Turing machine

Limitations of FA :- ① Finite amount of memory

② cannot perform comparsion

$L = \{a^n \mid n \geq 0\}$

Limitations of PDA :- ① It can perform only one comparsion

PDA = FA + 1stack

$L = \{a^n b^n \mid n \geq 0\}$

Turing machine (TM) :- Abstract computer

TM = FA + nstacks

↳ by Alan Turing in 1936

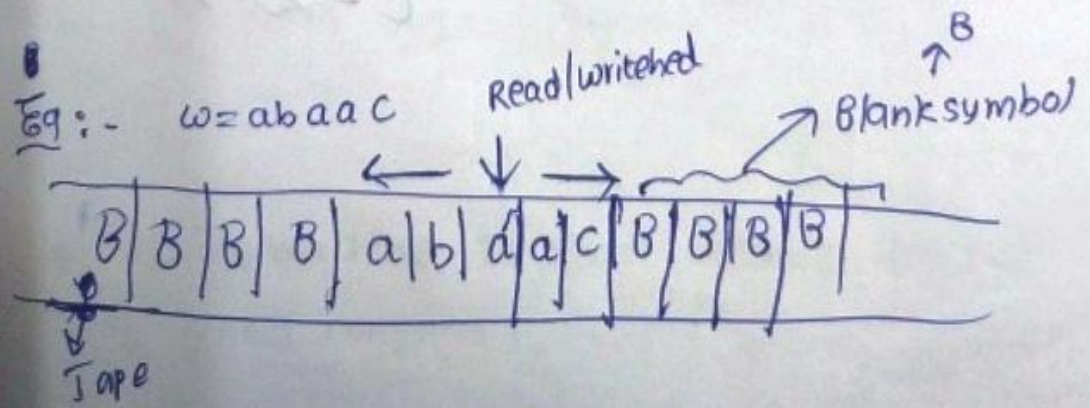
* Turing award - highest award in CSE field

TM < DTM — Determenstic TC
NTM — Non-determenstic TC

* By default PDA = NDA but TM = DTM

Turing machine has a tape of infinite length. Tape is divided into number of cells. Each cell will contain input strings.

Eg :- $\omega = abaac$



* In TM we can move in both directions i.e., either left or right

Functionality of R/W head :- (R/W → Read write)

① Read the current input symbol

② Read/update the current input symbol

③ move either left / right direction

$w = \{a^n b^n \mid n \geq 0\}$

Replaces a with X b with Y

$w = aabb$

B	B	B	~~a~~ X	~~a~~ X	~~b~~ Y	~~b~~ Y	B	B	B	B

↑ intially R/W points

First replace a with X then b with Y then again a with X b with Y after blank

$w = aabbb$

B	B	~~a~~ X	~~a~~ X	~~b~~ Y	~~b~~ Y	b	B	B

↑ can't perform any operation

suppose

$L = \{a^n b^n c^n \mid n \geq 0\}$

a — X
b — Y
c — Z
} replace

ctions i-e -, either

symbol

ion

b with y

first replace a with X
then b with y
then again a with X
b with y
after blank

n any operation

— X
- y
- Z
} replace

Formal definition of TM :- TM has 7 tuples

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

$Q \rightarrow$ Finite no. of states

$\Sigma \rightarrow$ set of alphabets

$\Gamma \rightarrow$ set of Tape symbols $\rightarrow \{\Sigma \cup B\}$

$B \rightarrow$ Blank symbol $B \in \Gamma$

$F \rightarrow$ set of final sets $F \subseteq Q$

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

L – left direction

R – Right "

$(q_0, a) \rightarrow (q_1, X, R)$ → right direction

→ replaced symbol

$q_0 \xrightarrow{a, X, R} q_1$

Two types of DTM :-

① DTM Eg :- $q_0 \xrightarrow{a, X, R} q_1$

② NTM Eg :- $q_0 \xrightarrow{a, X, R} q_1$

$q_0 \xrightarrow{a, Y, L} q_2$

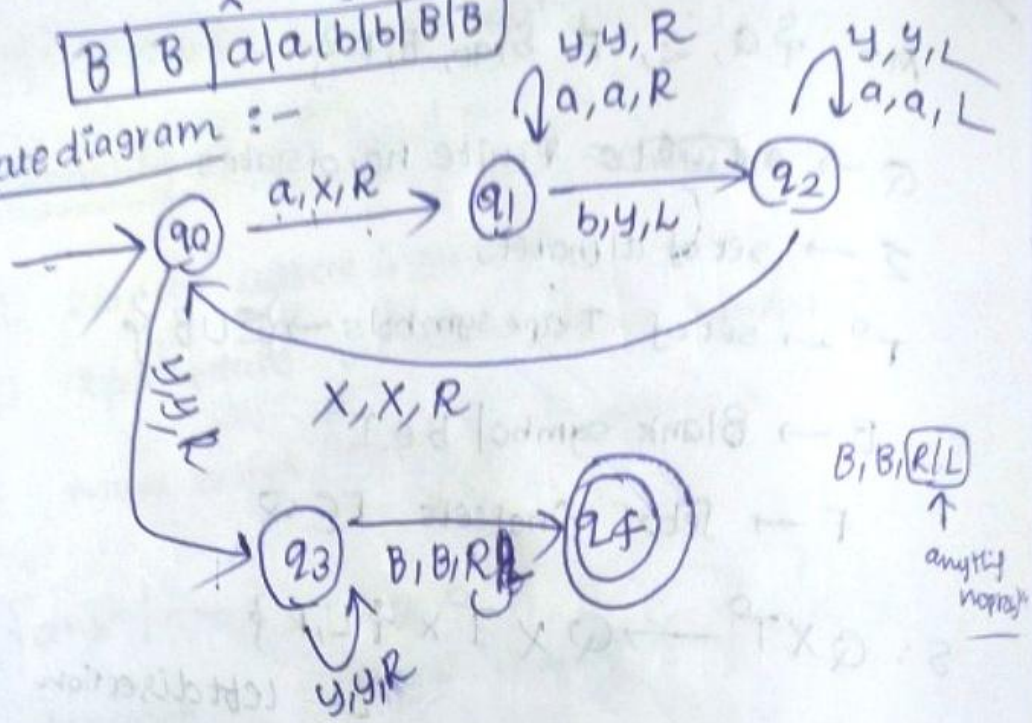
* Turing machine is most accurate model of general computers.

Eg:-

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

B	B	a	a	b	b	B	B
		x		y			

State diagram :-



a	a	a	b	b	b	B	B

~~a~~	~~a~~	~~a~~	~~b~~	B

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

$\Sigma = \{a, b\}$ $\quad T = \{a, b, X, y, B\}$

$q_0 = q_0$ $\quad B = B$ $\quad F = q_4$

Transition function:-

$\delta(Q \times T^{o}) \longrightarrow (Q \times T^{o} \times L, R)$

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

$\delta(q_0, y) \longrightarrow (q_3, y, R)$

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

$\delta(q_1, b) \longrightarrow (q_2, y, L)$

$\delta(q_1, y) \longrightarrow (q_1, y, R)$

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

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

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

$\delta(q_3, y) \longrightarrow (q_3, y, R)$

$\delta(q_3, B) \longrightarrow (q_4, B, R)$

Transition table:-

	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_4, B, R)
q_4	–	–	–	–	–

Configuration of Turing machine :-

It contains 3 items :- 1. current state
2. current tape content
3. current R/W head position

C_1 | B | a | a | b | b | B | $\Rightarrow$ | B | X | a | b | b | B | C_2

↑ current R/W head position

for state q, strings u, v over Σ

we write u q v q → current state
current tape content (u,v)
current head position is 1st symbol of string v

For Eg :- | X | a | y | b |

↑ current head position

we write (X, a, q y, b)

C_1 yields C_2 if TM can go from $C_1 \to C_2$ in one step $a, b, c \in T^0$ $u, v \in T^{0*}$

| u | a | c | v | q_i, q_j

↑ q_j

u a q_i b v yields u q_j a c v

if $(q_i, b) \to (q_j, c, L)$

state

tape content

head position

b|b|B C_2

sent R|ω head position

tstate

content (u,v)

l position is 1st

l of string v

osition

b)

$C_1 \to C_2$ in

T^*

u q_ja cV

a_j, c, L)

$u\,a\,q_i b V \to u\,a\,c\,a_j V$

$(q_i, b) \to (a_j, c, R)$

u	a	c	q_j V

u	a	b	V

$\downarrow q_j$

$\omega = ab$

$\downarrow q_0$

	a	b	

$\Rightarrow$

q_1

X	b	

$\uparrow$

⓪ $q_0 ab \vdash X q_1 b$ ①

$\vdash q_2 X y$ ②

$\vdash X q_0 y$ ③

$\vdash X y q_3 \cdot B$

$\vdash X y B q_4 \cdot B$

After consuming a,b we reached final state

string ab is accepted by Turing machine

$\omega = abb$ $\therefore$ abb $\to q_3$ it will halt

$q_0 abb \vdash X q_1 \cdot bb$

$\vdash q_2 \cdot X y b$

$\vdash X q_0 \cdot y b$

$\vdash X y q_3 \cdot b$

There is no transition for b in q_3. So, we halt in non-final state. abb is not accepted by TM

aab

$q_0 a\, ab \vdash X\, q_1 \cdot a \cdot b$

$\vdash X\, a \cdot q_1 \cdot b$

$\vdash X\, q_2 \cdot a \cdot y$

$\vdash a\, y$

$q_2 \cdot X$

$\vdash X\, q_0 \cdot a\, y$

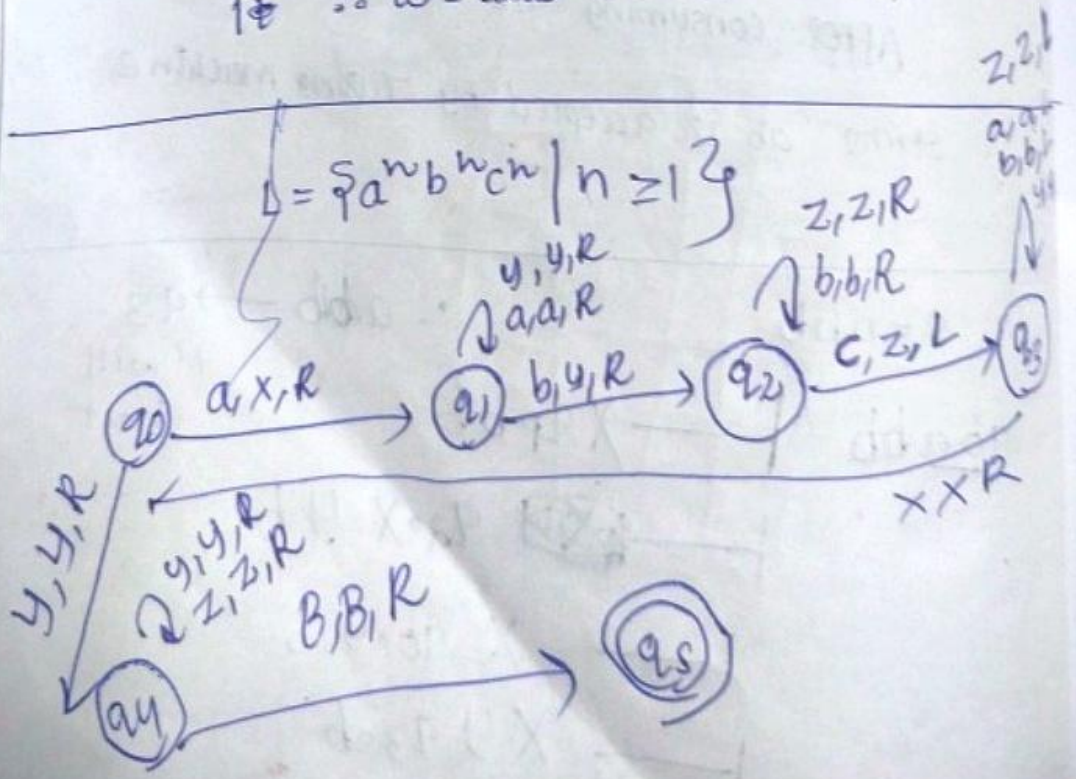
$\vdash X X\, q_1 \cdot y$

$\vdash X X y\, q_1 \cdot B$

There is no transition for "B" in q_1

$\therefore \omega = aab$ will halt in q_1

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



$\omega = ab$

$q_0 \cdot a\ b \vdash X\ q_1 b$

$\vdash X Y\ q_2 \cdot B$

initial configuration

Halts in q_2 which is non-final state. so

$\omega = abc$

$q_0 \cdot abc \vdash X\ q_1 b\ c$

$\vdash X Y\ q_2 \cdot c$

$\vdash X Y Z\ q_3$

$\vdash q_3 X Y Z$

$\vdash X\ q_0 \cdot Y Z$

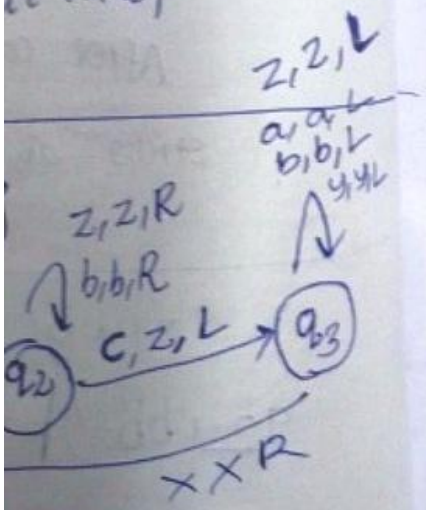
$\vdash X Y\ q_4 \cdot Z$

$\vdash X Y Z\ q_4 B$

$\vdash X Y Z\ B\ q_4 \cdot B$

initial configuration

final/accepting configuration



$\omega = aaabbbccc$ → 22 steps

$q_0 \cdot a\ aabbb$

Chruch Turing Thesis

① Anything that can be done in existing computer can also be done by Turing machine.

