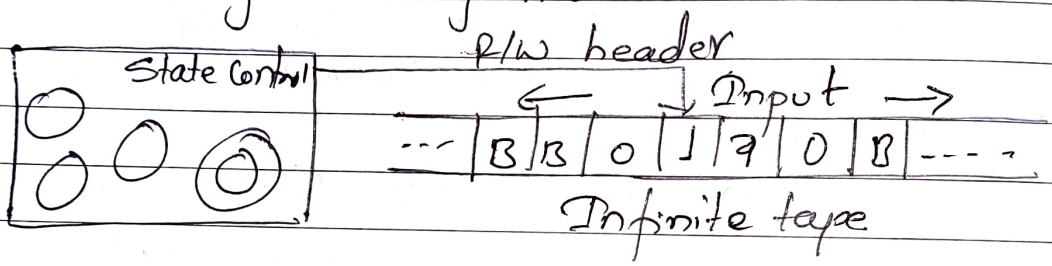
Unit-6 - Turing Machine.

- Ankit Pangeni

DATE

* Introduction to Turing Machine.

A turing machine is a finite automaton with a two-way access to an infinite tape Turing machine can be used to accept all context-free languages, but also languages such as $A=\{a^n b^n c^n\}$ $n \geq 0$. According to Church-Turin, every problem that can be solved on a real computer can also be solved by a turing machine.



B = Blank Symbol.

Formal definition:

A Turing Machine TM is defined by the Seven-tuples, $M=\{Q, \Sigma, \delta, q_0, F, T, B\}$ where

Q = Finite set of states

Σ = Finite set of input symbols

q_0 = Start state $q_0 \in Q$

F = set of Final states. $F \subseteq Q$

T = Tape symbols, Σ is always subset of T

B = Blank symbol. $B \in T$ but $B \notin \Sigma$

δ = Transition function defined by

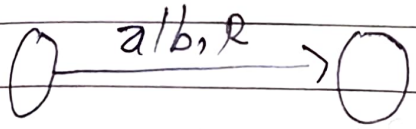
$$Q \times T \rightarrow Q \times T \times (R, L, S)$$

where R, L, S is the direction of movement of head. ie. Left, Right or Stationary

DATE

→ Rules of operation:-

At each step of the computation
- Read the current symbol
- Write the same cell
- Move exactly one cell either LEFT or RIGHT



where, a = denotes the symbol we are going to read
b = denotes the symbol we are going to write.
R = denotes the direction. (L or R)

There are:
- Initial state
- Final states; (two final states)
  - i) The ACCEPT state
  - ii) The REJECT state.
- Computation can either
  - i) HALT and ACCEPT
  - ii) HALT and REJECT
  - iii) LOOP (machine fails to HALT)

* Instantaneous Description for TM

A configuration of TM is described by ID of TM as in PDA. A string $x_1 x_2 \ldots x_{i-1} q x_i x_{i+1} \ldots x_n$ represents the ID of TM in which
- → q is the state of TM
- → the tape head scanning the i^{th} symbol from the left
- → $x_1 x_2 \ldots x_n$ is the portion of tape between the leftmost & rightmost non blank

DATE

* Language of Turing Machine

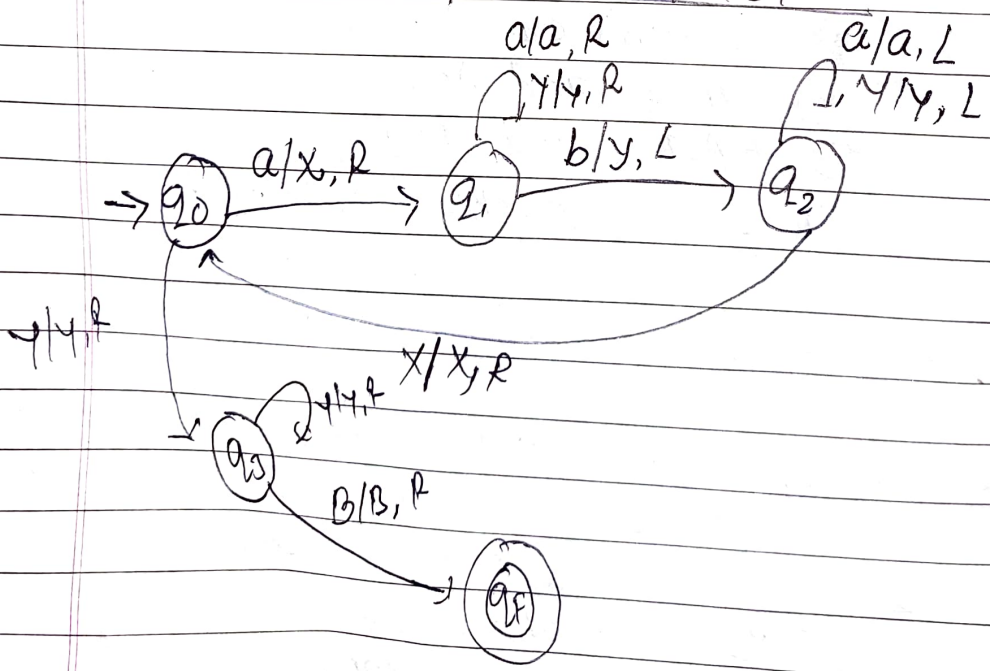
If $T = \{Q, \Sigma, \delta, q_0, F, \Gamma, B\}$ is a turing machine and $w \in \Sigma^*$, then language accepted by T, $L(T) = \{w \mid w \in \Sigma^* \text{ and } q_0 w \vdash^* \alpha p \beta\}$ for some $p \in F$ and any type of strin α and β

The set of languages that can be accepted by TM (using) are called recursively enumerable language

* Some examples of TM.

