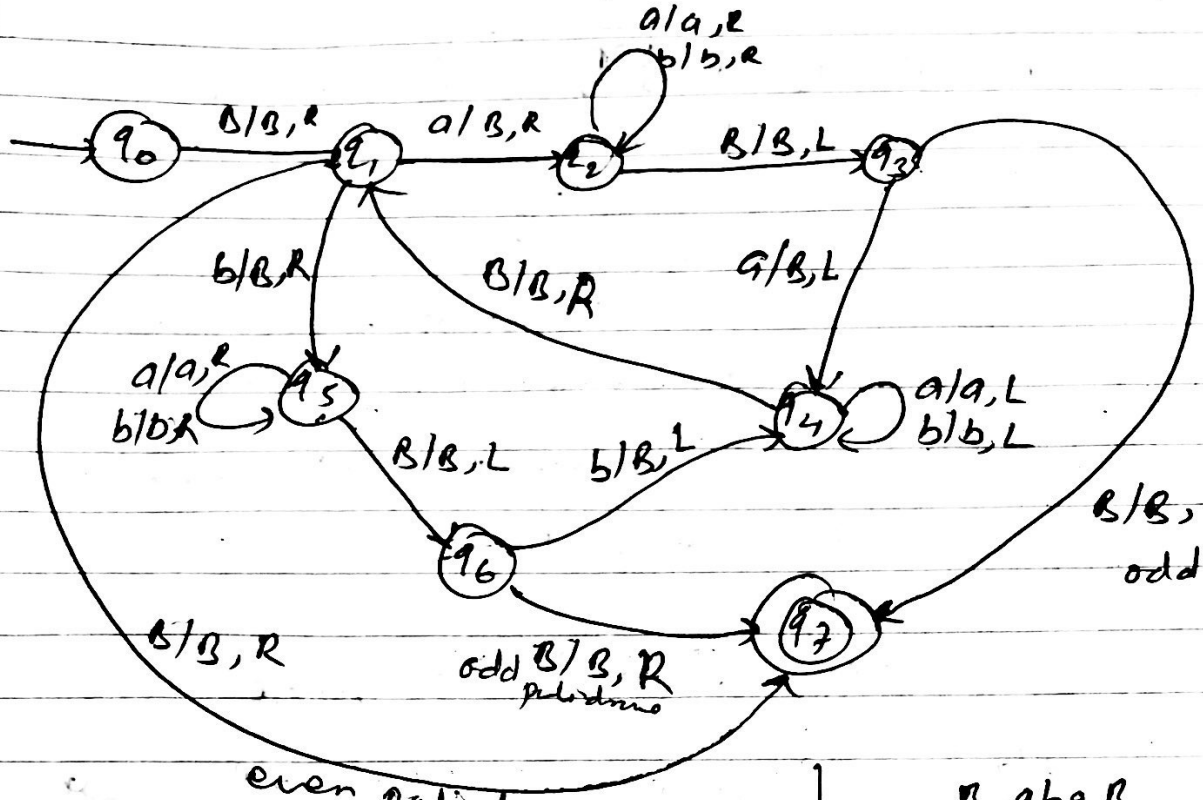
↑ q_0



Scanned with CamScanner

Q Design a TM that accepts even and odd palindromes.

Soln



for, w = BabbaB

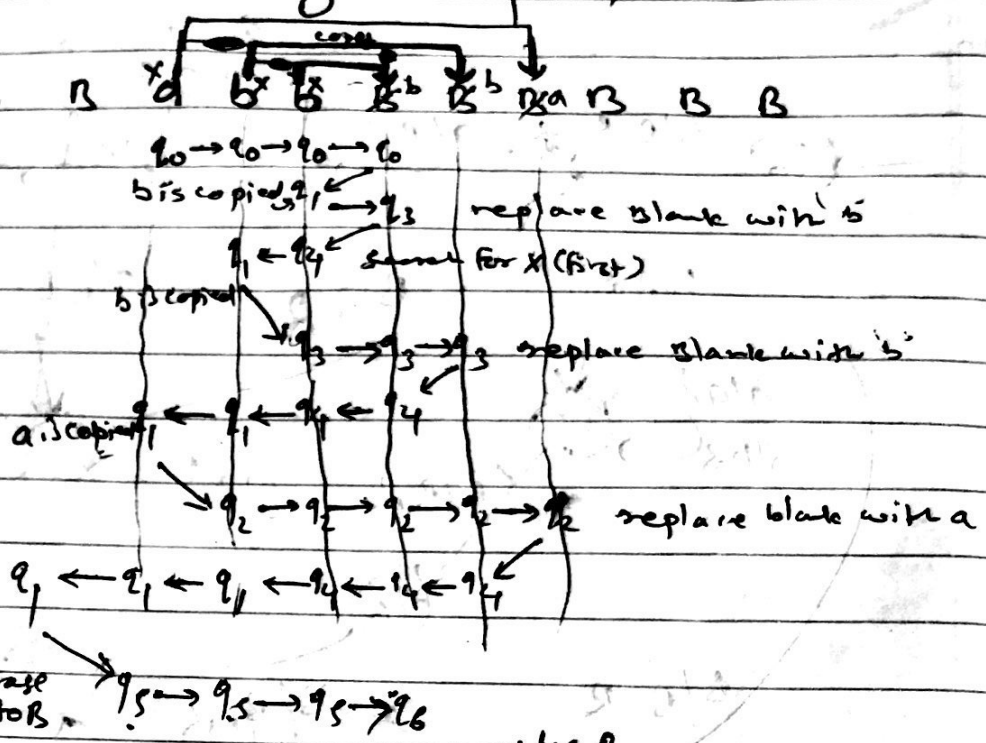
q_0 BabbaB
B q_1 abbaB
B B q_2 bbaB
B Bb q_2 baB
B Bbb q_2 aB
B Bbba q_2 B
B Bbb q_3 aB
B Bbb q_4 bBB
B Bb q_4 bbBB
B B q_4 bbBB
B q_4 B bbBB
B B q_1 bbBB
B BB q_5 bBB
B BBb q_5 BB
B BBB q_6 bBB
B BB q_4 BBB
B B B q_1 BBB
B BB q_7 BBB
Hence

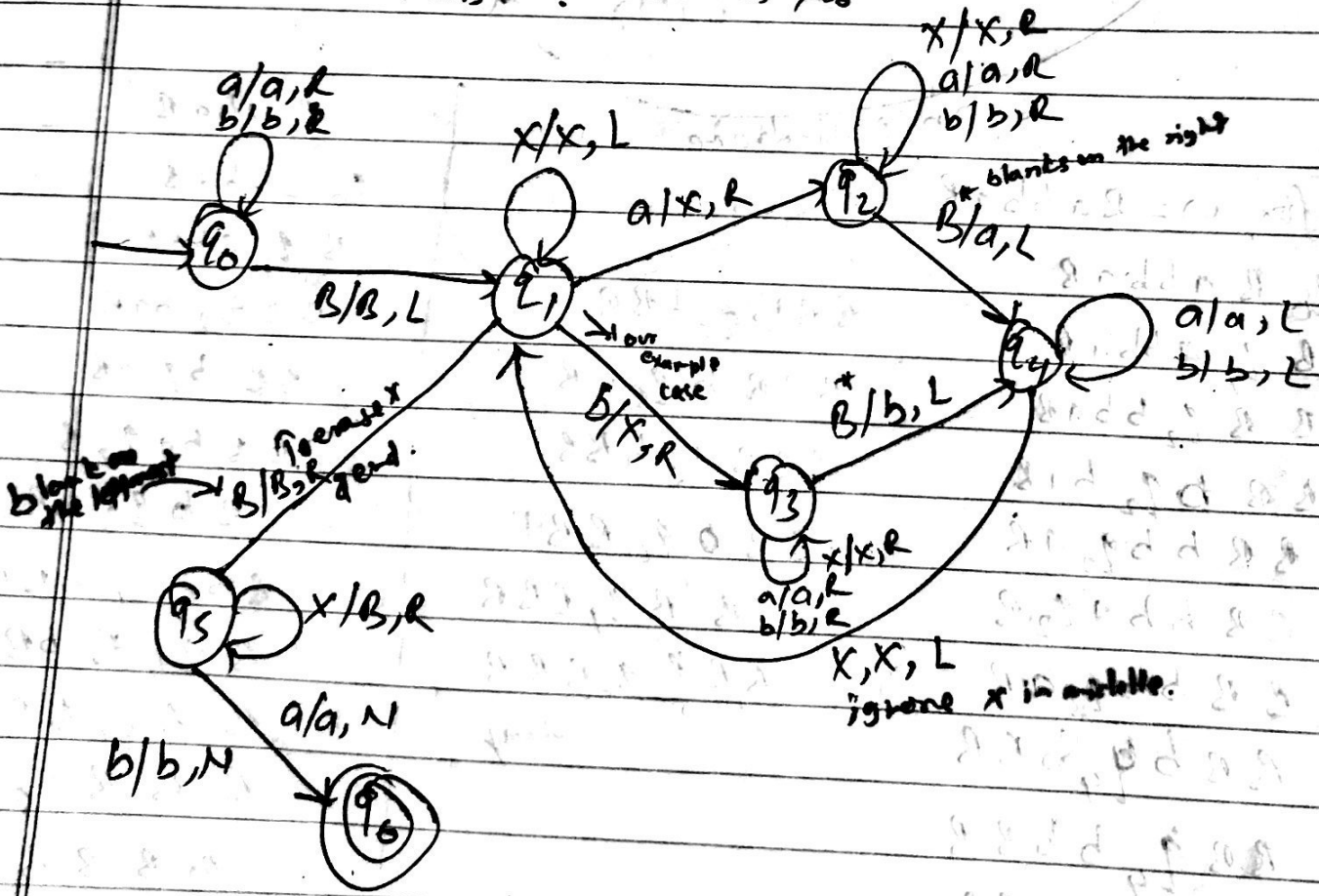
B aba B
q_0 B aba B
B q_1 aba B
B B q_2 baB
B Bb q_2 aB
B Bb q_3 aB
B B q_4 bBB
B q_4 B bBB
B B q_1 bBB
B B B q_5 BB
B B B B q_6 B
B B B B B q_7 B