② No one yet able to suggest a problem that can be solved by algorithm for which TM can't be designed.

③ Alternative modes has been proposed for mechanic computing but none of them is more powerful than Turing machine.

Tuples of TM : $M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$

$L(M)$ = Language of TM

$= \{ \omega \in \Sigma^* \mid q_0\omega \vdash^* \alpha p \beta$

where $p \in F$

$\alpha, \beta \in \Gamma^*$ (Tape w

* Lang. accepted by TM is called Turing Recognizable lang.

Recursive lang.

ω → TM

(R.L)

Recursive lang. :-

TM will halt at

TM will halt a

L as recursive

Recursive enumerat lang :-

TM will halt

TM will halt

LOOP.

* TM designed

& Total TM

* TM designed

non Hal

steps

hesis

existing computer

machine.

problem that can

which TM can't be

oposed for mechanical

s more powerful than

, q0, B, F}

M

q0w ⊢* αpβ

where p∈F

α, β ∈ T*

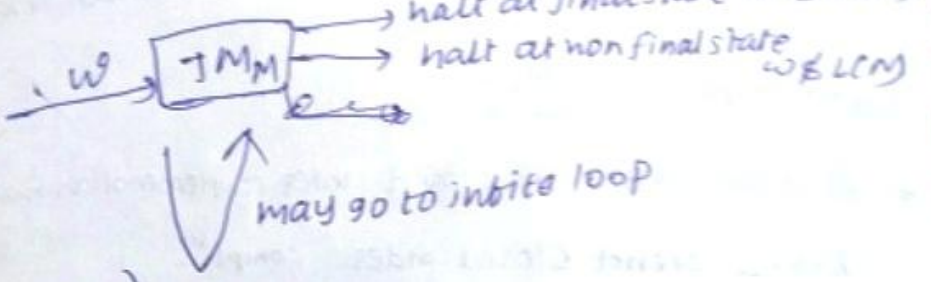
(Tape wise)

ued Turing

Turing Recognizable Lang.

Recursive lang.

Recursive enumerable lang.



(R.L)

Recursive lang. :- If L is a lang., for any string w∈L, TM will halt at final state & for any w∉L, TM will halt at non-final state, then we call L as recursive lang.

Recursive enumerable lang :- If L is a lang, for any string w∈L, TM will halt at final state & for any w∉L TM will halt at non-final state & go infinite loop.

* TM designed for RL is known as Halting TM & Total TM
* TM designed for REL is known as non Halting TM

* NON REL are the lang for which we can't Turing machine (undeciadble prob'm
↳ no algm exist)

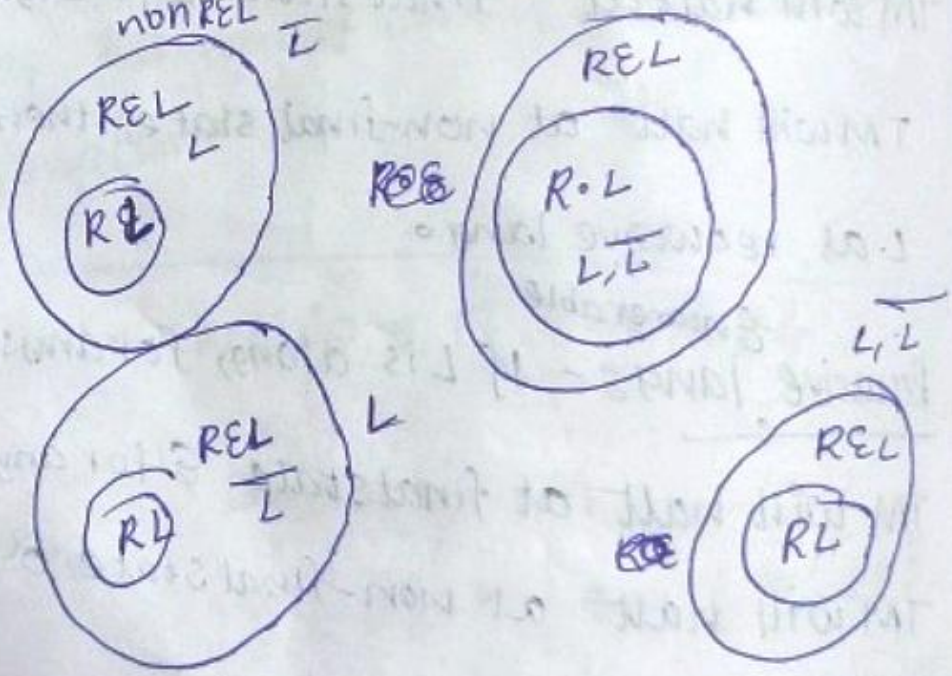
Important ~~pair~~ properties :-

* Recursive lang's are closed under complemenation but R.E.L arenot closed under compl^n.

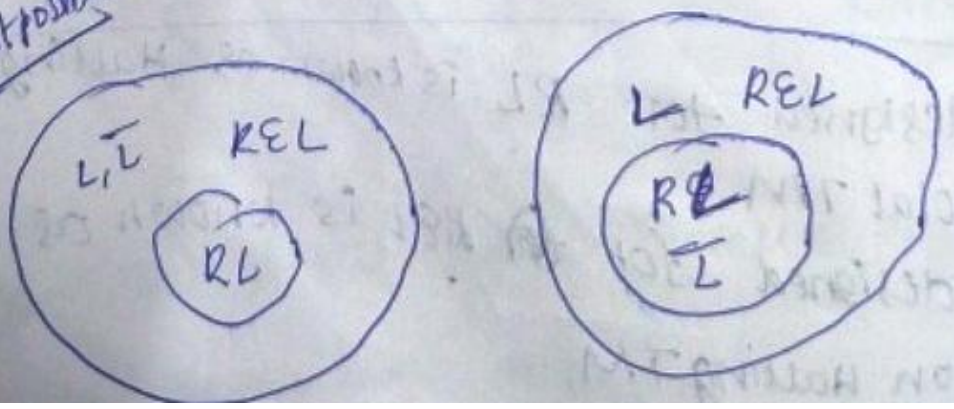
If L is R. Lang, $\overline{L}$ is also Recursive.

If L & $\overline{L}$ are R.E.L, then $\overline{L}$ must be Recu.L

Possible :-



not possible



Variants of Turing machine :-

Standard TM :- It has a tape of infinite length (in both directions)

B	B	a	b	B	B

R/W head can move in left & right direction

Turing machine with stay option :-

$$\delta(Q \times T) \longrightarrow (Q \times T \times (L, R, S))$$

$$(q_0, a) \longrightarrow \in (q, L/R/S)$$

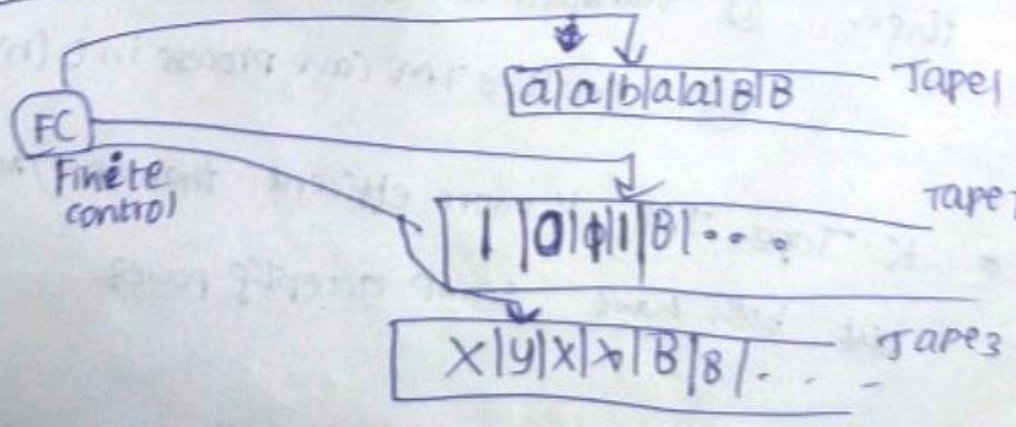
* The TM has same power as that of standard TM (same powers)

Turing machine with semi ∞ tape :- This TM has a tape which is infinite in one direction (either left & right)

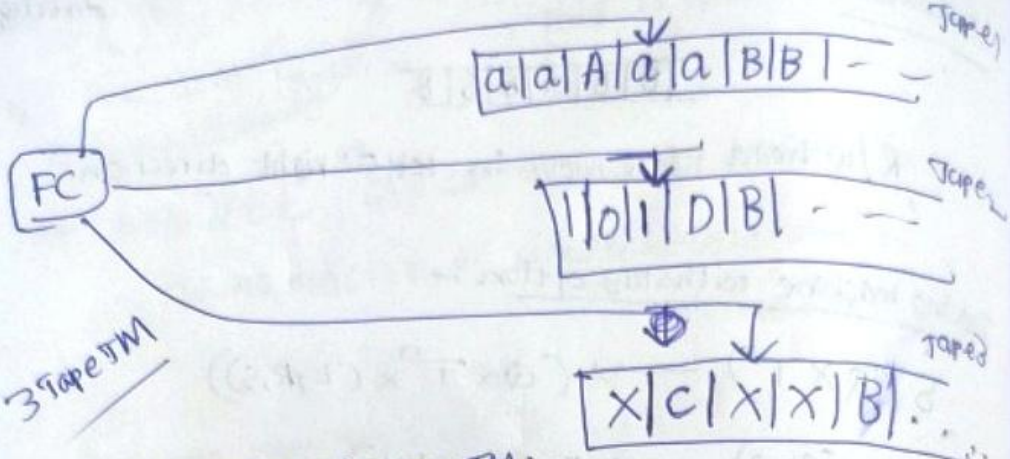
a	b	c	B	B	...

* It has same power as that of standard TM

Multitape TM :- This TM has more than one tape



$\delta(q_0, b|y) \longrightarrow (q_1, ADC, RLR)$



* Transition fun. of k tape TM :-

$\delta(Q, \Gamma^*) \longrightarrow (Q, X\ \Gamma^* X\ (L, R)^*)$

$\delta(Q, a_1 a_2 a_3 \cdots a_k) \longrightarrow Q\ X(b_1 b_2 \cdots b_k)\ X(L,R)^*$

* Power of Multi Tape TM is same as that of Stand.