Eg.1. $L = \{a^n b^n \mid n \geq 1\}$

: B B a a a b b B B --



$\therefore TM = (\{q_0, q_1, q_2, q_3, q_F\}, \{a, b\}, \delta, q_0, \{q_F\})$

Q Σ F

$\{a, b, X, Y, B\}$, B)

classmate

PAGE

DATE

Let's make transition table for the moves

	a	b	X	y	B
q_0	q_1, X, R			q_3, y, R	
q_1	q_1, a, R	q_2, y, L		q_1, y, R	
q_2	q_2, a, L		q_0, X, R	q_2, y, L	
q_3				q_3, y, R	q_F, B, R
q_F					

Also, transition function δ can be defined as

$\delta(q_0, a) = (q_1, X, R)$
$\delta(q_0, y) = (q_3, y, R)$
$\delta(q_2, y) = (q_2, y, L)$
⋮
so on.

Now, let's see the acceptance of string aabb by T

$q_0 aabb \vdash X q_1 abb$
$\vdash X a q_1 bb$
$\vdash X q_2 a y b$
$\vdash q_2 X a y b$
$\vdash X q_0 a Y b$
$\vdash X X q_1 y b$
$\vdash X X Y q_1 b$
$\vdash X X q_2 Y Y$
$\vdash X q_2 X Y Y$
$\vdash X X q_0 Y Y$
$\vdash X X Y q_3 Y$
$\vdash X X Y Y q_3 B$
$\vdash X X Y Y B q_F B$ ⇒ Halt & accept

Let's Check aba

$q_0 aba \vdash X q_1 ba$
$\vdash q_2 X y a$
$\vdash X q_0 y a$
$\vdash X y q_3 a$

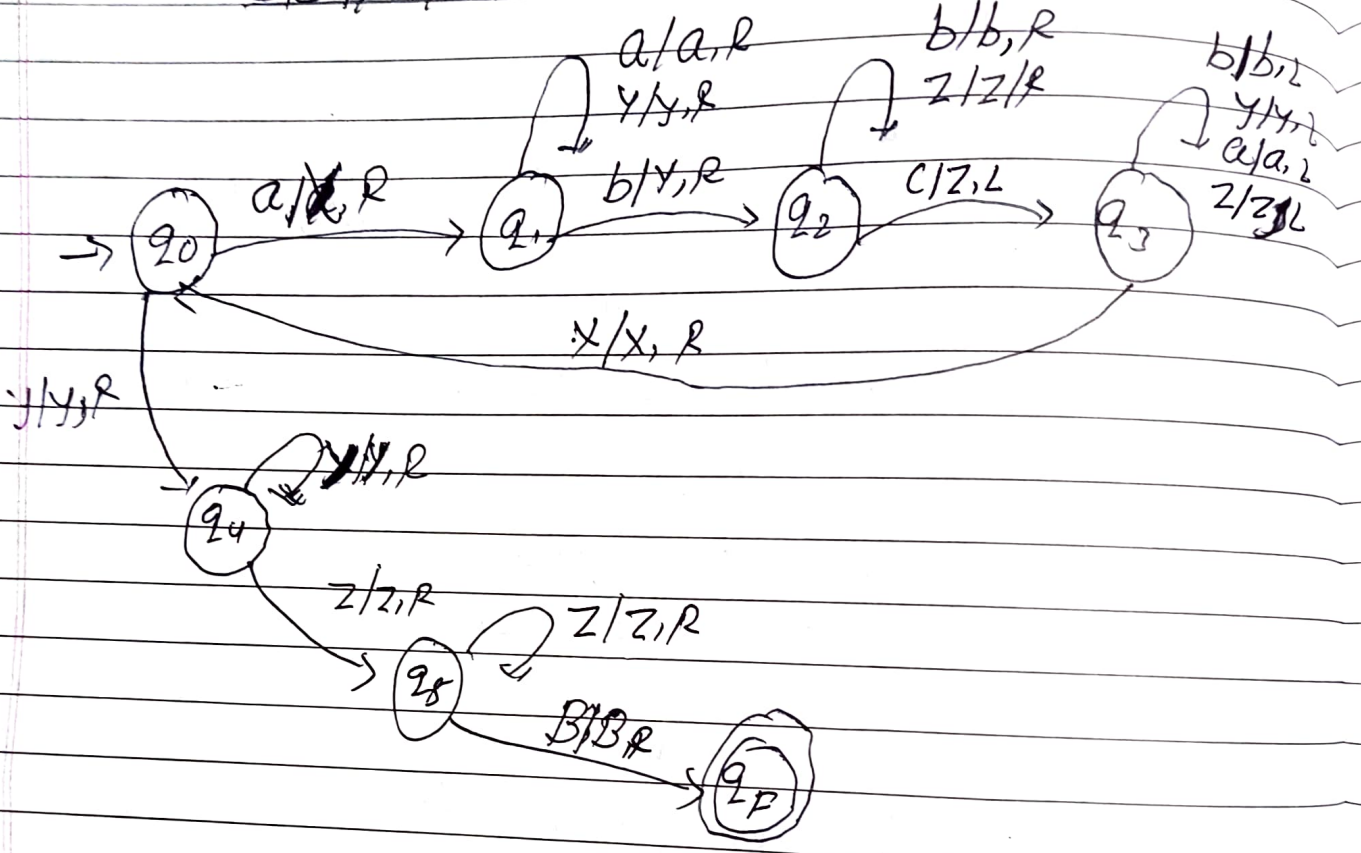
Halt & reject as q_3 has no move on symbol a