Scanned with CamScanner

Q. TM to reverse string over {a, b}

eg:- B B B X a b b b a B B B





Scanned with CamScanner

Subroutines:-

⇒ Subroutines are collection of interacting components.

⇒ A TM subroutine is a set of states that perform some useful process.

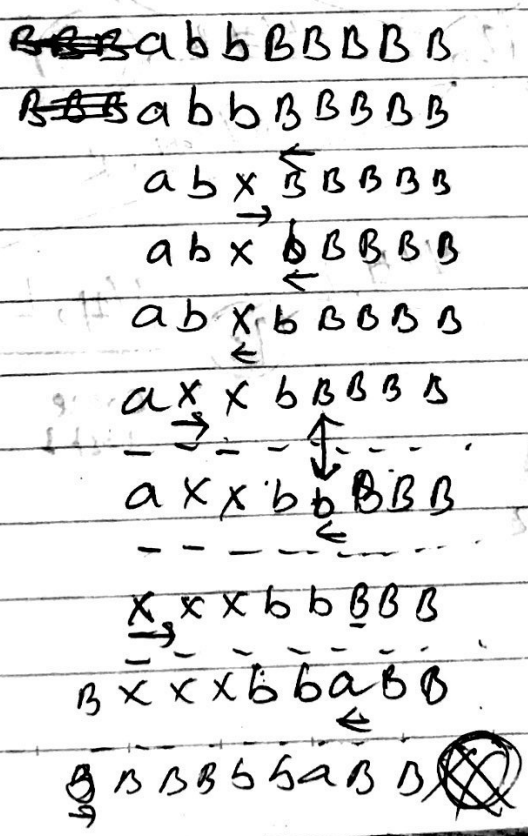
⇒ This set of states includes a start state and other states that serves as the "return" to pass control to ~~[illegible]~~ other set of states called subroutine

⇒ The "call" of a subroutine occurs whenever there is a transition to its initial state.

Example:- above.

Subroutine that copies a & b in blank spaces

Visualize:



Scanned with CamScanner

Date: Page:

Q TM to accept the strings of form ww^R

$w = abb$, $w^R = bba$

eg $w = ab$ $w^R = ba$

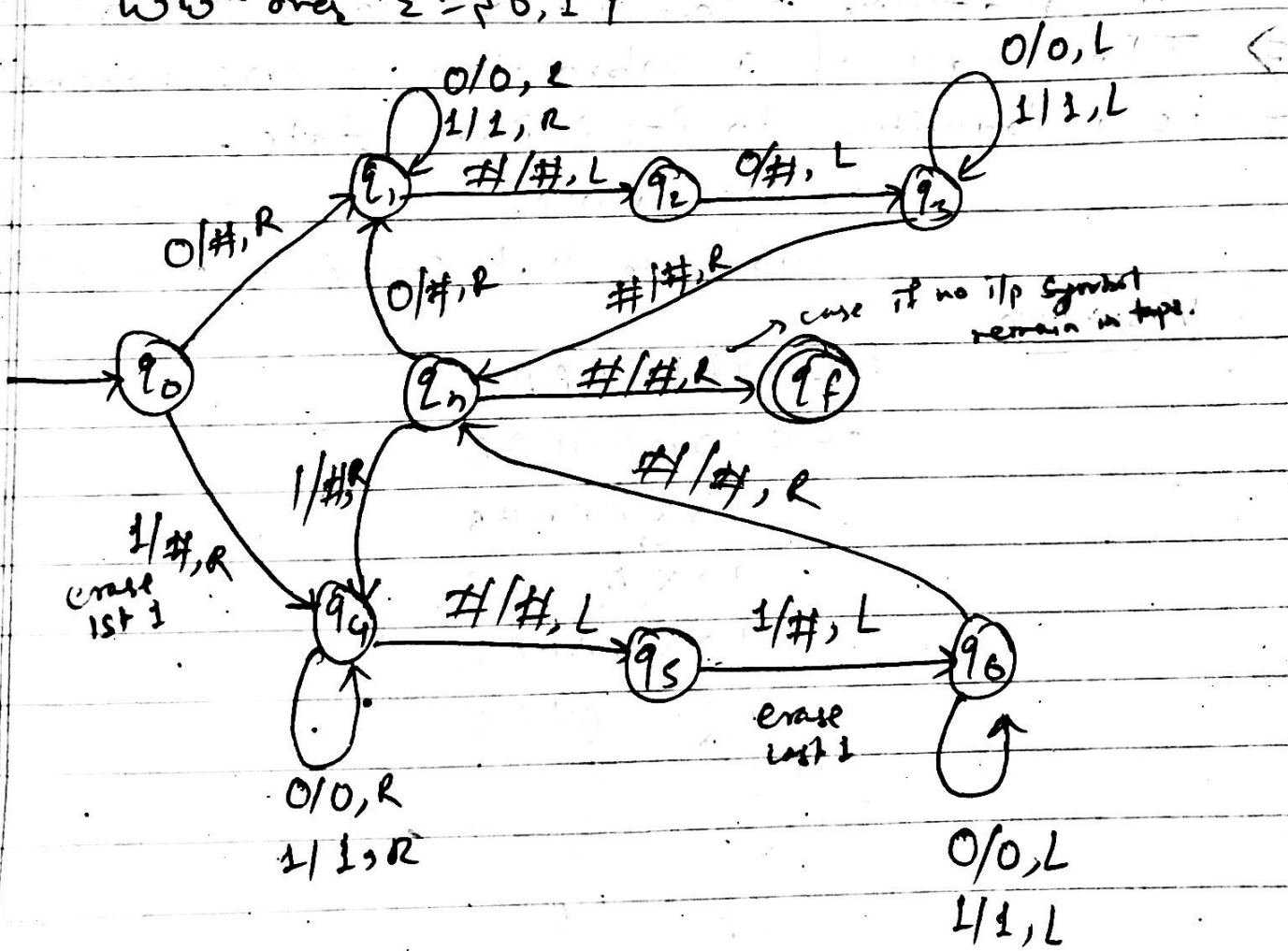
$\therefore ww^R = abba$

#	a	b	b	a	#	- -
#	#	b	b	a	#	- -
#	#	b	b	#	#	- -
#	#	b	b	#	#	- -
#	#	#	b	#	#	- -
#	#	#	#	#	#	

$w = 0110$ $w w^R = 0110$

#	0	1	1	0	#	- -
#	#	1	1	0	#	
#	#	1	1	#	#	
#	#	#	1	#	#	
#	#	#	#	#	#	

ww^R over $\Sigma = \{0, 1\}$?



Scanned with CamScanner

Date: Page:

The language of Turing Machine :-

⇒ If $T = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ is a Turing machine and $w \in \Sigma^*$, then language accepted by T, ~~L(T)~~

$$L(T) = \{w \mid w \in \Sigma^* \text{ and } q_0 w \vdash^* \alpha p \beta$$

for some $p \in F$ and any tape string α and β.

⇒ The set of Languages that can be accepted by using TM are called recursively enumerable languages.

⇒ The turing machine is designed to perform at least the following three roles;

1) As a language recognizer.
2) As a computer of function.
3) As an enumerator of string of a language.

Turing machine for computing a function.

⇒ A TM can be used to compute function. i.e. TM can also be used to decide whether a function is computable or not.