* But Multi Tape TM is more efficecent that Stand. TM (time to accept)

$L = \{a^n b^n \mid n \geq 1\} \longrightarrow$ "w" is string of length "n"

then ① standard TM can process in $O(n^2)$ mo

② Multipe TM can process in $O(n)$ mo

* K Tape TM is more efficent than (K-1) Multi TM but both have same accepting power

Therom :-

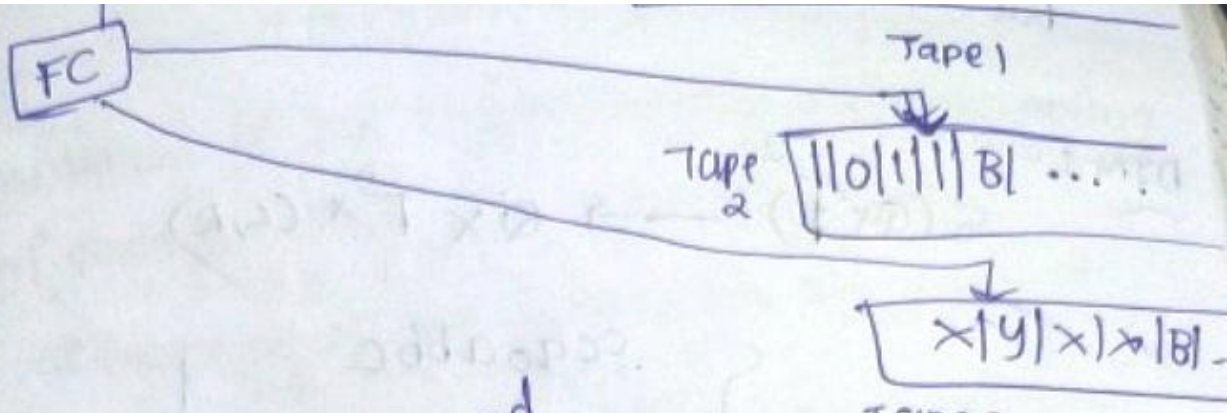
FC

$\delta(q_0,$

#	a	a	A

suppose Tap

#	a	a	A



Actual R/whead

Virtual R/whead

#	a	a	b*	a	a	#	1	0	1	1*	#	x	y*

#	B	-

$\delta(q_0, b/y) \longrightarrow (q_1, ADC, RLR)$

#	a	a	A	a*	a	#	1	0	1*	D	#	x	c	x*	x	#	B

suppose Tape1 reached Blank symbol

#	a	a	A	a	a	B	#	1	0	1	D	#	x	c	x	x	#

Non Deterministic Turing machine :-

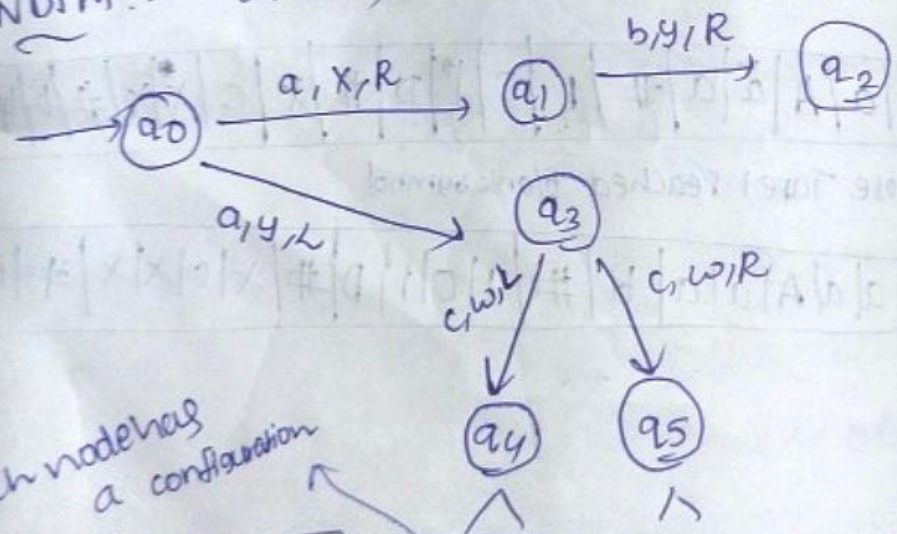
* DTM has same power as that of NDTM but DTM is more efficent than NDTM

DTM :- $\delta(Q \times \Gamma) \longrightarrow Q \times \Gamma \times (L, R)$

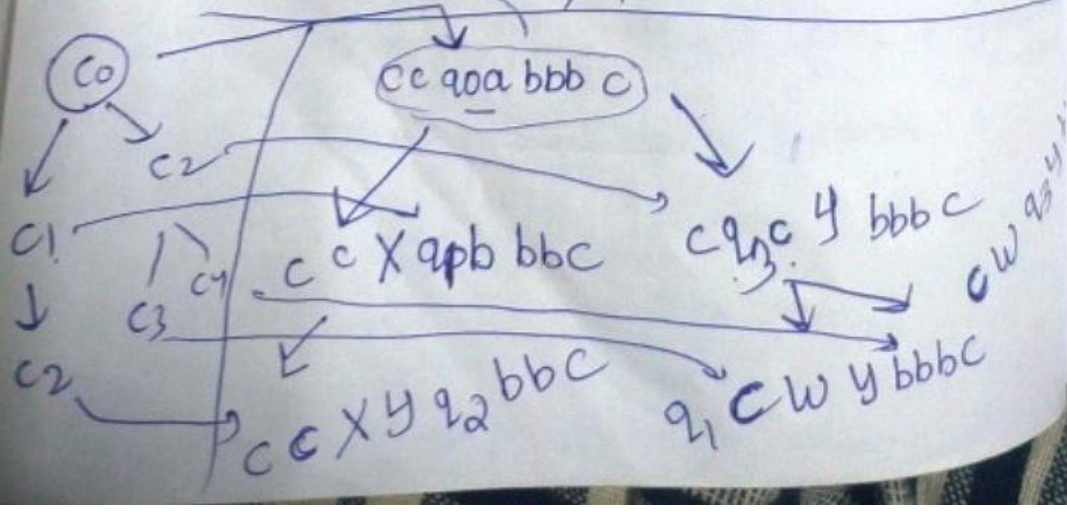
linear Tree ←

$cc q_0 abbc$
↓
$ccX q_1 bbc$
↓
$ccXY q_2 bc$

NDTM :- $\delta(Q \times \Gamma) \longrightarrow 2^{Q \times \Gamma \times (L,R)}$



Each node has a configuration

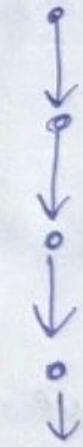


* ~~If Any b~~
* ~~Any bra~~

* If any branch configuratio
* If all bran nonfinal s

Loop :- compu is never tree are inf

DTM



accept/reje

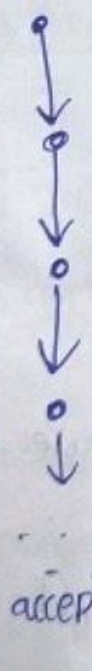
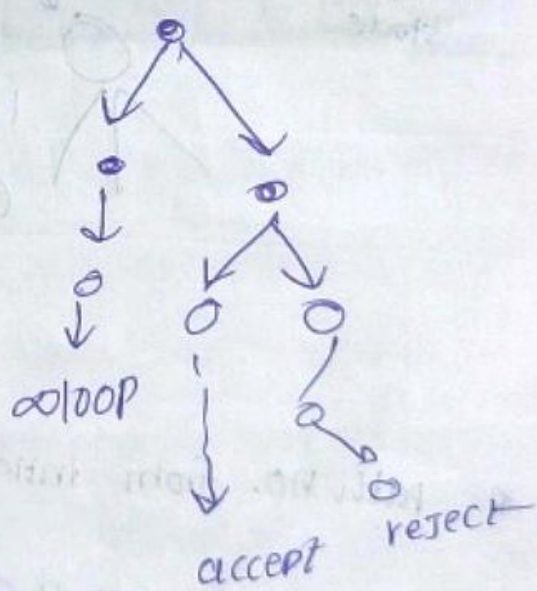
~~* If Any branch that has an accepting branch~~

~~* Any branch that has an a rejecting branch~~

* If any branch of the configuration tree leads to accepting configuration, then w is accepted by NDTM

* If all branches of the configuration tree halts at non final state w is rejected by TM (does not enter ∞ loop)

loop :- computation continues but accepting configuration is never encountered i.e., some branches in computational tree are infinite.

DTM	NDTM
	

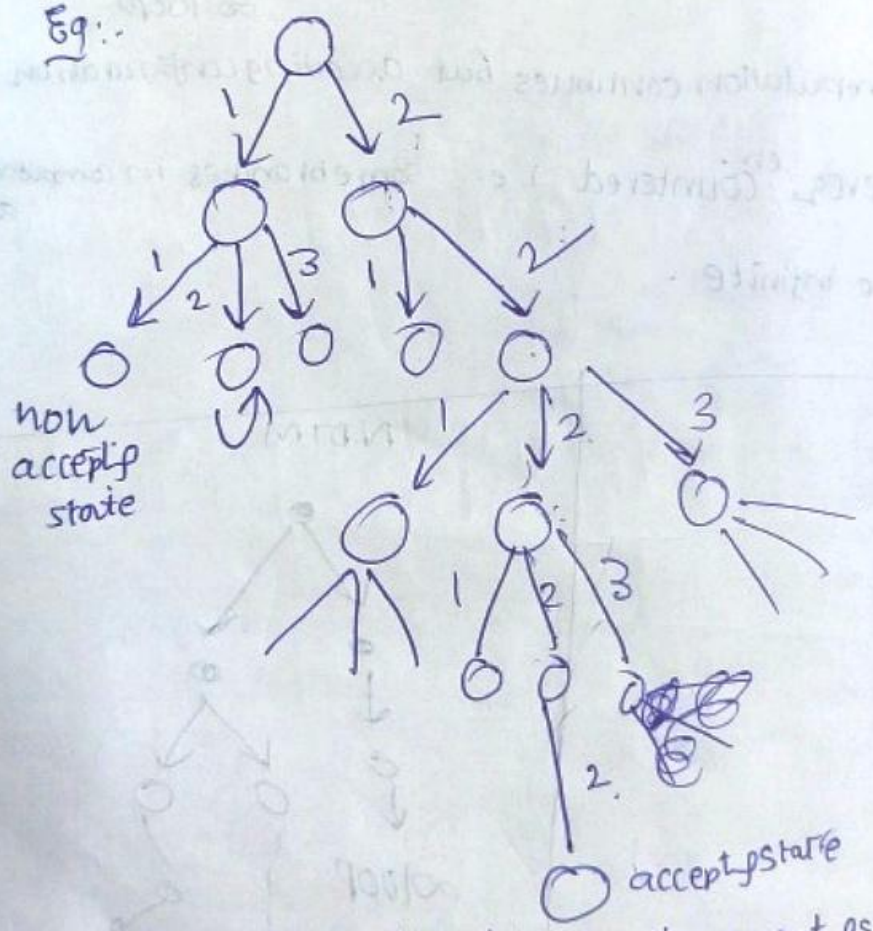
Theorem :- Every NDTM has an equivalent DTM

Proof :- given a NDTM (N) we have to construct equivalent DTM "D".

If N accepts on any branch, then D will accept

If N halts on any branch without accepting, then D will halt & rejected.

Eg :-



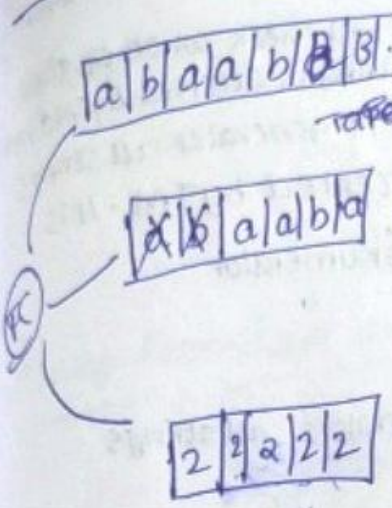
* path no. from Intial state to accept state :-

2 2 2 2 2

DFS :- we accept this. It may end in a loop

BFS :- yes we can apply.

DTM for NDTM :-



- Input Tape :- It will contain the input string. we can't modify
- Simulation Tape :- It is tape where we are doing simulation
- Address Tape :- It is used to compute the BFS. It tells which choices we have to make during computation.

Algorithm :-

① Intially tape 1 will contain input. Tape 2 & Tape 3 are empty.

step :- copy Tape1 to Tape2. Use tape2 as the actual tape. when choice occurs (nondeterministic branch point encounters) consult Tape3 because Tape3 contain the path. Each no. tells which choices we have to choose. Run the simulation all the way down the branch (till computation). Try next branch Increment the address Tape3. Repeat.

* If accept is encountered, halt & accept if all branches reject or die, then halt & reject.