PAGE

DATE

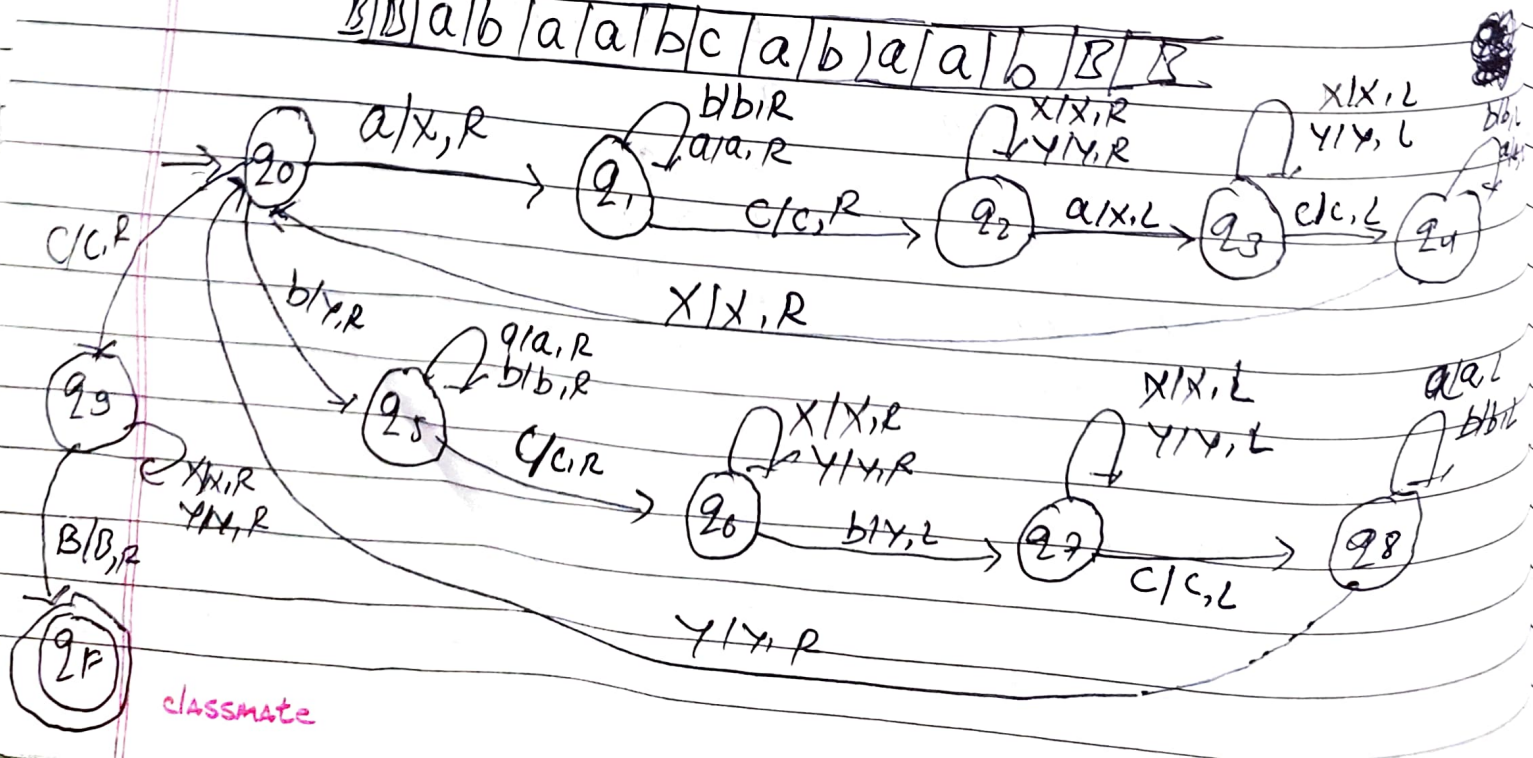
Ex-2 $L = \{ a^n b^n c^n \mid n \geq 1 \}$

		X	X	X	Y	Y		Z	Z	Z			
B	B	a	a	a	b	b	b	c	c	c	B	B	



Ex: $L = \{ wcw \mid w \in \{a,b\}^* \}$ (odd palindrome)