⇒ A function is said to be turing computable if there exicts a TM $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ such that it satisfies all the instants of the given function.

Scanned with CamScanner

⇒ Let us consider a function,

$f(w) = u$

Then, this function is said to be turing computable if there exists a TM such that

$$(q, \underline{\#}w\underline{\#}) \xrightarrow[M]{*} (h, \underline{\#}u\underline{\#})$$

⇒ Here, the underline in the blank symbol represents that the blank symbol is under the read/write head & h is the halt state.

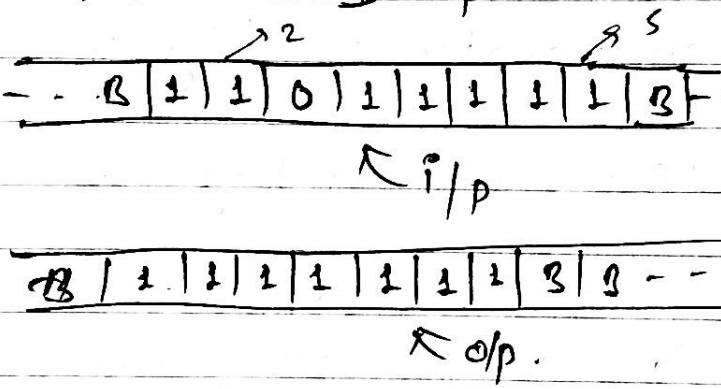
Turing computable

Design a TM which computes the function

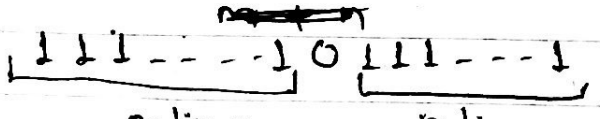
$f(x,y) = x+y$

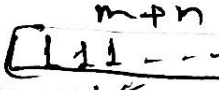
$f(2,5) = 2+5 = 7$

⇒



Let us construct a TM which performs the addition of two integers.





Scanned with CamScanner

m n

Data: Page:

1 1 ---- 1 0 1 1 --- 1 1 # # B

1 1 ---- 1 0 1 1 --- 1 1 # #

replace '0' by '1'

1 1 ---- 1 1 1 1 --- 1 1 # #

m + n + 1

go right until right most blank symbol reached

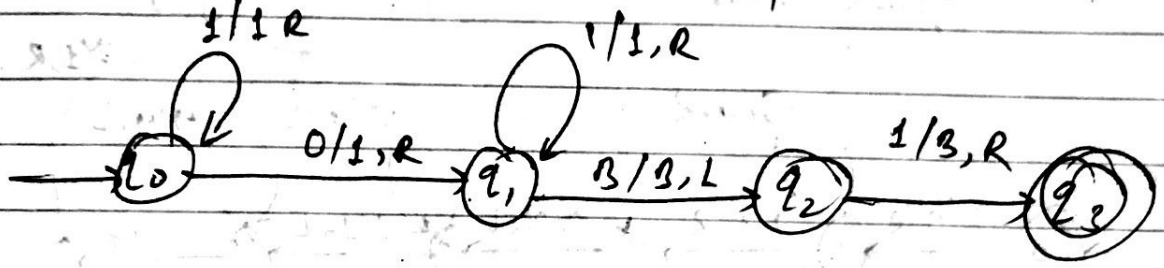
1 1 ---- 1 1 1 --- 1 1 # #

move one position left

1 1 --- 1 1 1 --- 1 1 # #

erase last 1 & move right.

1 1 --- 1 1 1 --- 1 # # #



eg by ID

q_0 1 1 0 1 1 1 1 1 B

⊢ 1 q_0 1 0 1 1 1 1 1 B

⊢ 1 1 q_0 0 1 1 1 1 1 B

⊢ 1 1 1 q_1 1 1 1 1 1 B

⊢ 1 1 1 1 q_1 1 1 1 1 B

⊢ 1 1 1 1 1 q_1 1 1 1 B

⊢ 1 1 1 1 1 1 q_1 1 1 B

⊢ 1 1 1 1 1 1 1 q_1 1 B

⊢ 1 1 1 1 1 1 1 1 q_1 B

⊢ 1 1 1 1 1 1 1 q_2 1 B

⊢ 1 1 1 1 1 1 1 B q_3 B

∴ accepted by Turing machine

Date:__________ Page:________

Example:- Design a TM which computes the function $f(x) = x+1$ for each x belonging to set of natural numbers.

Soln:- Given the function, $f(x) = x+1$, Here, we represent the input x on the tape by a no. of 1's on the tape.

i.e. for $x=1$, input will have B1B,
for $x=2$, " " " B11B
and so on,

Similarly, output can be seen by the no. of 1's on the tape when machine halts.

Let us configure a TM as;

$TM = (Q, \Sigma, \Gamma, \delta, q_0, B, \{q_f\})$;

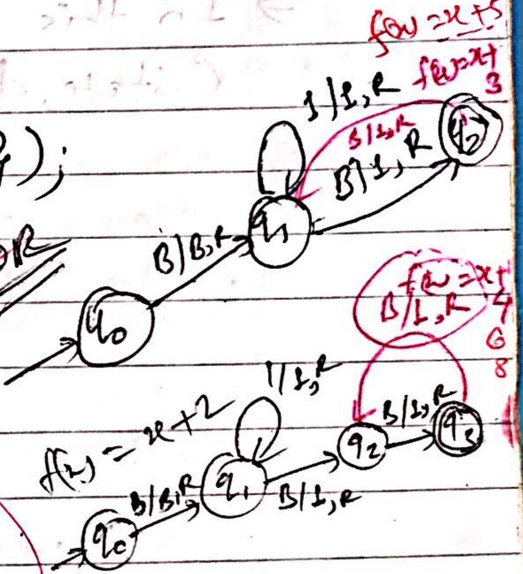
Where,

$Q = \{q_0, q_f\}$

$\Gamma = \{1, B\}$

Then δ can be defined as,

	B	1
q_0	$(q_0, 1, S)$	$(q_f, 1, R)$
q_f	–	–



let the input be $x=4$, So input tape at initial step consists of B1111B.

Then

$(q_0, B1111B) \vdash (q_0, 11111B)$

$\vdash (q_f, 11111B)$

Which means o/p is 5 #accepted

Scanned with CamScanner

Turing machine with storage in the state:

⇒ Generally in Turing machine, any state represents the position in the computation. But the state can also be used to hold a finite amount of data.

⇒ We can use finite control ~~not only~~ to represent a position in the computation of Turing machine ~~but~~ to hold a finite amount of data.

⇒ In this case state is considered as a tuple (State, data).

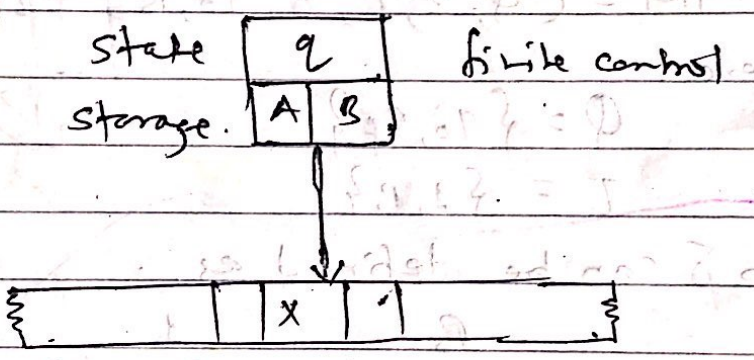


fig:- A TM viewed as having finite control and storage.

⇒ With this model of computation, δ is defined as,

$$\delta([q, A], X) = ([q_1, X], Y, R)$$

means that, q is the state and data portion associated to q is A. The symbol scanned on the tape is copied into the second component of

Scanned with CamScanner

If it remembers the 1st symbol on finite control. skips other until a blank. When blank encounters, it replaces remembered symbol with blank, the state and moves right entering state q_1 and replacing tape symbol by Y.

i.e. $w = 01111$ i.e. $w = 10000$

Example:-

This model of TM can be used to recognize the language like $01^* + 10^*$.

Thus the TM can be designed as;

$$M = (Q, \{0,1\}, \{0,1,B\}, \delta, [q_0, B], \{[q_2, B]\})$$

(Σ, Γ, initial, F)

δ can be

1) $\delta([q_0, B], a) = ([q_1, a], a, R)$ (for $a = 0$ or 1)

Initially, q_0 is the control state, and the data portion of the state is B. The symbol scanned is copied into the second component of the state; M moves right, entering control state q_1 as it does so.

2) $\delta([q_1, a], b) = ([q_1, a], b, R)$. where $\bar{a}$ is the complement of a i.e. $\bar{a} = 0$ if $a = 1$, $\bar{a} = 1$ if $a = 0$

In state q_1, M skips over each nonblank symbol and continues moving right. for $a = 0$ or 1.