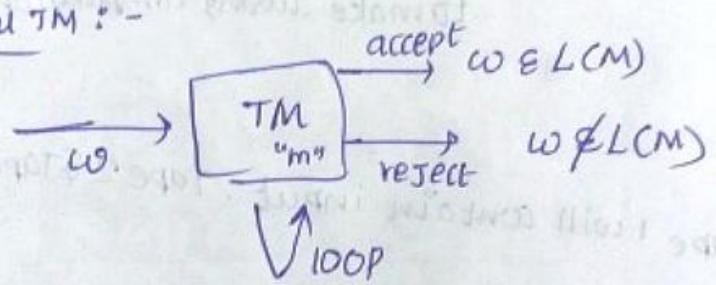
Enumerator :- → you can implement mathematical operations addition, one's complement, 2's compl.

TM can act as
① Transducer
② Acceptor → It will check whether given string is accepted by TM or not
③ generator → It generates all strings accepted by TM. It is also called Enumerator

(E)

Enumerator :- It is a generator. It generators all strings belonging to the language.

Normal TM :-



Enumerator = TM + printer

TM → working Tape / Input tape

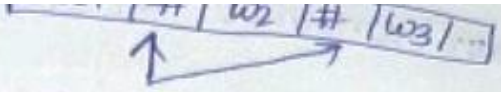
printer → It is nothing but output tape i.e. gives string generated by lang.

Enumerator has 2 tapes
- working Tape / Input tape
- Output Tape

L(M) → lang. accepted by TM

L(E) → lang. generated by Enumerator.

Enumerator

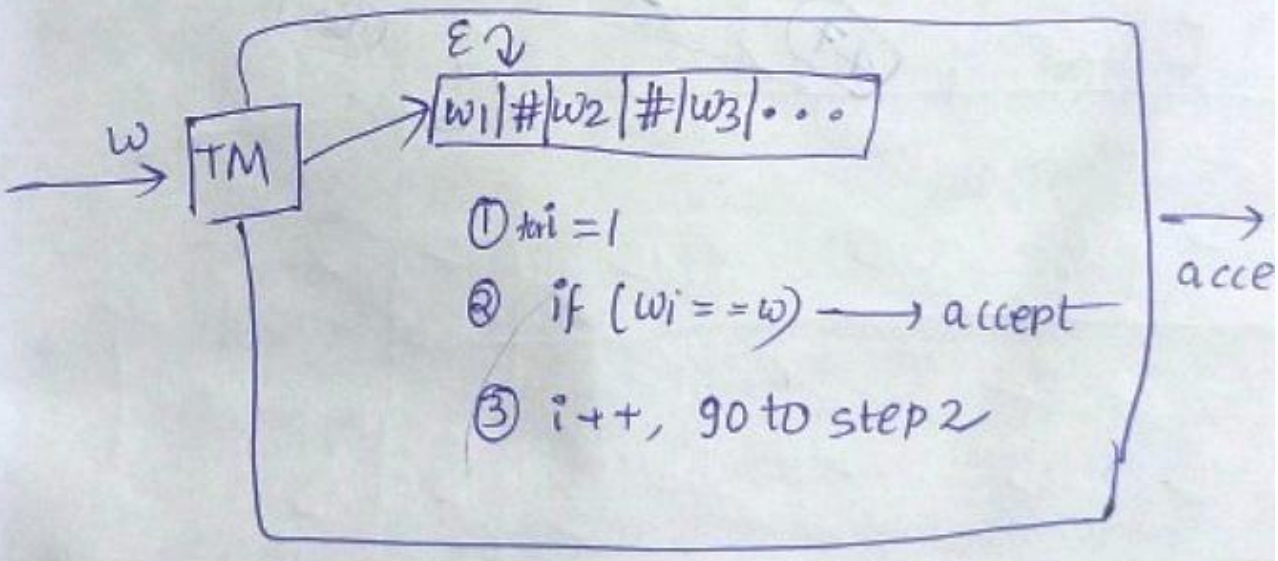


Two strings are seperated by hash (#) symbol.

Turing Recognizable lang. :- Lang. accepted by "TM"

Theorm :- A lang. is Turing recognizable if & only if some enumerator enumerates it.

Proof :- If part → If enumerator "E" enumerators the lang. 'A', then we can design a TM 'M' that recognizes/ accepts 'A'.



Turing machine 'M' accepts 'w' if Enumerator enumer 'w' in it's output tape

Only if part :- If TM recognizes a lang. 'A', then we can construct the following enumerator for 'A'

Assume that s_1 s_2 s_3 ... are the possible strings in Σ^*

① Ignore the input

② Repeat the following steps for $i = 1, 2, 3, \ldots$

③ Run 'M' for i steps on each input $s_1, s_2 \ldots s_i$

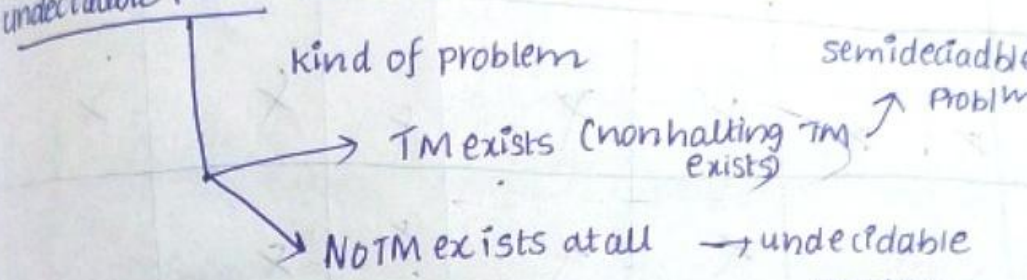
④ If any computation, print the corresponding s_i by enumerator.

~~Decidabiltity~~

Decidability

~~If~~ To solve a problem there must an algorithm, so there must be a halting Turing machine.

- undecidable problem :- There is no spefic alg^m to solve this kind of problem

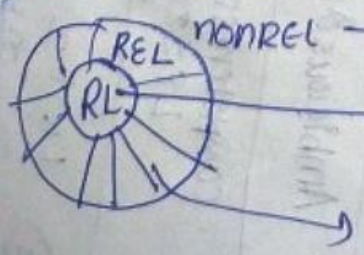


① we can convert any probl^m into a language

Recursive lang. :- A lang. is called recursive if there exists a TM that accepts string 'ω' if $\omega \in L(M)$ & rejects 'ω' if $\omega \notin L(M)$ [Halting/Total TM]

→ Any decidable probl^m can be converted into Recursivelang./decidable lang.

- ✱ Actually we call a probl^m undecidable if there exists no halting TM.



nonREL → undecidable

→ decidable probl^m

REL but not RL → semidecidable probl^m

Co. Turing recognizable lang.

* Recursive lang's are closed under u
& complement

* Recursive enumerable lang are cl
union, intersection but not c

Turing machine as probl

TM can solve any probl^m. Every pro
converted into a lang.

→ An instance of the probl^m is call
lang.

→ If the string is in the lang., TM

→ If the string is not in the lang,
reject.

Probl^m:—
Is this undirected graph is connecte

we can represented connected gr

A = { <G> | G is connected }

<G> = { (1,2, 3), (1,2), (1,3) }
vertices
(V)

V → decidable probl^m X → undec. probl^m	Reg. lang	DCFL	CFL	CSL	RL	REL
① membership problem $w \in \Sigma^*$	✓	✓	✓	✓	✓	X
② Infiniteness probl^m (L is finite / ∞)	✓	✓	✓	X	X	X
③ Emptiness probl^m If ($L = \phi$ or not)	✓	✓	✓	X	X	X
④ Equality probl^m (If $L_1 = L_2$)	✓	✓	X	X	X	X
⑤ Ambigious or not	✓	✓	X	X	X	X
⑥ completeness probl^m $L \in \Sigma^*$ or not	✓	✓	X	X	X	X
⑦ $L_1 \cap L_2 = \phi$ or not	✓	X	X	X	X	X
⑧ $L_1 \subseteq L_2$ or not (subset probl^m)	✓	X	X	X	X	X

* The complement of Turing recognizable lang. is called Co-Turing recognizable lang.

* Recursive lang's are closed under union, Intersection & complement

* Recursive enumerable lang are closed under union, intersection but not complement.

Turing machine as probl^m solver

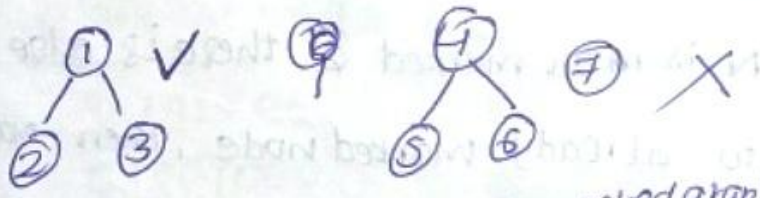
TM can solve any probl^m. Every probl^m can be converted into a lang.

→ An instance of the probl^m is called string of the lang.

→ If the string is in the lang., TM will accept

→ If the string is not in the lang, TM will reject.

Probl^m:— Is this undirected graph is connected or not



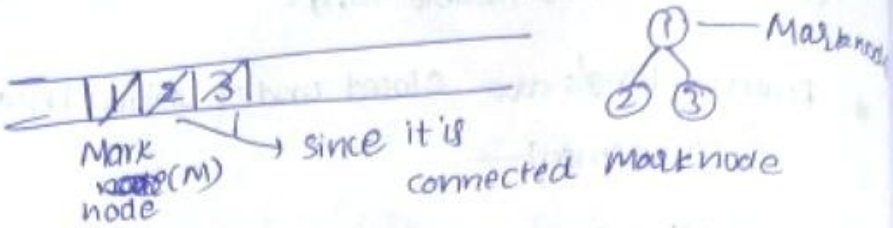
we can represented connected graph

$A = \{ \langle G \rangle \mid G \text{ is connected} \}$

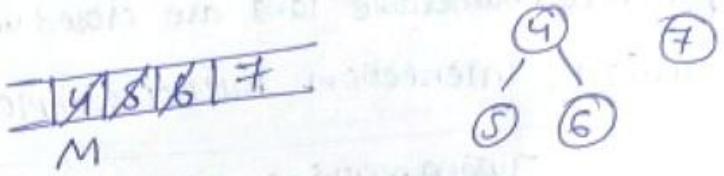
$\langle G \rangle = \{ \underbrace{(1,2,3)}_{\text{vertices } (V)}, \underbrace{((1,2),(1,3))}_{\text{edges}} \}$

For this G, $\Sigma = \{ (,), "1", 1, 2, 3, 4 \ldots, 9 \}$

G_1



G_2



* TM scans tape G_1 since all nodes are marked, it considers it as connected graph
* Sim/ly it scans G_2 since "7" is left. It considers it as un connected graph.

Algm for this :-

① scan a node & mark it

② Repeat for each node (N)
If N is un marked & there is edge from N to already marked node, then mark N
until no more nodes can be marked

③ for each node N if N is unmarked then reject & report graph is not connected.

P. Aswath

DFA :- Tuples = $\{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0\, q_1\, q_2 \cdots q_{n-1}\}$

Encode q_J as 0^{J+1} string

$q_0 \rightarrow 0^{0+1} \rightarrow 0$

$q_1 \rightarrow 0^{1+1} \rightarrow 00$

$q_2 \rightarrow 000$

$\Sigma = \{a_1, a_2, \cdots a_n\}$

Encode a_i as 0^i string

$a_1 \rightarrow 0$

$a_2 \rightarrow 00$

Eg :- $\Sigma = \{a, b\}$

Encoding of $a \rightarrow 0$ $b \rightarrow 00$

δ :- $\delta(q_J, a_J) \rightarrow (q_K)$

Encoded as $q_J 1 a_J 1 q_K$

acts as seperation b/w state & in

Eg :- $\delta(q_0, a_2) \rightarrow q_2$

$q_0 1 a_2 1 q_2$ [$\because q_J \rightarrow 0^{J+1}$

0100 1000 $a_i \rightarrow 0^i$]

F : set of final states in FA

suppose q_2 & q_4 are Final states of DFA

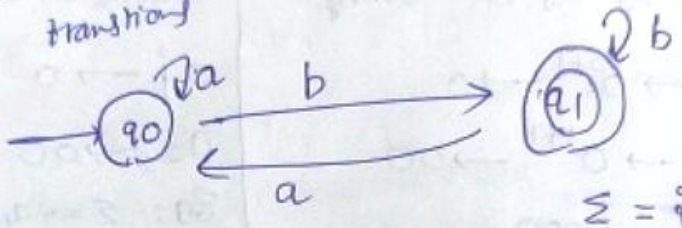
$q_2 1 q_4 \rightarrow$ 1 acts as seperation b/w Final sta

Encoded as 000 1 00000 [$\because q_J \rightarrow 0^J$

$$\underbrace{||0010001 \cdots\cdots ||}_{\text{encoding of all transitions}} \underbrace{\cdots\cdots ||}_{\text{encoding of final states}}$$

$$\underbrace{||T_1|T_2 \cdots ||}_{\text{encoding of transitions}} \underbrace{q_{f1}|q_{f2}| \cdots}_{\text{encoding of Final states}}$$

Eg :-



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

$a \to 0$, $b \to 00$

$\delta(q_0, a) \to q_0$ $[\because q_j \to 0^{j+1}$, $a^i \to 0^i]$

$\delta(q_0, b) \to q_1$

$\delta(q_1, a) \to q_0$

$\delta(q_1, b) \to q_1$

$T_1 : 01010$

$T_2 : 0100100$

$T_3 : 001010$

$T_4 : 00100100$

$$\langle A \rangle : ||\underbrace{01010}_{T_1}|\underbrace{0100100}_{T_2}|\underbrace{001010}_{T_3}|\underbrace{00100100}_{T_4}||\underbrace{0011}_{\text{Finalstate}}$$

$w = abb$ ($a \to 0$, $b \to 00$, $b \to 00$) This can represented 0100100

$\langle A, w \rangle$ Binary encoding of DFA 'A' with input string 'w'.