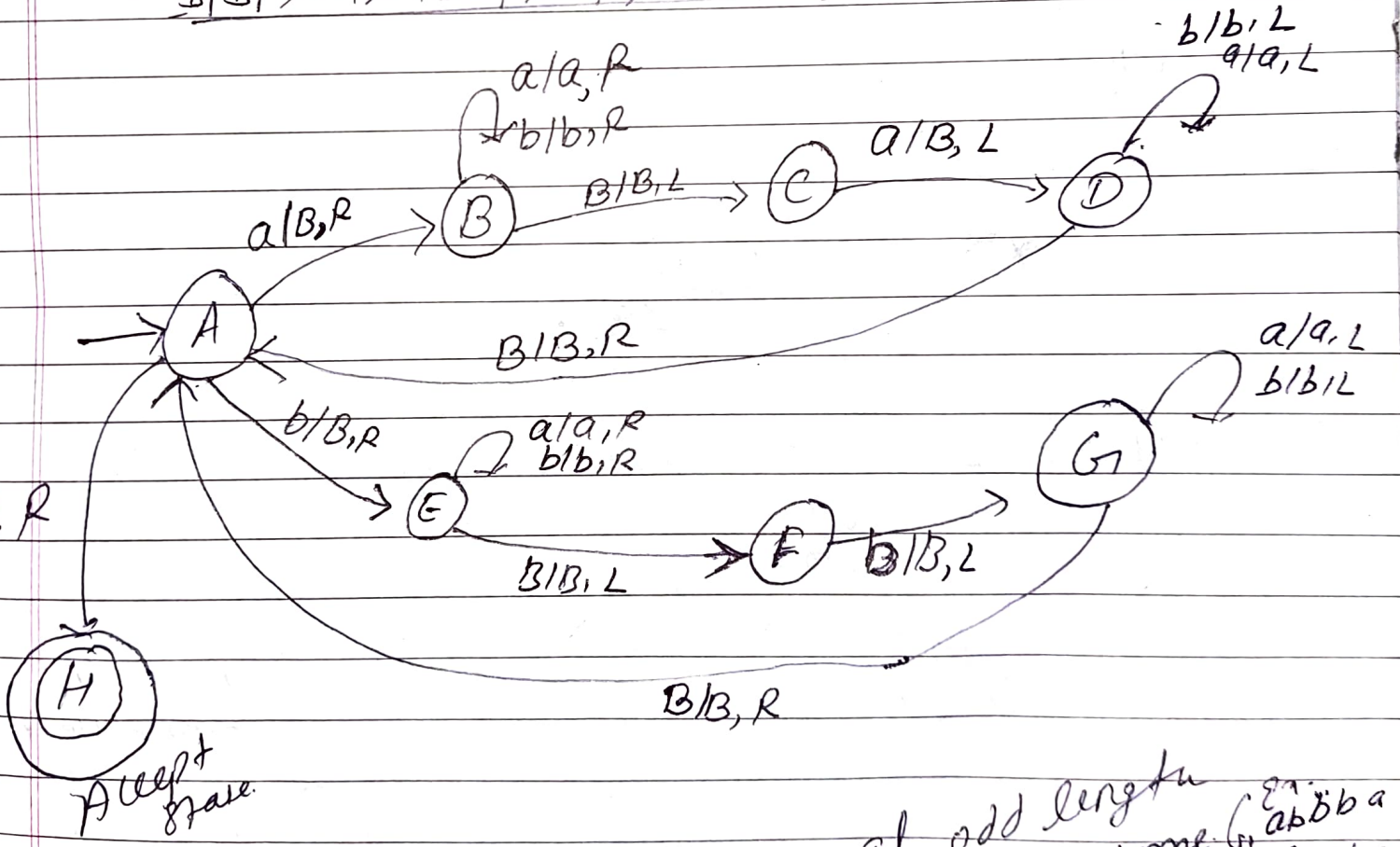
B	B	a	b	a	a	b	c	a	b	a	a	b	B	B



Ex: $L = \{ WW^R : W \in \{a,b\}^* \}$ (Even palindrome)

B B B B B B

- - B B a b a a b a B B ...



Note: If odd length palindrome (Ex: abbba, ababa) then, Join C and F to final state with B/B, R

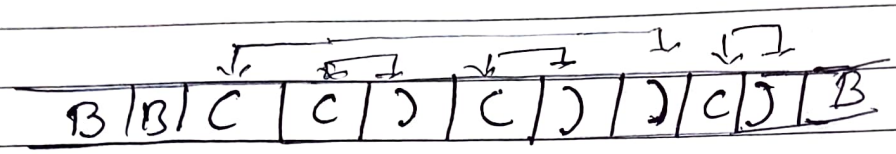
Note: If asked palindrome only, then combine both

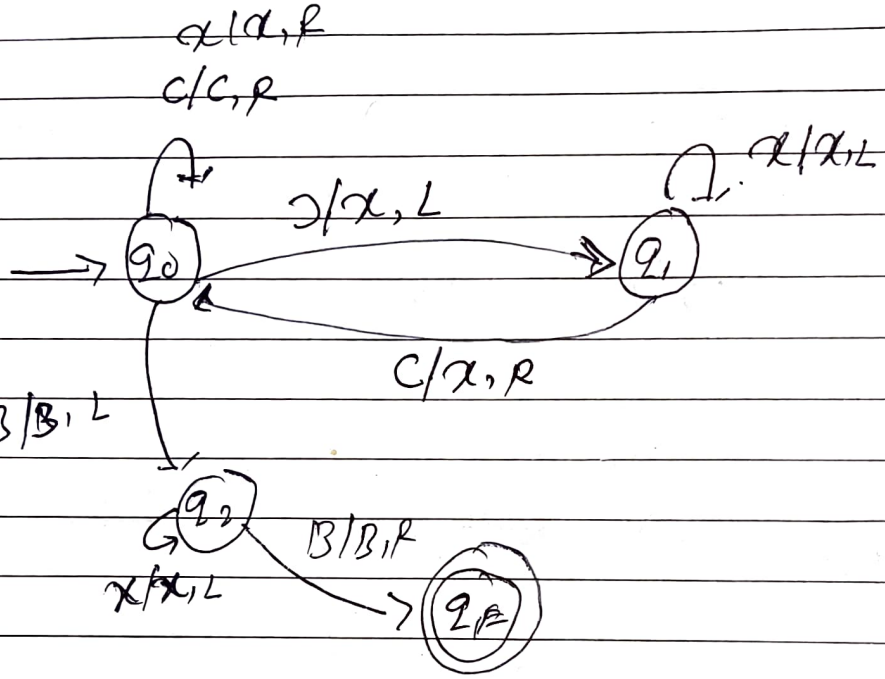
DATE

Ex: TM that accepts well formed string of parenthesis ie. balanced parenthesis.