3) $\delta([q_1, a], B) = ([q_2, B], a, R)$

If M reaches the first blank, it enters the accepting state. B is copied to the finite control.

Note:- M has no definition for $\delta([q_1, a], a)$; for $a = 0$ or 1, thus if M encounters a second occurance of same symbol it stopped

Scanned with CamScanner

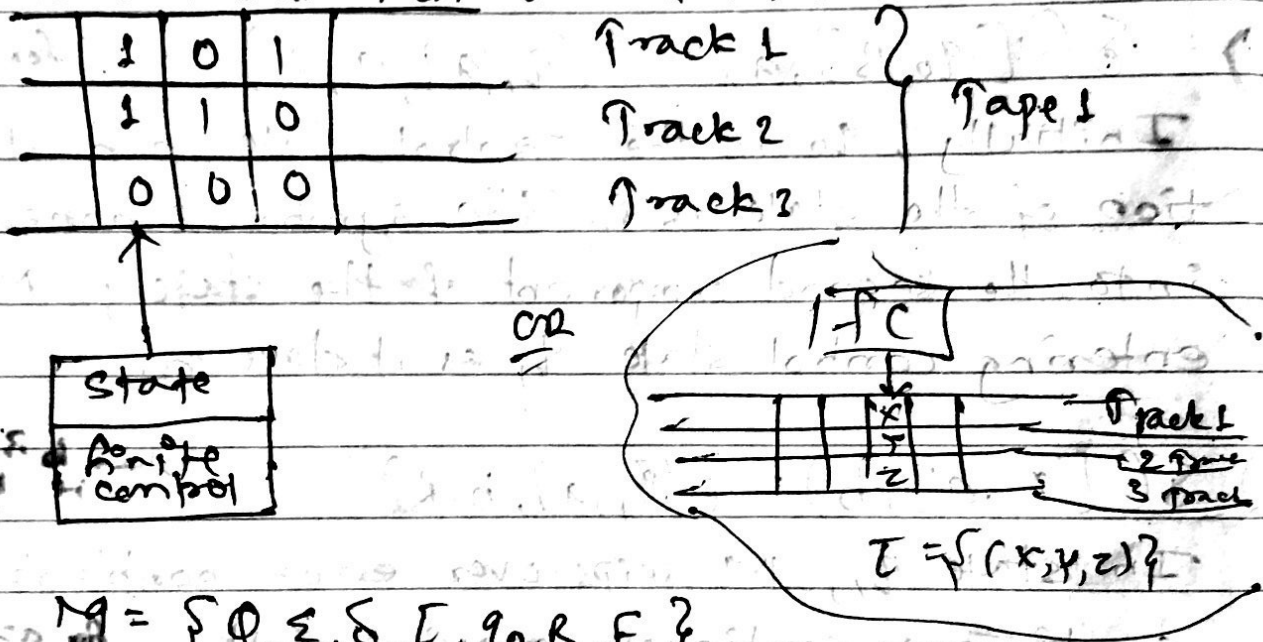
Initially in its control, it halts. without having entered the accepting state.

Turing Machine with multiple tracks :-

⇒ The tape of TM can be considered as having multiple tracks. Each track can hold one symbol.

⇒ One head reads/writes on all tracks simultaneously & then only it shifts to Left or right.

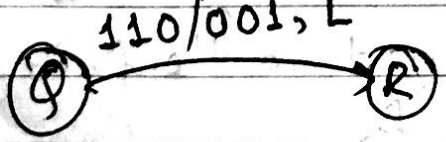
⇒ These machine do not extend what a TM can do.



$M = \{Q, \Sigma, \delta, \Gamma, q_0, B, F\}$

δ :

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



⇒ The tape head moves up and down scanning symbols in the tape at one position.

Scanned with CamScanner

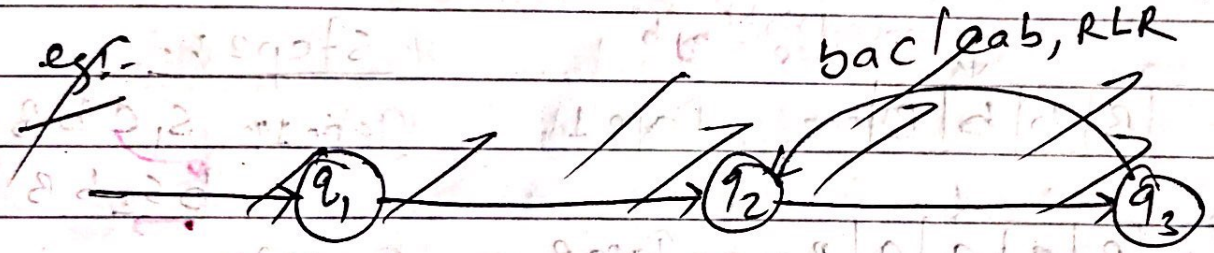
Date: Page:

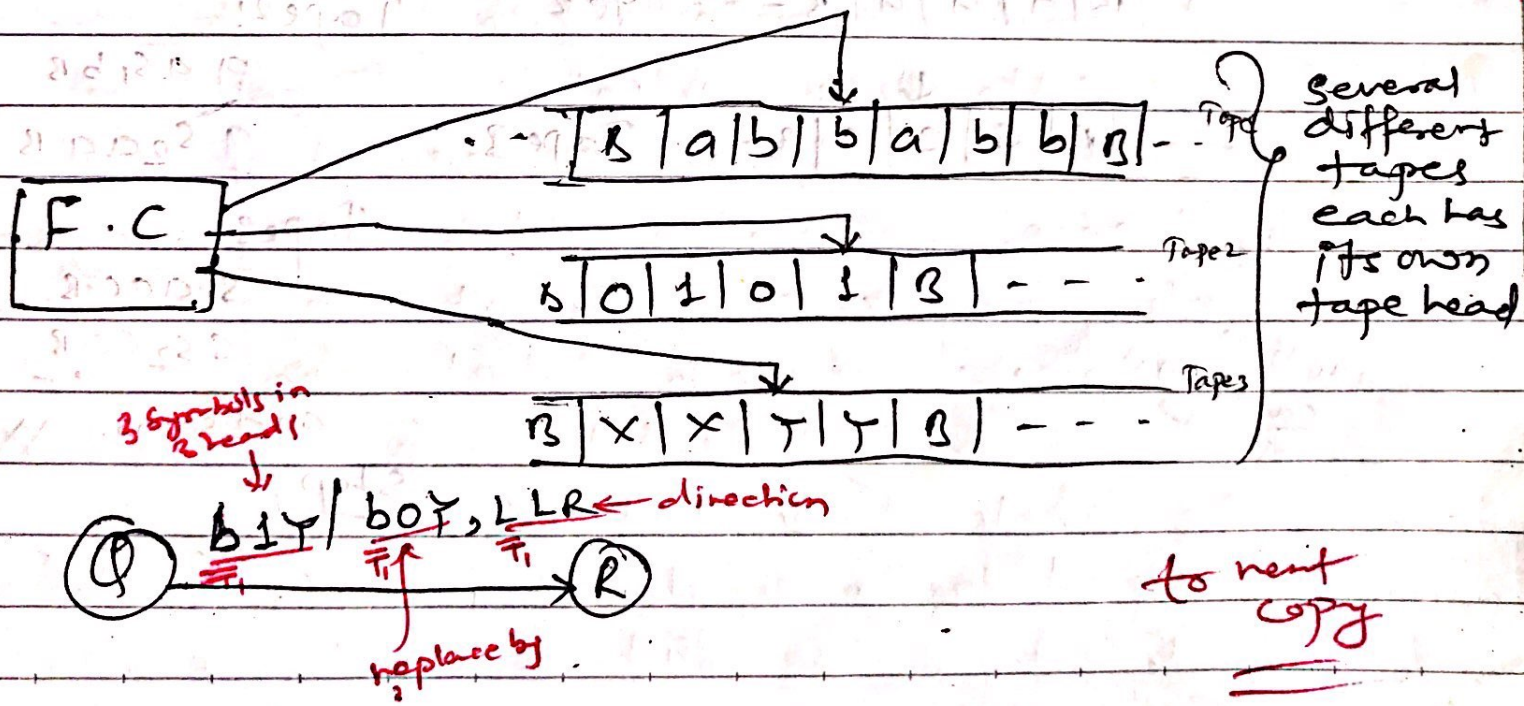
Multitape turing machines:-

- It is a regular TM with several tapes.
- Each tape has it's own head. (Each head read, write move simultaneously or individually)
- It is no more powerful than single Tape TM. It is not about speed, ~~efficiency~~ efficiency (fewer steps of computation) or ease of programming.
- Though multitape TM has fewer steps (several operation can be performed in single transition), it does not add addition computation power than single Tape TM. i.e. accepts the same language (Recursively enumerable & Largest) the single tape TM accepts.