Acceptance problem of DFA :- Theorem 4.1

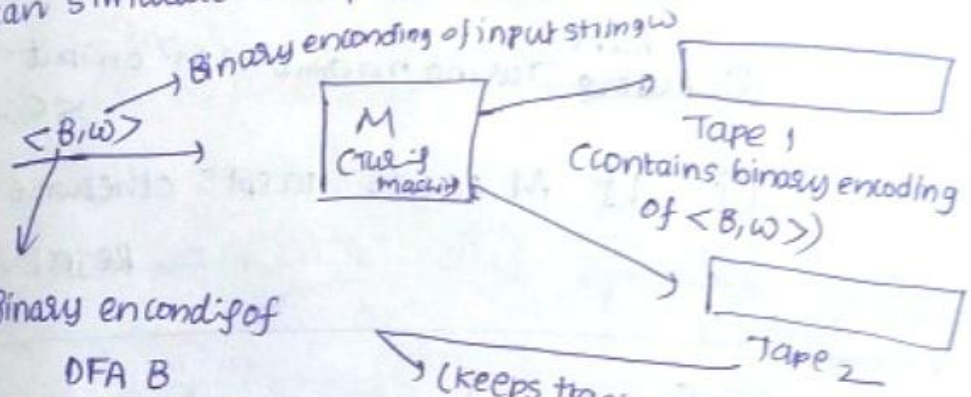
$A_{DFA} = \{ \langle B, \omega \rangle \mid B$ is a DFA that accepts string $\omega \}$

is a decidable probl^m.

each string in the lang. represents a DFA followed by input string

proof :-

We can simulate this probl^m with a 2 Tape TM



Algorith^m :- M = on input <B, ω> (T.M)

① simulate B on input string (ω)

② If simulation ends in a Final state (accepting state) of B then accept. If it ends in non Final state (non accepting state) then reject

is a decidable prob^{m}

proof:- ① convert NFA to DFA & use Therom 4.1

"on input $<B, w>$ where B is a NFA

① simulate NFA 'B' to equivalent DFA 'C' in

by using procedure used earlier units.

② Run Turing machine 'M' on input $<C, w>$

③ If M accepts, accepts otherwise Reject.

$A_{REX} = \{ <R, w> \mid$ R is a regular expression that generates string w $\}$ is a decidable prob^{m}.

Idea:- convert Regular expression into NFA & then use Therom 4.2

Proof:- Use DFS algorithm to mark the states of DFA

On input <A> where A is a DFA

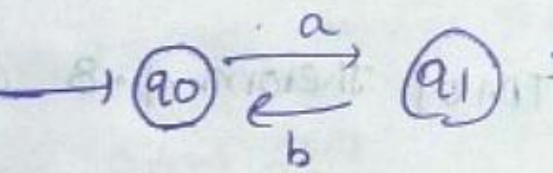
① Mark the start state of A.

② Repeat until now state get marked

Mark any state that a transition coming into it from any already marked state is marked,

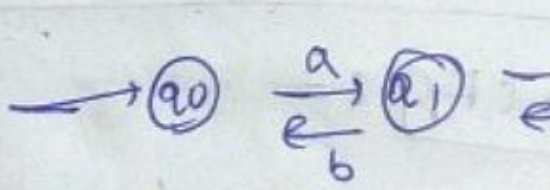
③ If no final state, accept, otherwise reject

Eg:-



~~q0~~	~~q1~~	~~q2~~

no final state marked → accept



~~q0~~	~~q1~~	~~q2~~

final state is marked, so reject

Therom 4.8

$EQDFA = \{ \langle A, B \rangle \mid A \,\&\, B$ are DFAs $\&\, L(A) = L(B) \}$

is a decidable problm

Proof:- construct the machine for DFA

$$L(C) = (L(A) \cap L(\overline{B})) \cup (L(\overline{A}) \cap L(B))$$

and check if $L(C) = \phi$

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

T: on input $\langle A, B \rangle$ when A & B are two DFA

① construct the DFA for language $L(C)$

② Run the TM of Therom 4.8 on input $\langle C \rangle$

③ If T accepts accept otherwise reject

Decidable prob^{m}s in CFG:-

① Does a CFG 'G' generate string 'w'?

membership property → CYK

② If $L(G)$ is empty?

$A_{CFG} = \{ \langle G, w \rangle \mid G$ is a CFG that generates $w \}$ is a decidable lang.

proof:- convert G into CNF form & use cyk algorithm

C: on input $\langle G, w \rangle$ when G is a CFG

① convert G into an equivalent grammar in CNF

② Run cyk alg^m on w of length (n)

③ If $S \in V_{i,n}$ accept otherwise reject

more efficient than below

or

② generate all strings that can be generated in $2n-1$ steps. If any of the strings is equal to given string 'w'. then CFG 'G' generates 'w'. If it is equal to w then accept otherwise reject.

$2n-1 \rightarrow a, b, c, d$

a ✓

$E_{CFG} = \{ \langle G \rangle \mid G$ is a CFG and $L(G) = \phi \}$ is a deciadble lang.

$S \to A, B$ $\quad A \to a \mid c$

$B \to b$

Proof :- Mark variables of G ~~syB~~ systematically if they generates terminal strings & check if S is unmarked.

R : On input $\langle G \rangle$ where G is a CFG

① Mark all terminal symbols in (G).

② Repeat until no variables get marked.

→ Mark every variable (A) such that G has a Rule $A \to V_1, V_2, \ldots V_k$ & $V_1, V_2 \ldots V_k$ are marked

③ If start symbol is not marked, accept otherwise reject

$EQ_{CFG} = \{ \langle G, H \rangle \mid G$ & H are CFG & $L(G) = L(H) \}$ is undeciclable problm (Proof can be discussed later)

Binary encoding of TM :-

$Q = \{q_1, q_2 \cdots q_n\}$

$q_n \rightarrow$ final state $\rightarrow$ If TM has multiple final state then convert multiple to a single final state

$q_1 \rightarrow$ initial state

encode q_J with 0^J string $\quad q_1 \rightarrow 0 \quad q_2 \rightarrow 00$

Assume the Tape symbol $(\Gamma) = X_1, X_2, X_3 \ldots X_j$

encode X_J with 0^J string

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

$X_1 \rightarrow 0$, $X_2 \rightarrow 00$, $X_3 \rightarrow 000$

we refer direction 'L' (left) as $D_1 \rightarrow 0$ & right as 'D2' $\rightarrow 00$

encode D_J with 0^j string $D_1 \rightarrow 0 \quad D_2 \rightarrow 00$

$\delta(q_J, X_J) \longrightarrow (q_J, X_L, D_M)$

$q_J 1 X_J 1 q_J 1 X_L 1 D_M$

$0^j 1 0^J 1 0^J 1 0^L 1 0^M$

Eg:- $\delta(q_2, X_2) \longrightarrow (q_3, X_4, R)$

$$\underset{q_2}{00}1\underset{X_2}{00}1\underset{q_3}{000}1\underset{X_4}{0000}1\underset{R}{00}$$

seperate one transition to another transcation with double '11' to indicate end of TM

use triple 111

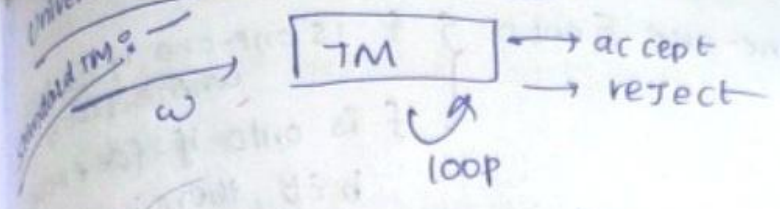
$$111 \underset{T_1}{\text{———}} 11 \underset{T_2}{\text{———}} 11 \text{———} \cdots 111$$

If M is a TM, <M> : Binary ends of TM

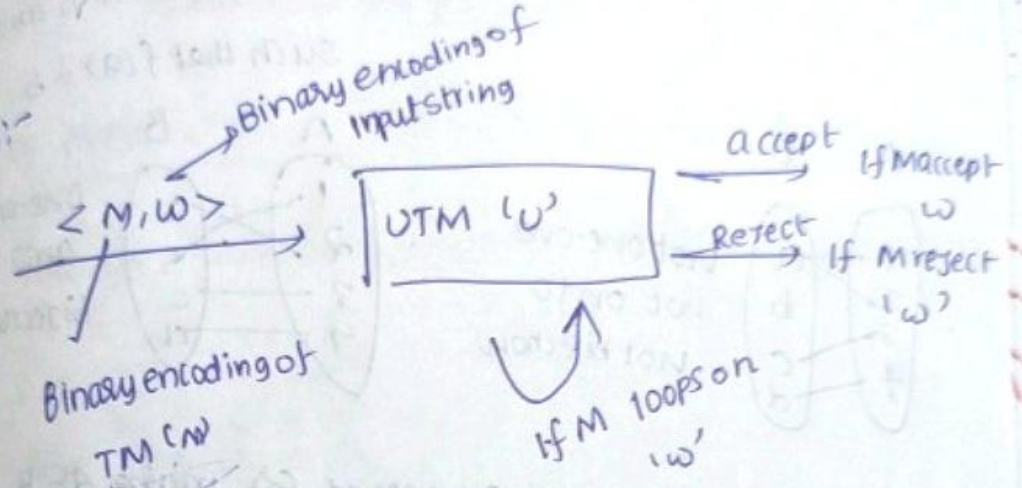
* no need to represent final state in TM since final state is the one which has highest number of zeros.

Universal Turing machine (UTM)

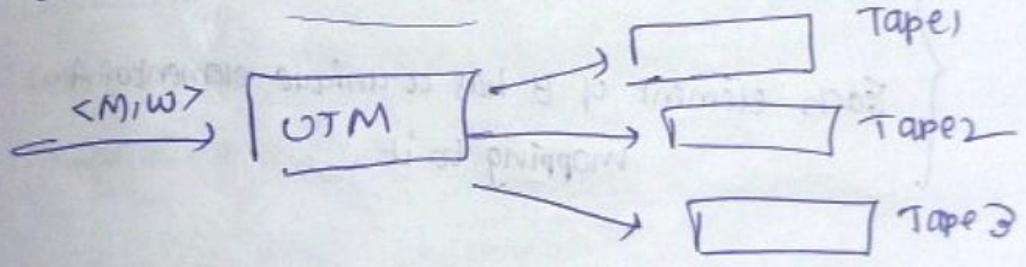
Standard TM :-



UTM :-



UTM acts as general purpose computer



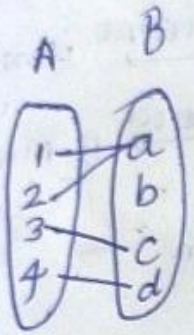
Tape1 → Binary encoding of TM 'M'

Tape 2 → Tape content of standard TM (M)

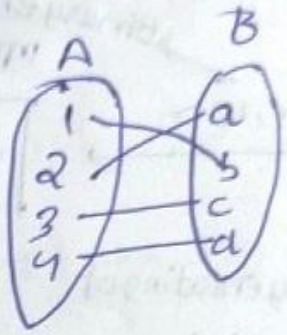
Tape 3 → State information of TM (M).