$L = \{(), (()), ()(()), (())(), \ldots\}$

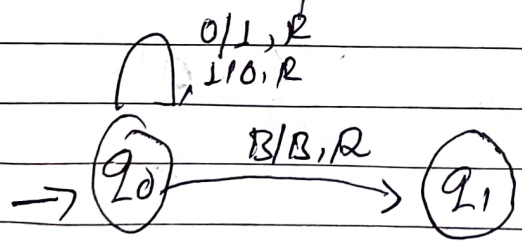
=)





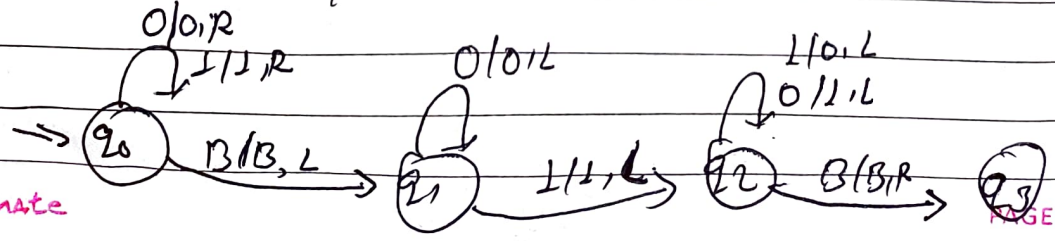
Ex: TM for 1's Complement : Let's take 1101

1's 1's is 0010



Note: We didn't use final state because we are not accepting anything, we are just converting input to output.

Ex TM for 2's complement : Let's take: 01011000 2's 2's is 10101000



classmate

PAGE

DATE

* Roles of Turing Machine

1. Turing Machine as a Language Recognizer.

A turing Machine can be used as a language recognizer to accept strings of certain language. Ex: A TM accepting $\{0^n 1^n \mid n \geq 1\}$ as we discussed earlier.

A Turing Machine T recognizes a string x (over Σ) if and only if when T starts in the initial position and x is written on the tape, T halts in a final state. A turing machine T does not recognize a string x, if T does not halt in a state that is not final

2. Turing machine as a Computing Function

A TM can be used to compute functions. For such TM, we adopt the following policy to input any string to the TM which is an input of the computation function.

i) The string 'w' is presented into the form BWB, where B is blank symbol, and placed onto the tape. The head of TM is positioned at a blank symbol which immediately follows the string w.

ii) The TM is said to halt on input w if we can reach to the halting state performing some operation

DATE

Definition:

A function $f(x) = y$ is said to be computable by a TM $(Q, \Sigma, \delta, q_0, \Gamma, B, \{q_f\})$ if $(q_0, BxB) \vdash^* (q_f, ByB)$, where x may be in some alphabet Σ_1^* and y may be in some alphabet Σ_2^* and $\Sigma_1, \Sigma_2 \subseteq \Sigma$

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

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

Sol^n Given function, $f(x) = x + 1$ - Here, we represent input x on the tape by a number of 1's on the tape.

For example, for $x = 1$, input will be BIB

for $x = 2$, input will be BIIB

for $x = 3$, input will be BIIIB & so on.

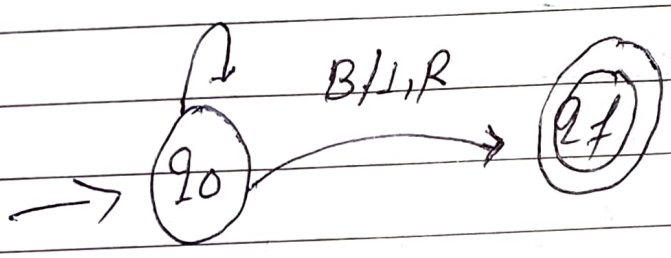
Similarly, output can be seen by the number of 1s on the tape when machine halts.

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

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

halt state q_f

B|B,R
1|1,R



Let, input $x = 2$, i.e the input tape contains B11B

So, q_0 B11B ⊢ B$q_0$11B ⊢ B1$q_0$1B ⊢ B11q_0B

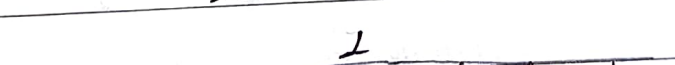
⊢ B111q_0B

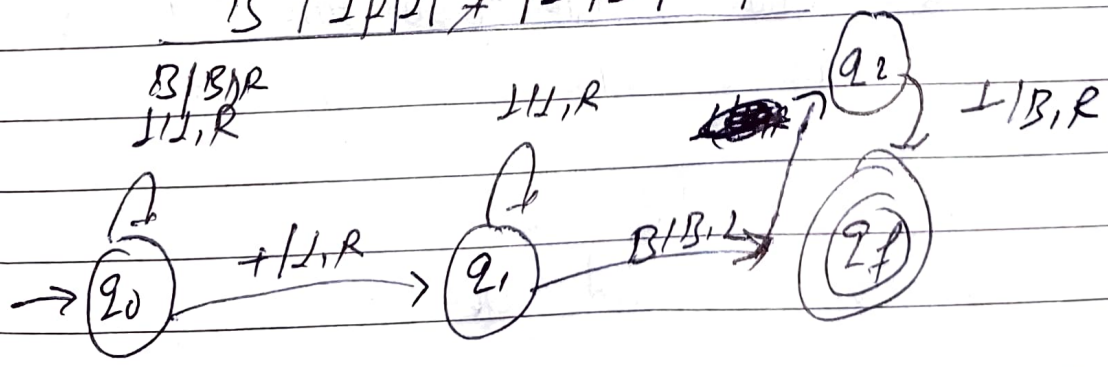
which means output is 3, as it contains 3 1's.