$$\delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times (L, R, S)^k.$$

eg:-



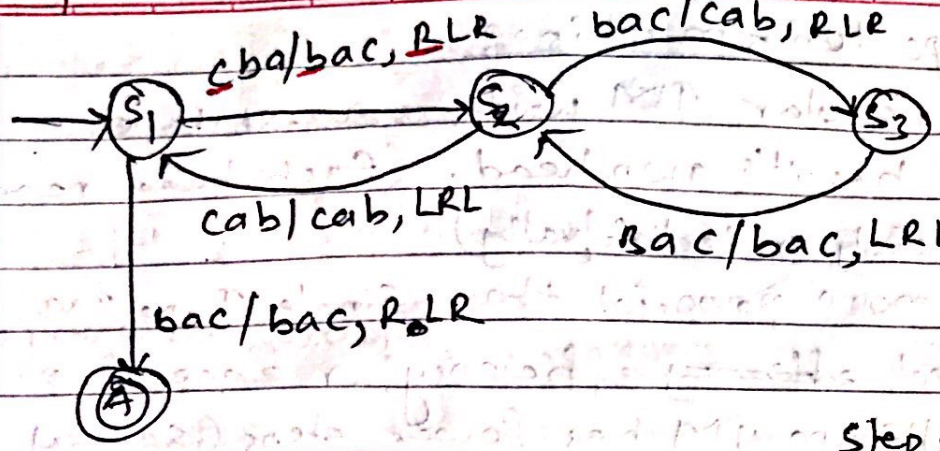


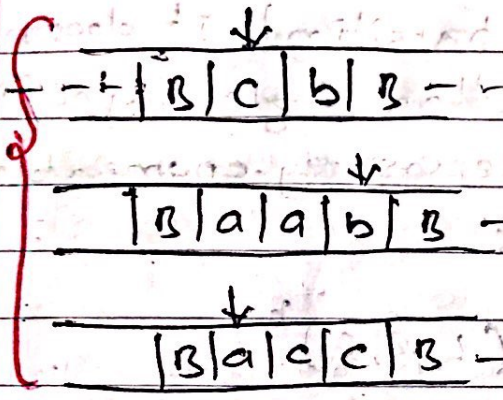
Scanned with CamScanner

if time

Date: Page:

eg



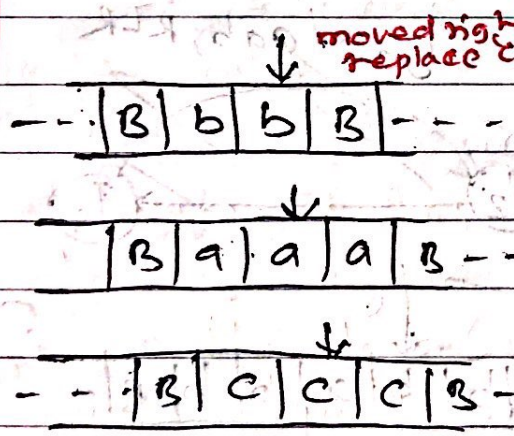


Step 1:-

Tape 1: s_1 c b B

Tape 2: a a s_1 b B

Tape 3: s_1 a c c B



Step 2:-

Tape 1: s_1 c b B

b s_2 b B

Tape 2:

a a s_1 b B

a s_2 a a B

Tape 3:

s_1 a c c B

c s_2 c c B

and so on.

Step 3:-

multitape can do word copy

Scanned with CamScanner

Date: _____ Page: _____

if time

Theorem:-

Every multitape TM has an equivalent single tape TM.

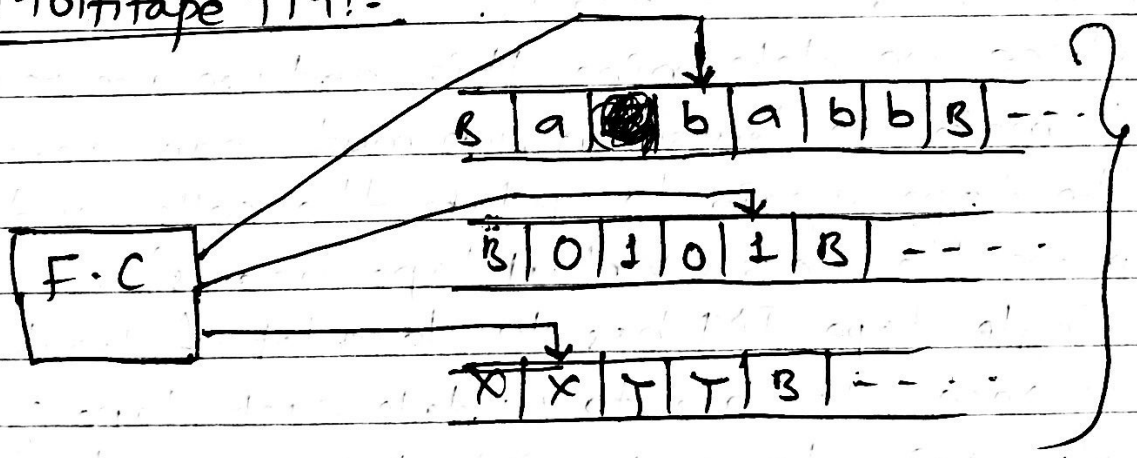
⇒ Here, "equivalent" means it decides / recognizes the same languages.

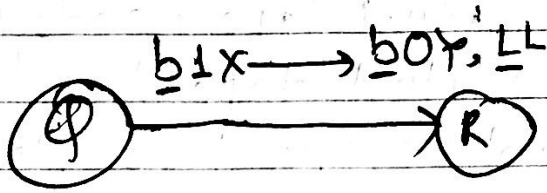
Proof ⇒

Given a multitape TM, we need to show how to build a single-tape TM.

- Need to store all tape's data in single tape.
- Each tape has a tape "head", which needs to be shown indicating how to store that information.
- Needs to transform a move in the multitape TM into one or more moves in the single tape TM.

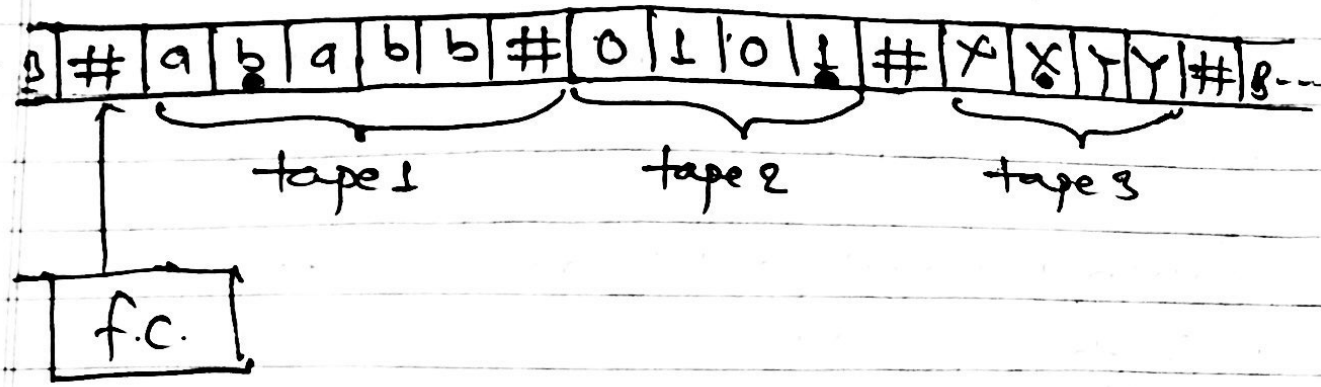
Multitape TM:-





Scanned with CamScanner

⇒ We need to represent all three tapes into one tape.



- Introduce blanks for separating ~~the~~ each tape's data.
- Add "dots" to show where Head "k" is.
- To simulate a transition from state Q, we must scan our tape to see which symbols are "under" the heads.
- Once we determine this, and we are ready to make the transition, we must scan across the tape again to update the cells and move the dots (virtual tape heads).
- Single tape TM has to do a lot of works at once, It has to update each tape symbols; going to each virtual tape heads and put marks on each virtual tape heads.
- It needs K different steps to implement in Single Tape TM which can be performed on a single step using Multitape TM. but we can still do the same task

Scanned with CamScanner

Date: Page:

for multitape & single tape TM.