Bijection function :- A function is if it is one-one & onto

- f is one-one if $f(a) \neq f(b)$ whenever $a \neq b$
- f is onto if for every $b \in B$, there is an $a \in A$ such that $f(a) = b$



not one-one
not onto
not bijection



One-one ✓
onto ✓
bijection ✓

Bijection is called correspond for pairing A & B

- Every element of A maps to unique element of B
- Each element of B has a unique element of A mapping to it.

Sets
- Finite sets → Always countable
- Infinite sets → countable, uncountable

countable set :- A set 'S' is called countable if it's element can be one-one correspond with set of natural no's.

Uncountable sets :- A set 'S' is called if it is infinite & has no one-one correspondence with set of natural no's

Natural no $\rightarrow \{1, 2, 3, 4 \ldots\}$

set of even no's :- It is a countable set because it has one-one correspondence with set of natural no's

$$S = 0, 2, 4, 6, 8, \ldots$$
$$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$$
$$N = 1, 2, 3, 4, 5, \ldots$$

set of real no's :- It is a uncountable set since there is no one-one correspondence with sets of natural no's

set of Integers :- countable

$$Z = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

$$0, 1, -1, 2, -2, 3, -3, \ldots$$
$$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$$
$$1 \; 2 \; 3 \; 4 \; 5 \; 6 \; 7$$

$\Sigma = \{a, b\}$ set of alphabets

Σ^* = universal Lang. = $\{\epsilon, a, b, aa, bb, ab, ba, \ldots\}$ containing all possible strings on a,b

$N = \{1\ 2\ 3\ 4\ 5\ 6\ 7\}$

$\therefore$ It is a countable set

power set of Σ^* is represented as 2^{Σ^*}

2^{Σ^*} = set of all possible lang's in alphabets a,b

$\therefore$ It is uncountable (proof later)

Th :- set of all Turing machine is called a countable set

Proof :- Every TM can be encoded using Binary strings but every binary string doesnot represents a TM.

$\Sigma = \{0, 1\}$ Σ^* = set of all possible binary strings on ab

Σ^* is countable set

M = set of all TM's

We know that $M \subseteq \Sigma^*$

Since Σ^* is countable set, M is a countable set

W.K.T subset of countable set is a countable set

$\therefore$ M is a countable set Here M represents set of all TM's.

Therom :- 2

Proof :- Us

Assume th

$\Sigma = \{a,$

$L_1 =$

$\Sigma^* = \epsilon$

1) 1

2) 0

3) 1

4) 0

5) 0

6) 1

Diagonal

$L =$

$L^c =$

We cann

$\therefore$ Therefore

$\therefore$ So, 2^{Σ^*}

Therom :- 2^{Σ^*} is a uncountable set

Proof :- Using proof of contradication

Assume that 2^{Σ^*} is a countable set

(Diagonization method)

$\Sigma = \{a, b\}$ $\Sigma^* = \{\epsilon, a, b, aa, ab, ba, aaa, \ldots\}$

$\downarrow$

$L_1 = \{\epsilon, a\}$ $L_2 = \{a, b\}$ $L_3 = \{\epsilon, a, aa\}$

$\Sigma^* = \epsilon, a, b, aa, ab, ba, bb, bb, aaa, \ldots$

1)	1	1	0	0	0	0	0	0	0	—	—
2)	0	1	1	0	0	0	0	0	0	—	—
3)	1	1	0	1	0	0	0	0	0	—	—
4)	0	0	1	0	0	0	0	0	0	—	—
5)	0	0	0	1	0	0	0	0	0	—	—
6)	1	0	0	0	0	0	0	0	0	—	—

Diagonal elements also represents a lang.

$L = 110000 \ldots$

$L^c = 001111 \ldots$

We cannot map L^c to set of natural no's

$\therefore$ Therefore, our assumption 2^{Σ^*} is a countable set is false

$\therefore$ So, 2^{Σ^*} is a uncountable set

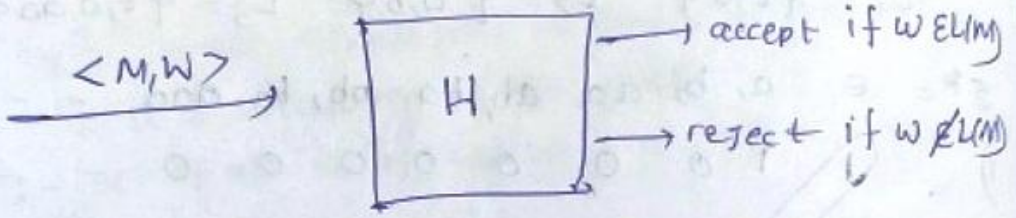
Acceptance probl^m in TM :-

$A_{TM} = \{ < M, w > \mid M$ is a TM & M accepts $w \}$ is undecidable

→ acceptance pb^m of TM

Proof by contradaction :- Assume that A_{TM} is decidable

Suppose H (universal TM) decides A_{TM}



we can construct a new TM D

D = "on input < M > where i s a TM

Run 'H' on input (M, < M >) binary encoded

if H accept, reject

If H reject, accept

$$D(<M>) = \begin{cases} \text{accept if M doesn't accept } <M> \\ \text{Reject if M accept } <M> \end{cases}$$

$$D(<D>) = \begin{cases} \text{accept if D doesn't accept } <D> \\ \text{Reject if D accept } <D> \end{cases}$$

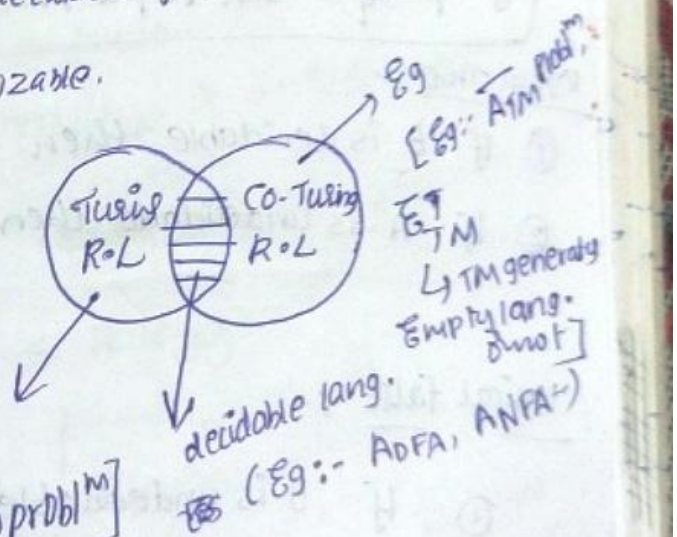
H is telling that D rejects, but D is telling it accepts
H is telling that D accepts, but D is telling itself accept
(Neither D nor H exist)

This is contradiction to our assumption that A_{TM} is decidable

∴ so A_{TM} is undecidable problm)

Co-Turing Recognizable lang.:- A complement of Turing Recognizable lang. is called co-turing recognizable lang.

* A lang. is called decidable if it is Turing Recognizable & co-Turing recognizable.



Reducibility

A reduction is a way of converting one probl^m (pb^m) into another prDbl^m. so that the sol^n of the 2nd pb^m is used to solve the 1st pbm.

pb^m A is reduced to pb^m B can be represented mathematically as $A \leq B \rightarrow$ Easy pb^m

Hard pbm

Finding area of rectangle $\leq$ Find length & breadth of rectangle

$\boxed{\text{If } p \rightarrow q \text{ then } \sim p \rightarrow \sim q}$ ← We know that

logical True :-

① If B is decidable then A is decidable

② If A is undecidable then B is undecidable

True sol^n

logical false :-

① If B is undecidable then A is undecidable.

② If A is decidable then B is decidable

Notes:- A TM is called Decider, if it halts in final state on input string ω & halts on non final state on input string ω. It will never go infinite loop. Decider is also halting TM.

$A_{TM} = \{ \langle M, \omega \rangle \mid M$ is turing machine & M accepts $\omega \}$ is undecidable

~~Halting pb^m~~

Halting pb^m :-

$HALT_{TM} = \{ \langle M, \omega \rangle \mid M$ is a TM & M halts on $\omega \}$

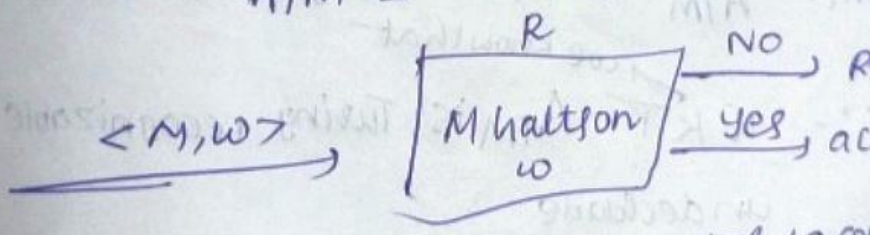
is undecidable.

Proof:- (proof by contradaction)

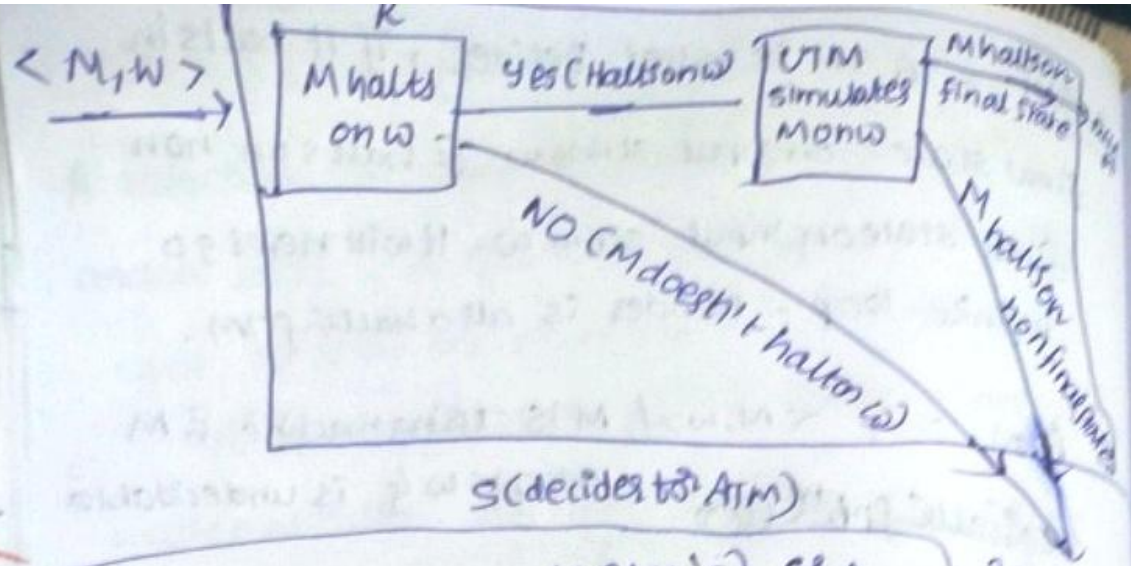
Assume that $HALT_{TM}$ is decidable pb^m

then we can ~~convert~~ construct a decider R

$$A_{TM} \leq Halt_{TM}$$



Now we can use the decider "R" to construct decider "S" for A_{TM}



Thus we can design decider 'S' for A_{TM} pb^m, so

A_{TM} pb^m is decidable.

But we know that A_{TM} is undecidable pb^m

So our assumption that $HALT_{TM}$ is decidable is false

$\therefore$ $HALT_{TM}$ pb^m is undecidable.

Theorem:- $\overline{A_{TM}}$ is not Turing recognizable

Proof:- W.K.T we know that A_{TM} is Turing recognizable & undecidable

(proof by contradiction) Assume that $\overline{A_{TM}}$ is Turing - recognizable. Already W.K.T A_{TM} is Turing Recognizable

If A_{TM} & $\overline{A_{TM}}$ is Turing recognizable, then A_{TM} is decidable but W.K.T A_{TM} is undecidable pb^m

Hence our assumption is false.

m, so

pb^m

ecidable

e

zable Ep

is Turing-

A_{TM} is

then A_{TM}

lable pb^m

Pg 2 Q1m

$\therefore \overline{A_{TM}}$ is not Turing recognizable.

Therom :- Prove that EQ_{TM} & $\overline{EQ}_{TM}$ are not Turing Recognizable [∵ $EQ_{TM} = \langle M_1, M_2 \rangle \mid M_1$ is equal to M_2]

↳ [∵ not recursive enumarable lang.]