Ex-2. Let's take $f = x + y$ over $\Sigma = \{1\}$

Let $x = 2, y = 3$ then

$11 + 111 = 11111$



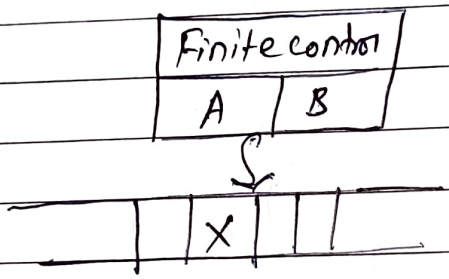


Note: No need to write B/B,R - ~~ϕ~~

DATE

3. Turing Machine with storage in its state.

In TM, a state can be used to hold a finite amount of data. The finite control of machine consists of a state q & some data portion. In this case, a state is considered as a tuple - (state, data)



δ is defined by:

$\delta([q,A], X) = ([q_1, X], Y, R)$ means q is the state and data portion of q is A. The symbol scanned on the tape is copied into the second component of the state & moves right entering state q_1 & replacing the symbol by Y.

4. Turing Machine as a enumerator of strings of a Language

Consider a multi-tape TM that uses a tape as an output tape, in which a symbol once written can never be changed, & those whose tape head never moves left. Suppose also that, on the output tape, TM writes strings over some alphabet Σ separated by a marker symbol #.

We can define L(TM) to be set of w in Σ^* such that w is eventually printed between a pair of #'s on the output tape. Thus the language of this type of TM is called Turing Enumerable languages

PAGE

Note: Transducer:- It means it gives output to a given input. ie. input conversion to output

DATE

5. Turing Machine as Subroutine.

Subroutines are collection of interacting components. A TM subroutine is a set of states that perform some useful process. This set of states include a start state and other states that serve as the return to pass control to another set of states called subroutine. The "call" of a subroutine occurs whenever there is a transition to its initial state

Ex: A TM that accepts even & odd paindrome

* In Summary, the roles of TM are:

1) As a language recognizer:- TM can be used for accepting a language like FA and PDA.

2) As a computer of function:- 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.

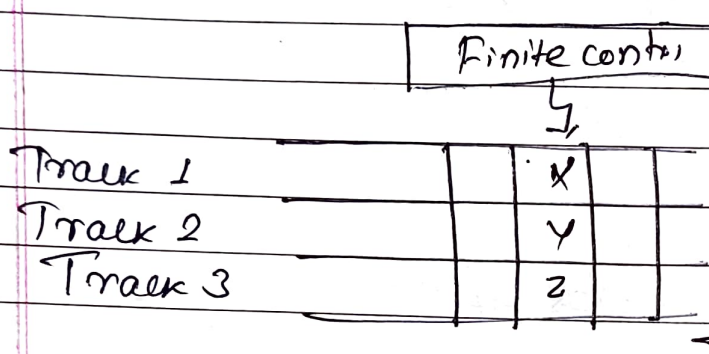
3) As an enumerator of string of a language:- It outputs the strings of a language, one at a time in some systematic order that is as a list.

PAGE

DATE

* Turing Machine with Multiple Tracks.

The tape of TM can be considered as having multiple tracks. Each track can hold one symbol, & the tape alphabet of the TM consists of tuples, with one component for each track. Following figure illustrates the TM with multiple tracks:

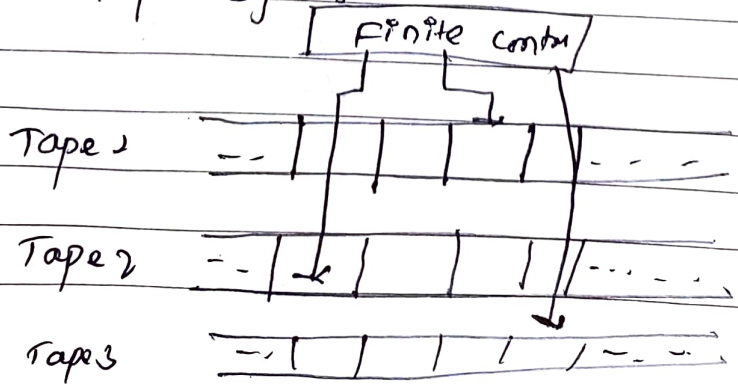


The tape alphabet Γ is a set consisting of tuples like $\Gamma = \{(x,y,z),\ldots\}$

The tape head moves up & down scanning symbols in tape at one position

* Turing machine with Multiple tapes

A TM can have more than one tape. Adding an extra tape adds no power to the computational model, but only the ability to accept the language is increased. A multitape TM consists of finite control and finite number of tapes. Each tape is divided into cells & each cell can hold any symbol of finite tape alphabets. The set of tape symbols include a blank and the input symbol



DATE

In the multi-tape, initially

⇒ Input w is placed on the 1st tape
⇒ All other cells of the tapes hold blanks
⇒ TM is in initial state
⇒ The head of the first tape is at the left end of input
⇒ All other tape heads are at some arbitary cell.
Since all tapes except 1st tape consists completely blank

Equivalence of Multitape-TM & Multitrack-TM.
or (Simulating one tape TM with multi-tape TM)

Theorem: Every language accepted by a multitape TM is recursively enumerable

OR

Any languages that are accepted by a multitape TM are also accepted by one tape TM.