Non-Deterministic Turing Machine :-

⇒ A non-deterministic Turing Machine (NTM)

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

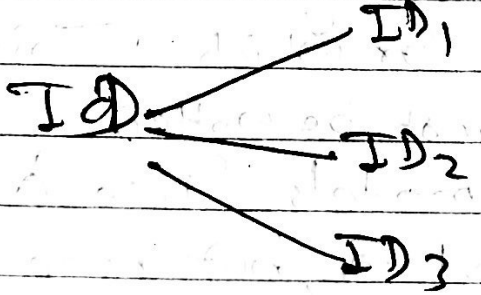
δ is the transition function which differs from deterministic turing machine.

⇒ Here, the transition function δ is such that for each state q and tape symbol X, $\delta(q, X)$ is a set of triplets,

$$\{(q_1, Y_1, D_1), (q_2, Y_2, D_2) \cdots (q_k, Y_k, D_k)\}.$$

where k is any finite integer.

⇒ The NTM can choose, at each step, any of the triples to be the next move. It cannot pick a state from one, a tape symbol from another, and the direction from another.



Scanned with CamScanner

Church-Turing Thesis:-

⇒ The church-Turing thesis states that in most common form that "every computation or algorithm can be carried out by a Turing machine."

⇒ This statement was proposed by two mathematician (Alonzo Church & Alan Turing) It is not a mathematically precise statement so unprovable However, it is now generally assumed to be true.

⇒ Some arguements for accepting the thesis are:

1. Anything that can be done on existing digital computer can also done by a Turing machine.
2. No one has yet been able to suggest a problem, solvable by what we consider an algorithm, for which a Turing machine program cannot be written.
3. Alternative models (like λ calculus by Turing) have been proposed for, but none of them is more powerful than the Turing machine model.

Turing thesis is still an assumption.
that has not been proved wrong.

Scanned with CamScanner

Date: Page:

Universal TM:- Simulating work of TM

⇒ A TM capable of simulating the behavior of any TM.

⇒ In a general TM, once δ is defined, the machine is restricted to carrying out one particular type of computation (hardwired). i.e. special purpose computer.

⇒ Digital computers, on the otherhand, are general purpose machines that can be programmed/reprogrammed to do different jobs at different times.

⇒ Consequently, Turing machine cannot be considered equivalent to general purpose digital computers.

⇒ This objection can be overcome by designing a reprogrammable Turing machine, called a Universal Turing Machine.

⇒ A Universal TM M_U is an automation that, given as input the description of any TM M and a string w, can simulate the computation of M on w.

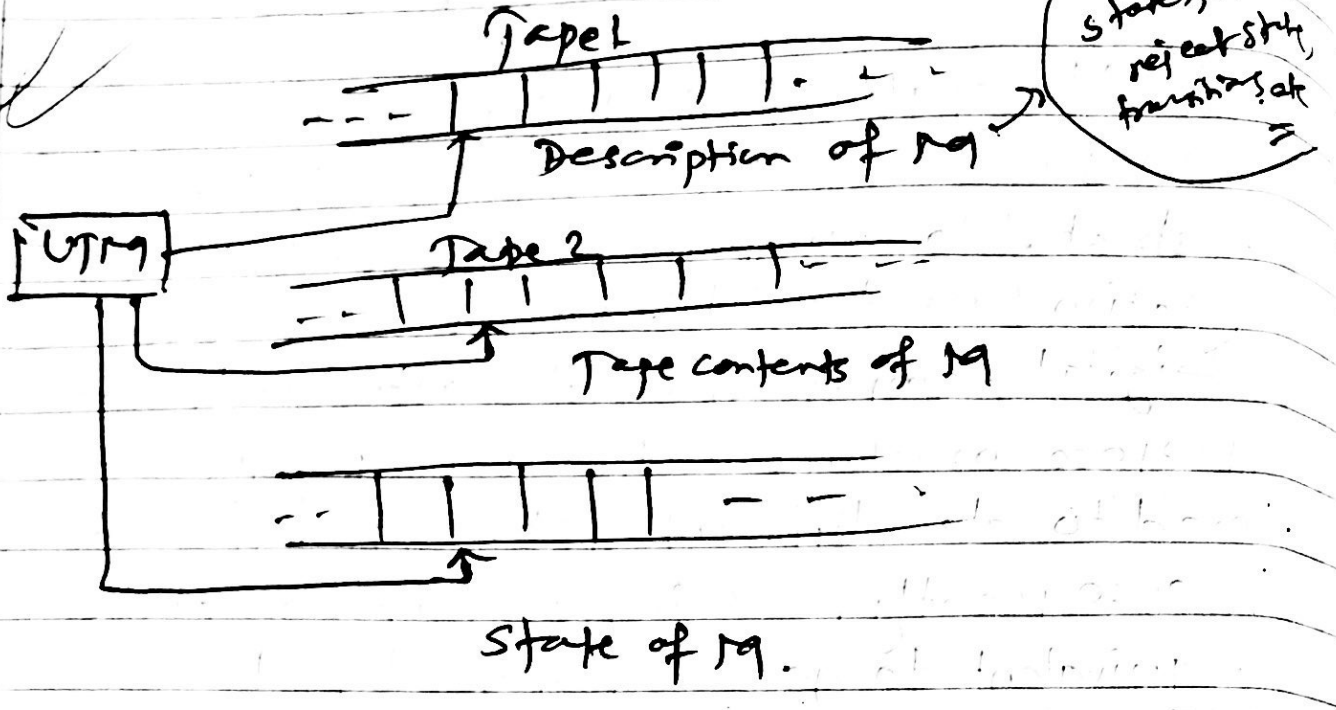
⇒ More precisely, UTM can simulate the behavior of an arbitrary TM over Σ.

TM are hardwired :- They execute one program at a time.

Soln:-

UTM :- reprogrammable + simulates any other TM.

Scanned with CamScanner



Description of M:-

Here, Symbols are encoded as ~~strings~~

eg:- Symbols a b c d ...
encoding 1 11 111 1111 ...

State encoding:-

States	q_1	q_2	q_3	q_4
encoding	1	11	111	1111

Head move encoding:-

move:	L	R
encoding:	1	11

Transition encoding.

Transition: $\delta(q_1, a) = (q_2, b, L)$

encoding: 10100111011

Comma (separator) encoded as 0.

Separators

Date: Page:

Halting problem:-

- In computability theory, the halting problem is a decision problem which can be stated as follows.

"Given a description of a program ~~and~~ and a finite input, decide whether a program finishes running or will run forever.

for eg:- in pseudocode,

```
while (true)
    continue
```

```
n ≥ 5
while (n > 1)
do something
n = n + 1
```

does not halts, rather it goes forever in an infinite loop.

on the other hand,

```
Printf "Hello world"
```

does halt.

Scanned with CamScanner

Date: Page:

Turing Machines and Computers :-

Assignment — Page 332 - 337

Simulating a TM by a computer :-

⇒ To simulate a TM by a computer, A serious question arises when we consider how our program is to simulate the TM tape.

⇒ Tape of TM grow infinitely long, but the computer's memory - main memory, disk and other storage devices are finite.

Can we simulate an infinite tape with a fixed amount of memory?

⇒ The answer is we cannot if we have no opportunity to replace the storage devices.

⇒ If we remove the hard disk ~~with~~ and replace it with empty, let us assume that as many disks as the computer needs is available.

⇒ Thus in above mentioned conditions only we can simulate TM as,

- We can arrange by placing disks in two stacks,
- One stack holds the data in cells on the TM tape that are located to the left of the tape head, and the other stack holds data to the right of the tape head.
- In case, while the processing goes on the leftmost or rightmost end then the disk is about to be full,

Scanned with CamScanner

the computer has to print message swap left or swap right so that human operator replaces with empty disk.

Simulating a Computer by a Turing machine:-

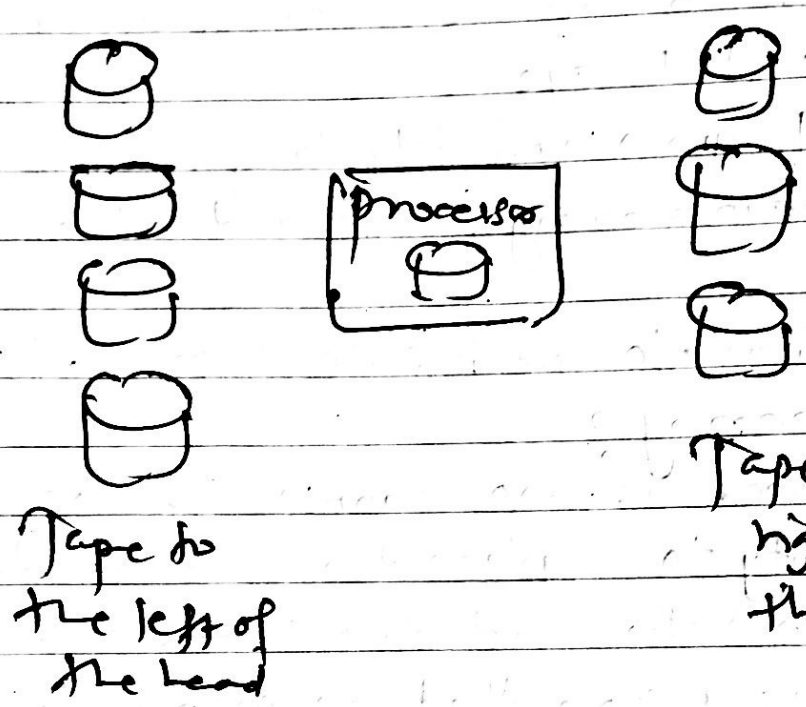


fig:- Simulating a TM with a common computer

Simulating a computer by a TM:-

=> TM uses several tapes, & it could be converted to a single tape TM.