[∵ $\overline{EQ}_{TM} = \langle M_1, M_2 \rangle \mid M_1$ is not equal to M_2]

Proof :- $A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$ is undecidable but Turing recognizable.

$\overline{A_{TM}}$ is not Turing recognizable

① $\overline{A_{TM}} \leq_M EQ_{TM} \Leftrightarrow A_{TM} \leq_M \overline{EQ_{TM}}$

② $\overline{A_{TM}} \leq_M \overline{EQ_{TM}} \Rightarrow A_{TM} \leq_M EQ_{TM}$

[∵ $A \leq_M B \Leftrightarrow \overline{A} \leq_M \overline{B}$ if A is undecidable then $w \notin A$ $f(w) \notin B$ then B is also undecidable]

TO prove EQ_{TM} is not recognizable :-

① $\overline{A_{TM}} \leq_M EQ_{TM} \Rightarrow A_{TM} \leq_M \overline{EQ_{TM}}$

[∵ $(M, w) \in A_{TM} \Rightarrow f(M, w) = (M_1, M_2)$]

$\langle M, w \rangle$

The following T.M F computes mapping reduction from A_{TM} to $\overline{EQ_{TM}}$

F: on Input $\langle M, w \rangle$ where M is a Turing machine & w is a string.

M_2 : on any input

Run M on ω, if it accepts ω then accept everything.

$$L(M_2) = \begin{cases} \Sigma^* & \text{if M accept } \omega \\ \emptyset & \text{if M reject } \omega \end{cases}$$

output (M_1, M_2)

$\overline{A_{TM}} \leq_m EQ_{TM}$ if $\overline{A_{TM}}$ is not a Turing recognizable lang.

EQ_{TM} is not a Turing recognizable lang.

to prove $\overline{EQ_{TM}}$ is not Turing recognizable

$\overline{A_{TM}} \leq_m \overline{EQ_{TM}} \Rightarrow A_{TM} \leq_m EQ_{TM}$

M_1 : on any input accept $L(M_1) = \Sigma^*$

M_2 : on any input

Run M on ω, if it accepts ω the

$$L(M_2) = \begin{cases} \Sigma^* & \text{if M ac} \\ \emptyset & \text{if M} \end{cases}$$

Output (M_1, M_2)

$\overline{A_{TM}} \leq \overline{EQ_{TM}} \Rightarrow$ if $\overline{A_{TM}}$ is

$\overline{EQ_{TM}}$ is not a Turing

Prove $Halt_{TM}$ is undecidable using mapping reduction

Proof:-

$A_{TM} \underset{M}{\leq} Halt_{TM}$ [$\because A \underset{M}{\leq} B$ if A is undecidable then B is '']

Define a computation function 'f'

input to the function $f = \langle M, w \rangle \in A_{TM}$

output of the func^n $f \langle M, w \rangle = \langle M', w \rangle \in Halt_{TM}$

The following TM 'M' computes the func^n f

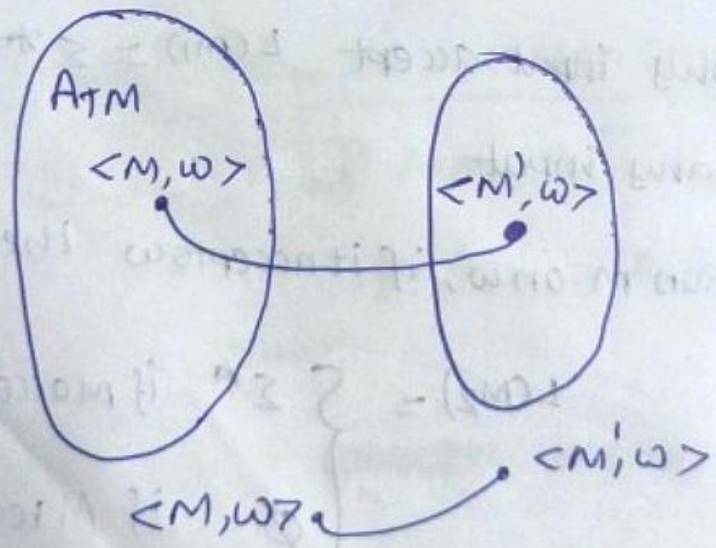
M' : on input w

Run M on w

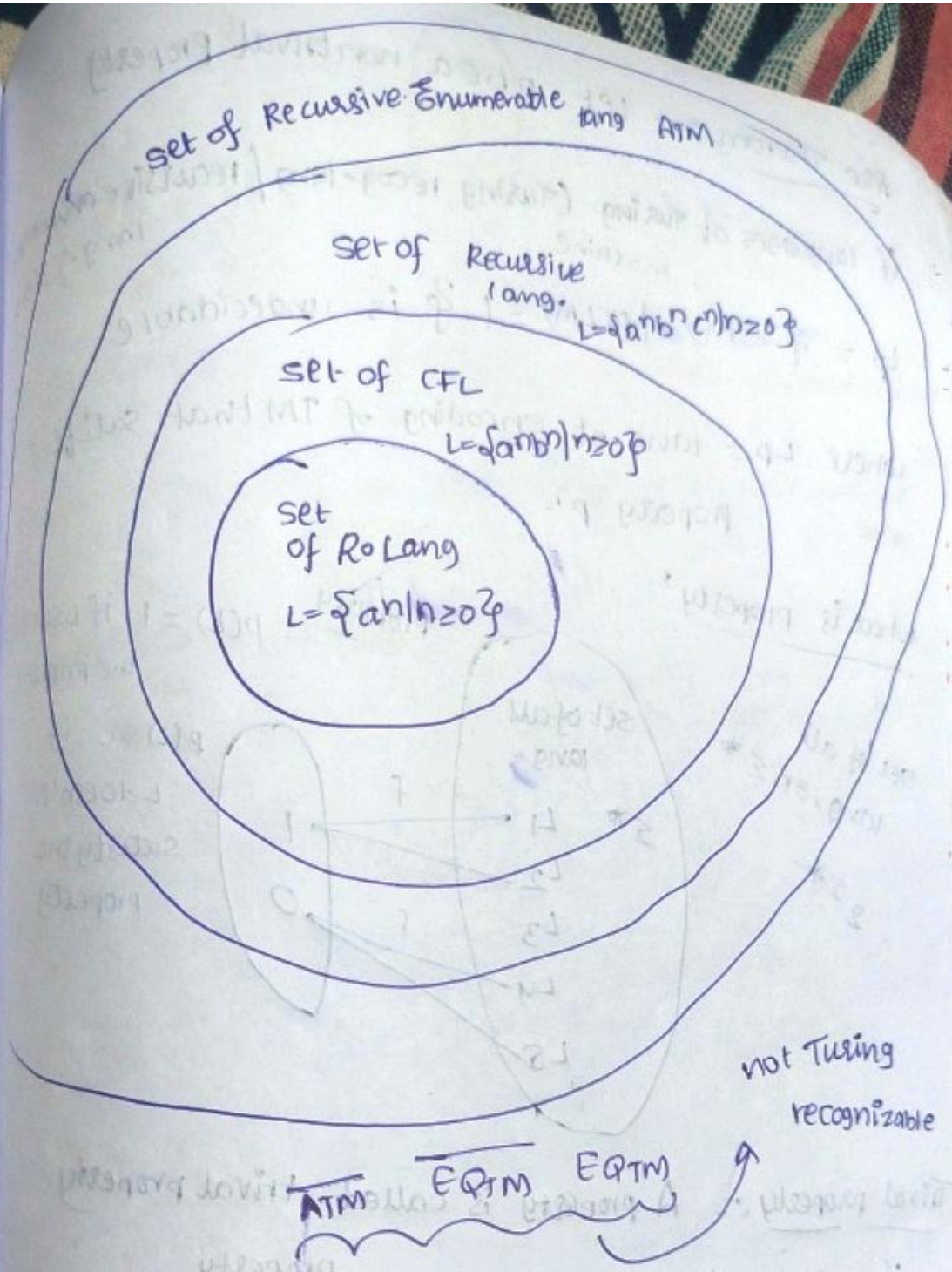
if M accepts, accept

M reject, enter loop

output $\langle M', w \rangle$



$A_{TM} \leq HALT_{TM}$ Already we know that A_{TM} is undecidable $\therefore$ Therefore $Halt_{TM}$ is undecidable


set of Recursive Enumerable lang ATM
set of Recursive lang.
L={a^n b^n c^n | n≥0}
set of CFL
L={a^n b^n | n≥0}
set of Ro Lang
L={a^n | n≥0}
not Turing recognizable
ATM
EQTM
EQTM

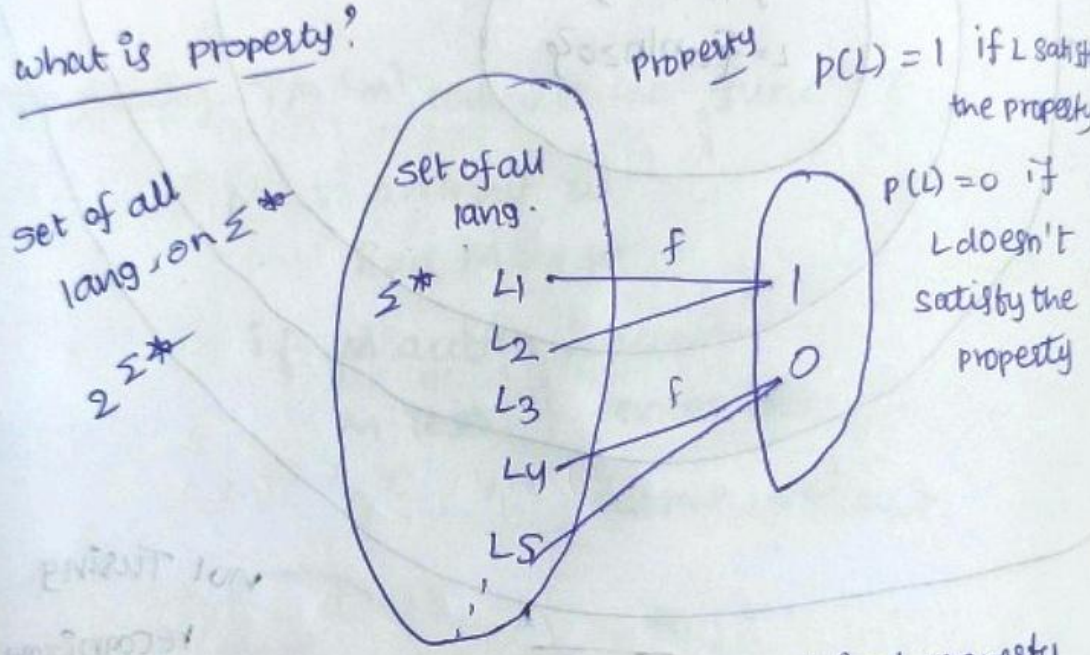
Rice Therom :- Let p be a nontrival property

if languages of Turing machine (Turing recog. lang / recursive enumerable lang.) then

$L_p = \{ \langle M \rangle \mid p(L_M) = 1 \}$ is undecidable

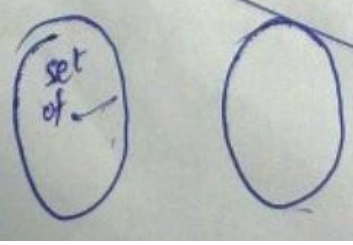
where L_p = lang of encoding of TM that satisfies property 'p'.

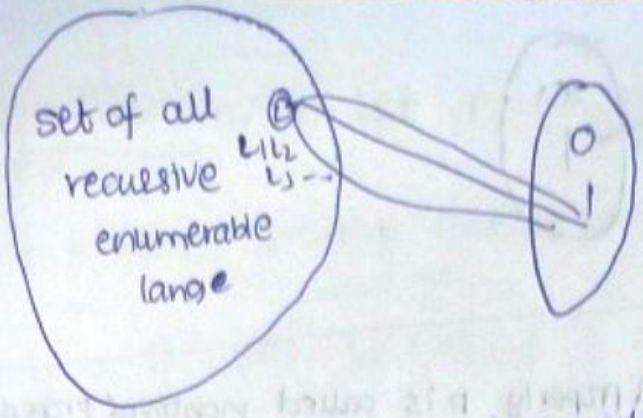
what is property?