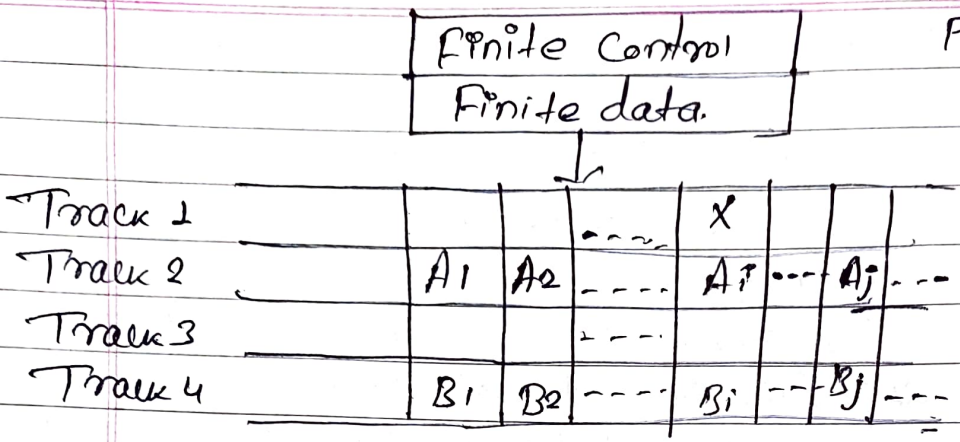
Proof: Let L be a language accepted by n-tape turing machine, T_1. Now, we have to simulate T_1 with a one-tape TM, T_2 considering there are 2n tracks in the tape of T_2.

For simplicity, let's assume n=2, then for simulating 2-tape turing machine, we need a tape having 4-tracks. The 2nd & 4th tracks holds the contents of the 1st & 2nd tapes of T_1, track 1st holds the position of the head of tape 1, & track 3rd holds the position of the 2nd tape head.

PAGE

classmate

Let's assume n=2, then for n>2 is the generalization of this case. Then, total number of tracks in T2 will be 4. The 2nd & 4th tracks of T2 hold the contents of 1st & 2nd tapes of T1. The track in T2 holds head position of tape 1 of T1 & track 3 in T2 holds head position of tape 2 of T1.



Now, to stimulate the Move of T1.

→ T2's head must visit the n-head markers so that it must remember how many markers are to its left at all times. That count is stored as a component of T2's finite control

→ After visiting & storing scanned symbol, T2 knows what tape symbols are being scanned by each of T1's head.

→ T2 also knows the state of T1, which it stores in T2's own finite control. Thus T2 knows what move T1 will make.

→ T2 now revisits each of the head markers on its tape, changes the symbol in track representing corresponding tapes T1 & moves the head marker left or right if necessary.

Finally, T2 changes the state of T1 as recorded in its own finite control. Hence T2 has simulated one move of T1. We select T2's accepting states all those states that record T1's state as one of the accepting state of T1. Hence, whatever T1 accepts T2 also accepts.

DATE

Non-Deterministic Turing Machine.

A non-deterministic TM (NTM), $T = (Q, \Sigma, \delta, q_0, F, \Gamma, B)$ is defined exactly the same as an ordinary TM, except the value of transition function δ.
In NTM, the values of δ are subsets, rather than the single element of the set $Q \times \Gamma \times \{R, L, S\}$.
Here, the δ is such that for each state q & tape symbol x, $\delta(q, x)$ is a set of triples.

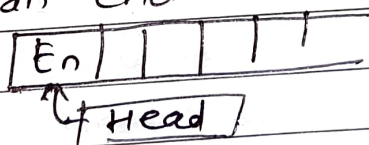
$$\{(q_1, Y_1, D_1), (q_2, Y_2, D_2), \ldots\ldots, (q_k, Y_k, D_k)\}$$

where k is any finite integer
The NTM can choose, at each step, any of the triples to be the next move.

Restricted Turing Machines

1. Semi-Infinite Tape:-

A Turing Machine with a semi-infinite tape has a left end but no right end. The left end is limited with an end marker.



It is also a two track type:-

i) Upper track: It represents the cells to the right of the initial head position
ii) Lower track: It represents the cells to the left of the initial head position in reverse order

Turing machines with semi-infinite tape are equivalent to standard Turing machines.

PAGE

DATE

* Church. Turing Thesis

It is a mathematically un-provable hypothesis about the computability. This hypothesis simply states that -

"Any algorithmic procedure that can be carried out at all can be carried out by TM".

This statement was first formulated by Alonzo Church a mathematician & a logician in 1936. According to him, "No computational procedure will be considered an algorithm unless it can be represented by a Turing Machine".

In more precised language, : An algorithm is a procedure that can be executed on a TM. Church's Thesis another use is that, when we want to describe a solution to a problem, we will often satisfy with a verbal description of the algorithm, translating it into detailed TM implementations.

* Universal Turing Machine

A machine that can simulate the behaviour of an arbitary TM is called a Universal TM. Thus, we can describe a universal Turing Machine TU as a TM, that on input $\langle M, w \rangle$; where M is a TM & N is a string of input alphabets, simulates the computation of M on input w and satisfies two properties

→ TU accepts $\langle M, w \rangle$ iff M accepts w

→ TU rejects $\langle M, w \rangle$ iff M rejects w

classmate

PAGE

DATE

* Encoding of Turing Machine (Binary Encoding)

As we know, universal turing machine is capable of simulating other turing machines. and the input given to the universal turing machine is in binary encoded format of the turing machine. ie. in the form of {0,1}.

Universal TM is denoted by ⟨M,w⟩ where M is the encoded form of the turing machine that the universal TM simulates and w is the input string.

An Example, Given a turing machine

TM = { {q0, qf}, {0,1}, {0,1,B}, δ(q0,0)=(q0,0,R),

(Q) (Σ) (Γ)

δ(q0,1)=(q0,1,R), δ(q0,B)=(qf,1,L)

(δ)

q0, B, qf }

(q0) (B) (qf)

Now, we are going to convert this to encoded format. whenever the TM enters into the accepting state on some input, the universal TM has to accept it. Here, each and every tuples of TM is represented in terms of number of ~~bytes~~ bits of {0,1}.