- first tape might represent the entire memory of the computer.
- Next tape might represent the instruction counter (PC), which accounts the next instruction

Scanned with CamScanner

to be executed.

- Another tape can be used for memory address and con.
- To execute the instruction, the TM must find the contents of ~~the~~ one or more memory address
  - first the desired address is copied to one tape tape 3 and compared with the address of another tape tape 1, until a match is found.

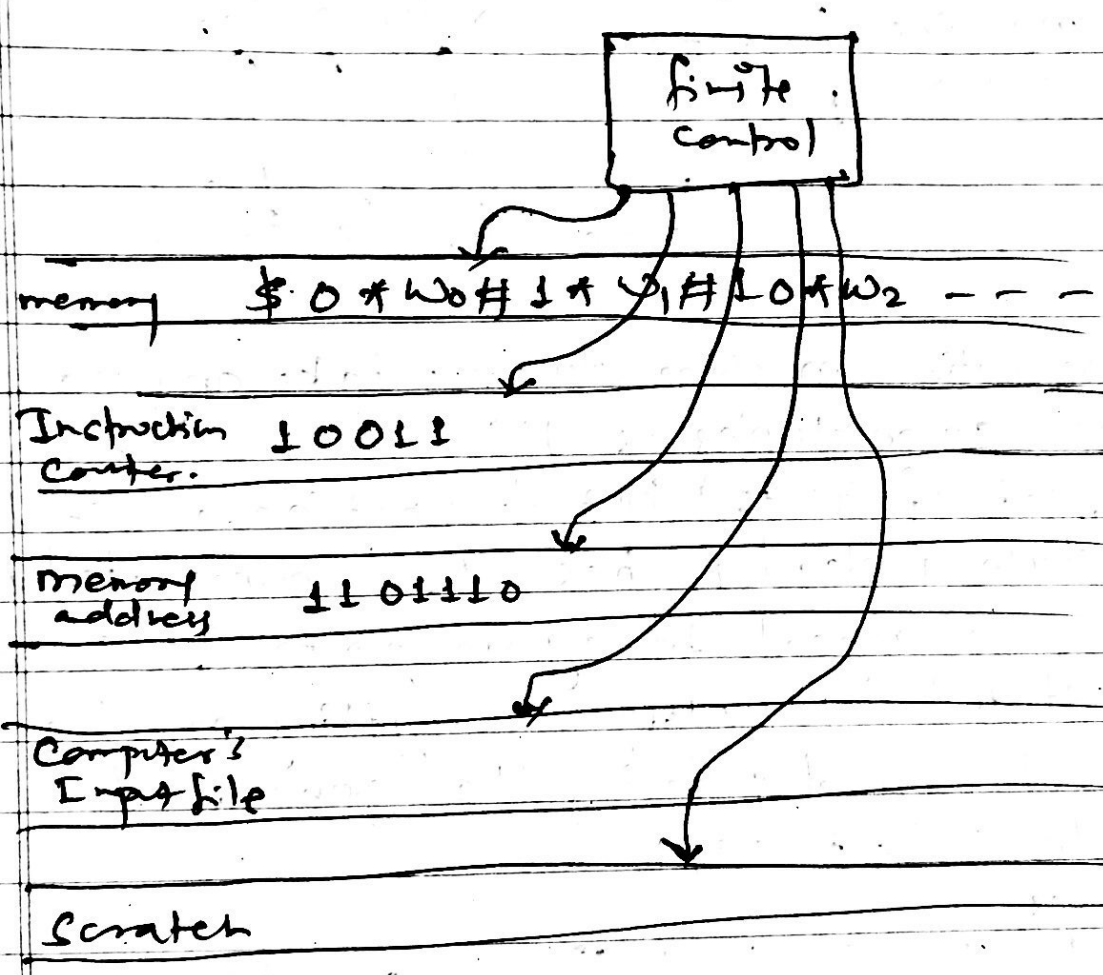


fig:- A TM that simulates a typical computer.

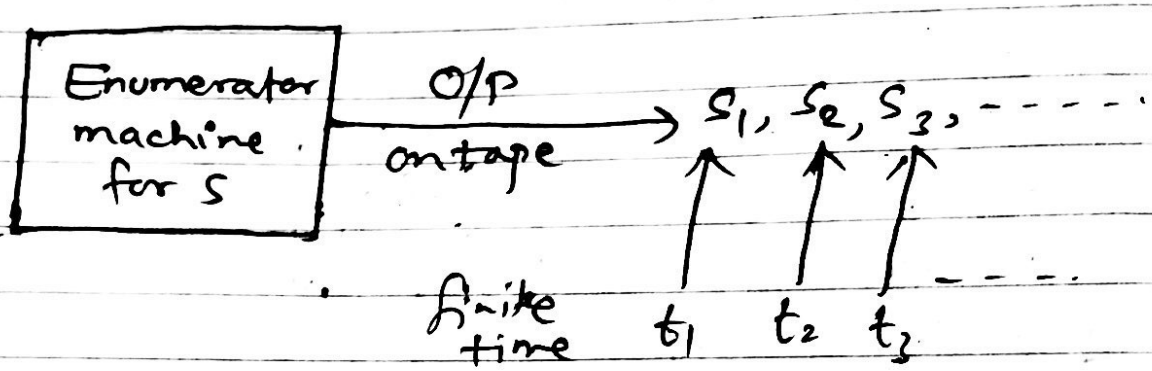
Scanned with CamScanner

Date: Page:

Enumerator:-

⇒ Let S be a set of strings. An enumerator for 'S' is a turing machine that generates (prints on tape) all the strings of 'S' one by one and each string is generated in finite time.

Strings :- $s_1, s_2, s_3 \dots \in S$

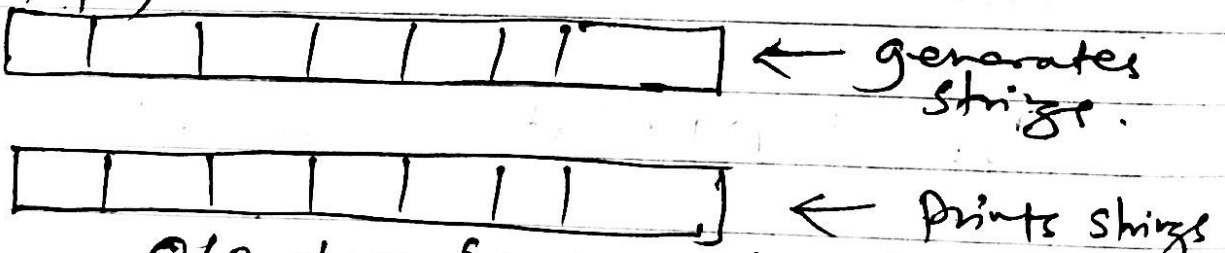


⇒ Enumeration machine does not take any i/p.

⇒ It has same tuples as TM such as (Σ, Γ, finite control, set of states) but does not have final/reject state instead it has q_e (enumerated state)

⇒ Once q_e reached. It again go to q_s state (start state) by erasing everything on the working tape and printing the generated strings into the output tape.

⇒ Two tapes,



Scanned with CamScanner

Multitape TM cont...

In the multitape TM, initially,

1. The Input (finite sequence of input symbols) w might be placed on the first tape.
2. All the other cells of the tapes hold blanks.
3. TM is on initial state q_0.
4. The head of the first tape is at the left end of the input.
5. All the other tape heads are at some arbitrary cell.

⇒ A move of multitape TM depends on the state and the symbol scanned by each of the tape head. In one move, the multitape TM does the following.

1. The control enters in a new state, which possibly may be same previous state
2. On each step, a new symbol is written on the cell scanned, these symbols may be same as the symbols previously there.
3. Each of the tape head makes a move either left or right or remains stationary. Different head may move different direction independently. ie. If head of 1st tape moves leftward, at the same time other head can move another direction or remains stationary.

Scanned with CamScanner

Equivalence of Multi-tape TM and Multi-track TM.