Trival property :- A property is called trival property if all lang's satisfies the ~~lang~~ property or none of the lang. satisfies the property. [all are mapped to either 0 or 1]

~~Eg :- If the lang's L Recursive enumerable lang is a property is there is a TM for lang's~~





<u>Non-Trivial property</u> :- A property p is called non trivial property if it is satisfied by some of the recursive enumerable lang's & not satisfied by some

(or) If there exists TM's M_1 & M_2 such that property $P(L(M_1)) = 1$ & $P(L(M_2)) = 0$

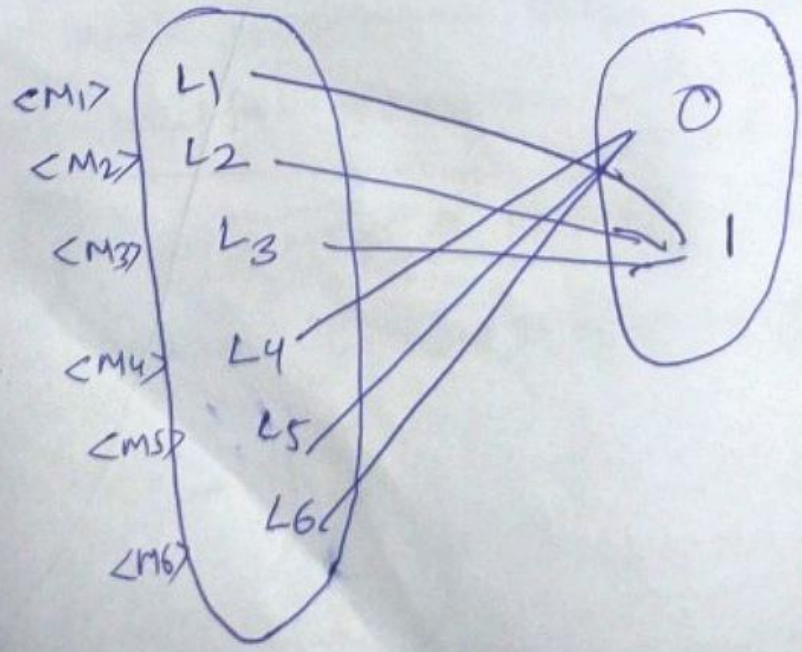
P : L is a finite lang → Trivial

P : L has 0010 substring → Trivial

P : $L = \Sigma^*$ → Trivial

P : $L = \phi$ → Trivial

P : (L is a regular lang.) L_1, L_2, L_3 → Regular

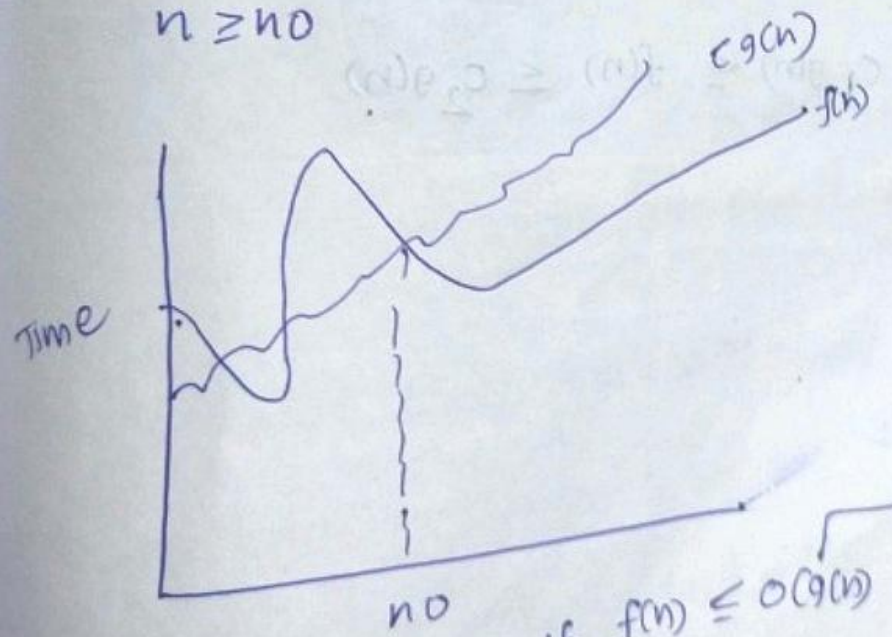


Algorithm :- step by step procedure to solve a problem in finite no. of steps.

Analysis of alg^m

- Space complexity :- amount of space taken by alg^m to run as function of input length
- Time complexity :- amt of time taken by alg^m to run as a function of input length
  - Big O notation (worst case) Upper bound
  - Omega (Ω) notation Best case lower bound

Big O notation :- $f(n)$ is $O(g(n))$ if there exists a possible constant c such that $f(n) \le c \cdot g(n)$ for $n \ge n_0$

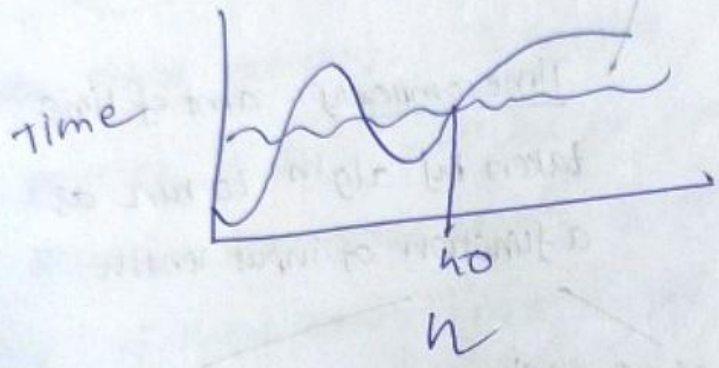


If $f(n) \le O(g(n))$ higher

then $g(n) = \Omega(f(n))$

Omega notation :-

$f(n)$ is $\Omega(g(n))$ if there exists a positive constant c & n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$



* if $f(n) = O(g(n))$ & $f(n) = \Omega(g(n))$
  then $f(n) = \Theta(g(n))$ ← Theta notation

* If $f(n) \leq O(g(n))$ then $g(n) = \Omega(f(n))$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

P.Kowshik

A = { } A is CFL, so A is decidable

Below is the TM that decides A

M_1 : "on input w"

① Scan across the tape & reject if 0 appears right of 1 $\longrightarrow O(n)$

② Repeat if both 0's & 1's remains in tape

scan the tape & cross one 0 & one 1
$\hookrightarrow O(n^2)$

③ If all 0's & 1's are crossed, accept otherwise reject $\hookrightarrow O(n)$

Eg :- $O(n/2) \times O(n) = O(n^2)$

{ n/2 times, each time Time com = O(n) }

M_1 takes total $O(n) + O(n^2) + O(n) = O(n^2)$

M_1 needs $O(n^2)$ steps to decide if w is in L or not. M_1 is deterministic TM

Let M be a deterministic TM, that halts on all inputs. The running time of M is a function $f: N \to N$ where $f(n)$ is the max. no of steps that M uses on every input of length n. if $f(n)$ is running time of TM 'M', we can say M runs in $f(n)$ & M is a $f(n)$ time TM [Eg:- $O(n^2)$ time TM]

<u>Time complexity class</u> :- we define the Time complexity class TIME $f(n)$ to be all lang's that can decided by $O(f(n))$ time Turing machine

Eg:- $\underbrace{B, C, D, E}_{O(n^2)}$ $O(n^2)$

(A, B, C, D, E)

Can I decide $A = \{0^k 1^k \mid k \geq 0\}$ in less time?

yes $O(n \log n)$

M_2: on Input w

1) If 0 appears on the right of 1, reject $O(n)$

2) Repeat if both 0's & 1's remain in tape) $O(n \log n)$

a) If total no. of 0's & 1's is odd, reject

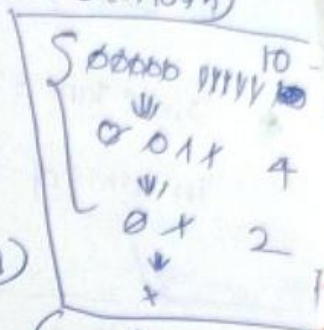
b) Scan the tape, cross off every other 0. Then cross every other 1

if all 0's & 1's are crossed, then accept.
otherwise reject $\rightarrow O(n)$

$$O(n) + O(n \log n) + O(n) = O(n \log n)$$

M_1 & $M_2 \rightarrow$ single tape TM

M_3 :- multi tape TM $\rightarrow O(n)$



$O(\log_2 n)$ each time
scan full tape $O(n)$
$\therefore O(n \log n) \rightarrow$ optimised TC

Algm using multitape TM :-

M_3 : On input string w:

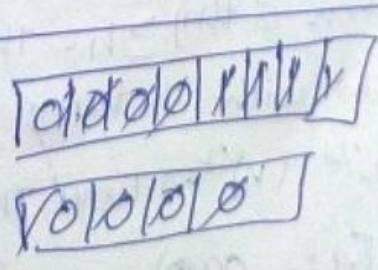
① If 0 appears on the right of 1, reject

② Scan across 0s on tape1 until first 1. At the same time, copy 0s to tape2

③ scan tape 1 & tape2 together. Each time, match a 0 with a 1.

④ If all 0s & 1s match, accept. otherwise reject.

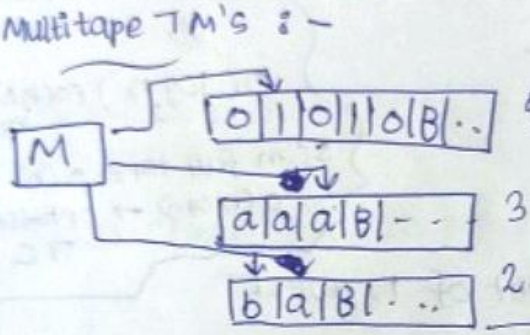
M_3 : ~~decide~~ decide $A = \{0^n 1^n \mid n \geq 0\}$ in $O(n)$ time



* Single Tape & Multi tape TM have the same power in terms of computability whether a probl^m can be solved / not solved

* Single Tape <-> Multi tape TM does not have same power in terms of complexity (how fast a probl^m can be solved)

Multitape TM's :-



$\delta(q_1, 1ab) \rightarrow (q_2, xyz, LLR)$

K Tape TM :-

$\delta(q_1, a_1 a_2 \ldots a_k) \rightarrow (q_j, b_1 \ldots b_k, R \ldots L)$

$\delta : Q \times T^k \rightarrow Q \times T^k \times (L,R)^k$

Single Tape TM for M (i.e., S) :-

Intial

① # 0 1 0 1 0 # a a a # b a #

Blank :- not active place

② #

active portion

Initially

① # 0 1* 0 1 1 0 # a a a* # b* a #

② # 0* x 0 1 1 0 # a a* y # z a* #

Theroms:-

Let $t(n)$ be a function & $t(n) \geq n$. Then for every $t(n)$ time TM that works with K tapes, there is an equivalent $O(K^2 t(n)^2)$ time TM that works with one tape (Eg:- $\underbrace{O(n)}_{5 \text{ Tape}} \rightarrow \underbrace{5^2 O(n^2)}_{5} = 25 O(n^2)$

Ref :- (If Multitape TM runs in $t(n)$ the single Tape TM runs in $k \times t(n) + k+1$
$= O(k\, t(n))$)

To simulate single step in multitape TM, we have to do two scans in single tape TM [∵ 1 scan — for getting current R/W head position
1 scan — to update & relocate]

$O(k\,(t(n))) \longrightarrow$ Time required for single scan

$O(2 \times k(t(n))) \longrightarrow$ " " " two scans in single tape TM.

single Tape TM

To simulate single step in Multitape TM $= O(k(t(n)))$
$= O(t(n))$ [∵ k → constant]

" " $t(n)$ steps in Multitape TM

$= t(n) \times O(t(n))$

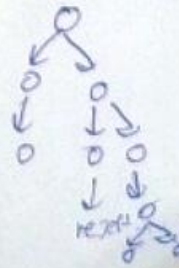
$= O(t(n)^2)$

$O(n) + O(t(n)^2)$
$= O(t(n)^2)$ } → see slides

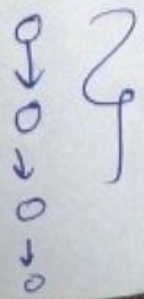
$O(n^c) \longrightarrow$ Polynomial Time Bounds where $c > 0$

single tape TM & Multitape TM → Polynomial Time Bound

complexity in Nondeterministic TM = Max branch



DTM :-



} linear

DTM VS NTM decider :- [not valid for NTM that goes in ∞ loop]

single tape NTM decider have $t(n) \rightarrow$ single tape DTM decider have $\left[2^{O(t(n))}\right]$

* Every node

P.Kowshik