Let, the Encoding structure (function) is:

e(q0) = 1
e(qf) = 11
e(0) = 1
e(1) = 11
e(B) = 111
e(L) = 1
e(R) = 11

PAGE

classmate

Let '0' be the delimiter for elements within the tuple.
'00' be the delimiter for the tuples of the TM
So, the final description for universal TM in binary encoded form:

$\langle M, w \rangle \Rightarrow \langle$ 1011 00 1011 00 10110111 00 1010101010110 (labels: Q, Σ, Γ, δ(q0, 0) = (q0, 0, R))

10110101101011 0 101110110101 00 1 00 111 0011 (labels: δ(q0, 1) = (q0, 1, D), δ(q0, B) = (qf, 1, L), ⊔ = q0, ⊔ = B, qf)

∴ $\langle M, w \rangle$ = ⟨ 101110010110010110111001010101010110101101011010110101101011 0101110110101001001110011 , w ⟩

where w is the input string

Note: if input w = 001, then write its encoded form as 101011

Ex.2:- Let TM T is given as:

$T = (\{q_1, q_2, q_3, F\}, \{a, b\}, \{a, b, B\}, \delta, q_1, B, F)$

where δ is defined by & Input $w = ab$

$m_1 = \delta(q_1, b) = (q_3, a, R)$

$m_2 = \delta(q_3, a) = (q_1, b, R)$

$m_3 = \delta(q_3, b) = (q_2, a, R)$

$m_4 = \delta(q_3, B) = (q_3, b, L)$

⇒ Let the Encoding be.

$e(q_1) = 1$ $e(L) = 1$

$e(q_2) = 11$ $e(R) = 11$

$e(q_3) = 111$ $e(F) = 1111$

$e(a) = 1$

$e(b) = 11$

$e(B) = 111$

Now, $e(m_1) = 1 0 11 0 111 0 1 0 11$

$e(m_2) = 111 0 1 0 1 0 11 0 11$

$e(m_3) = 111 0 11 0 11 0 1 0 11$

$e(m_4) = 111 0 111 0 111 0 11 0 1$

$e(w) = 1 0 11$

So, final description for universal TM is:

<M, w> = <1 0 11 0 111 00 1011 00 1 0 11 0 111 00
1 011 0 11 1 0 1 0 11 0 111 01 01 01 1 011 0 11 0 11 0
11 0 1 011 0 111 011 0111 01101 00 1 00 111 00
01111 , 1011 >

DATE

* Turing Machines and Computers

The Turing Machines and computers both can accept the same, recursively enumerable languages. Since the notation of "a common computer" is not well defined mathematically, the arguments in this are necessarily informal. TMs and computers can be divided in to two parts:

i) A computer can simulate a Turing Machine
ii) A Turing Machine can simulate a computer & can do so in an amount of time that is at most some polynomial in the number of steps taken by the computer.

* Difference between TM & other Automata (FSA & PDA)

The most significant difference between the TM and other automata is that; In a TM, processing a string is no longer restricted to a single left to right pass through input.

The tape head can move in both directions & erase or modify any symbol it encounters. The machine can examine part of the input, modify it, take time to execute some computation in a different area of the tape, return to re-examine the input, repeat any of these actions and perhaps stop the processing before it has looked at an input.

DATE

Enumerating the Binary Strings

We shall need to assign integers to all the binary strings so that each string corresponds to one integer, & each integer corresponds to one string. If w is a binary string, then treat 1w as a binary integer i. Then we shall call w the ith string. That is ϵ is the first string, 0 is the second, 1 is the third, 00 the fourth, 01 the fifth & so on. Equivalently, strings are ordered by length, and strings of equal length are ordered lexicographically. Here after, we shall refer to the ith string as w_i.

Questions asked from this chapter

Q. Construct a Turing Machine that accepts the language of odd length strings over alphabet $\{a, b\}$. Give the complete encoding for TM as well as its input string $w = abb$ in binary alphabet that is recognized by universal Turing Machine. (2078 - 10 marks) (2067 - 5 marks)

Q. Define Turing machine and explain its variations. (2078 - 5 marks)

Note: Variations → multitape, multitrack, non-deterministic TM.

Q. Define a Turing Machine. Construct a TM that accept $L = \{wcw^R \mid w \in (0,1)^*\}$ and 'c' is $\in$ or '0' or '1'. Show that string 0110 is accepted by this TM with sequence of ID. (2076 - 10 marks)

Q. Describe the Turing machine with multiple tape, multiple track and storage in state. (2076 - 5 marks)

Q. Show that the TM with one tape & a TM with multiple tape are equivalent. (2071 - 5 marks)

Q. Describe universal TM & its operations. What type of languages are accepted by universal TM? (2067 - 5 marks, 2076 5 marks)

Q. Define TM. Construct TM for $L = \{a^n b^n \mid n \geq 0\}$ (2072 - 5 marks)

Q. Draw TM to accept palindromes over $\{a, b\}$ (2063 - 5 marks)

(2068 - 5 marks)

DATE (2069 - 5 marks)

Q. What is the role of TM? Explain. (2072 - 5 marks)

Q. Give a formal definition of TM. How it differs from FA & from PDA? (2067 - 5 marks) (2075 - 5 marks)

Q. Construct a TM for even length strings over {0,1}. (2076 - 5 marks)

Q. Explain non deterministic TM with example. (2069 - 5 marks)

Q. Explain about recursive enumerable & recursive language. (2070 - 5 marks)