Theorem:- book page 316

Every Language accepted by a multitape TM is recursively enumerable.

proof:-

The proof is suggested by figure below.

Suppose Language L is accepted by a k-tape TM (multitape) M.

- We simulate M with a one-tape TM N (single tape) whose tape we think of as having 2k tracks (multitrack).
- Half of these tracks hold the tapes of M, a
- Other half of the tracks each hold a single marker that indicates where the head for the corresponding tape of M is currently located.
- figure assumes k = 2 (for multitape)

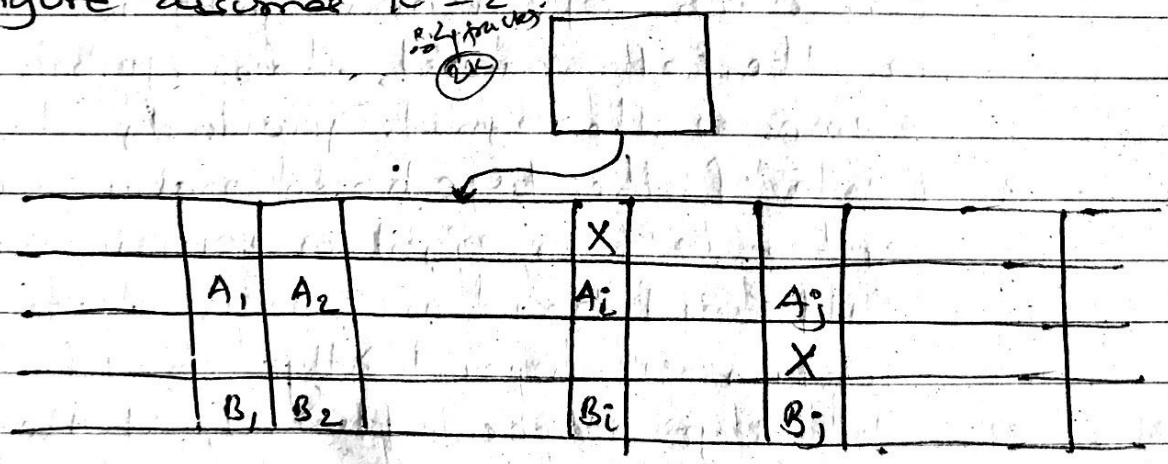


fig:- Simulation of 2 (k) -tape TM by a single Tape (4 tracks) Turing Machine.
(2k)

Scanned with CamScanner

- The second and fourth tracks holds the contents of the first and second tapes of M; → multitape TM
- Track 1 holds the position of the head of tape 1,
- Track 3 " " " " " second tape head.
- To simulate a move of M, N's head must visit the k head markers.
- So that N not get lost, it must remember how many markers are there in 'M' is stored as a component of N's finite control.
- After visiting each head marker and storing the scanned symbol in finite control, N knows what tape symbols are being scanned by each of M's heads.
- 'N' also knows the state of M, which it stores in N's own finite control. Thus, N knows what move M will make.
- 'N' now revisits each of the ~~M's head~~ head Marker on it's tape, changes the symbol in the track representing the corresponding tapes of M, and moves the head markers left or right, if necessary.
- finally, N changes the state of M as recorded in it's own finite control. At this point, N has simulated one move of M.
- Thus, whenever the simulated M accepts, N also accepts, & N does not accept otherwise.

Scanned with CamScanner

Non - Deterministic TM.

Restricted Turing Machines:-

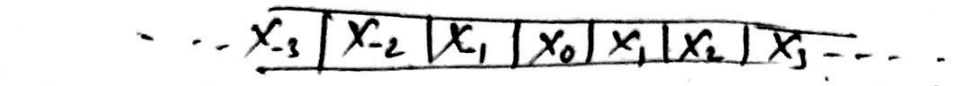
- Restrictions on the TM also give exactly the same language-recognizing power.
- We replace the TM tape that is infinite in both directions by a tape that is infinite only to the right.
-

Turing Machine with Semi-Infinite Tapes:-

- In fact, we can assume the tape is semi-infinite, i.e. there are no cells to the left of the initial head position.
- The trick behind the construction is to use two tracks on the semi-infinite tape.
  - The upper track represents the cells of the original TM that are at or to the right of the initial head position.
  - the lower track represents the positions left of the ~~head~~ initial position, but in reverse order.
  - The exact arrangement is suggested in figure,

Scanned with CamScanner

in original TM



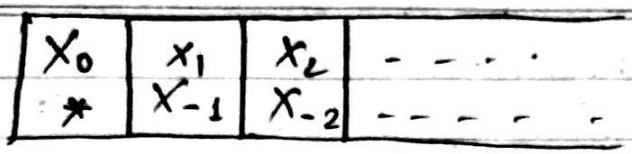


fig:- A semi-infinite tape can simulate a two-way infinite tape.

- The upper track represents cells $X_0, X_1, X_2, \ldots$, where X_0 is the initial position of the head; X_1, X_2, and so on, are the cells to it's right.
- Cells X_{-1}, X_{-2}, and so on, represent cells to the ~~left~~ left of initial position in Original (General) TM.
- Notice the * on the leftmost cells of bottom track. This symbol serves as an endmarker & prevents the head of semi-infinite TM from accidentally falling off the left end of the tape.

Scanned with CamScanner

Multi-stack Machines:-

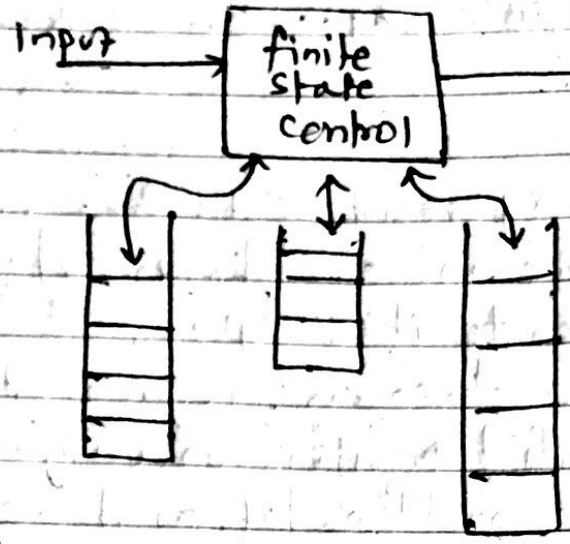


fig:- A machine with 3-stacks.

- A k-stack machine is a deterministic PDA with k-stacks.
- It obtains it's input, like the PDA does, from an input source, ~~rather than having the input placed on a tape or stack, as the TM does.~~
- The multistack machine has a finite control, which is one of a finite set of states.
- It has a finite stack alphabet, which it uses for all it's stacks.
- A move of the multistack machine is based on:
  1. The state of the finite control.
  2. The input symbol ~~chosen~~ read, which is chosen from the finite input alphabet.
     (× ε-symbol input is not used ∵ to make deterministic.)
  3. The top stack symbol on each of it's stacks.

Scanned with CamScanner

- In one move, the multistack machine can;
  a) Change to a new state.
  b) Replace the top symbol of each stack with a string of zero or more stack symbols. There can be • (and usually is), a different replacement string for each stack.

Thus a typical transition rule for a k-stack machine looks like:

$$\delta(q, a, X_1, X_2, \ldots, X_k) = (p, \gamma_1, \gamma_2, \gamma_3, \ldots \gamma_k)$$

- The interpretation of this rule is that in state q, with X_i on top of the ith stack, for $i = 1, 2, \ldots k$ the machine may consume 'a' (~~either~~ an input symbol) from its input, go to state p, & replace X_i on top of the ith stack by string γ_i, for each $i = 1, 2, 3, \ldots k$.
- The multistack machine accepts by entering a final state.

Assignment:- (page 327 book)

If a language L is accepted by a TM, then L is accepted by a 2-stack Machine.

Scanned with CamScanner

Counter Machines:-

- A counter machine may be thought of as, in one of two ways:

(1) - The counter machine has the same structure as the multistack machine, but in place of each stack is a counter.
- In one move, the counter machine can:
  a) change state.
  b) Add or subtract '1' from any of its counters, independently. However a counter is not allowed to become negative, so it cannot subtract '1' from a counter that is currently '0'.

(2) A counter machine may also be regarded as a restricted multistack machine.
The restrictions are as follows.
  a) There are only two stack symbols, which we shall refer to as Z_0 (the bottom of the stack marker) and X.
  b) Z_0 is initially on each stack.
  c) We may replace Z_0 only by a string of the form $X^i Z_0$, for some $i \geq 0$.
  d) We may replace X only by X^i for some $i \geq 0$. That is Z_0 appears only on the bottom of each stack, and all other stack symbols, if any, are X.

Scanned with CamScanner