TOS [Sudeep Sir]

Sipser: intro. to theory of computation

Computability theory: problems that can be solved by a computer.

Computer cannot predict with 100% accuracy the following problem's solution:

i) proving correctness of a mathematical statement

ii) result of futuristic events (result of today's game)

1932 - Alan Turing - Turing m/c - mathematic model.

problem solved by Turing m/c $\Leftrightarrow$ solvable by any m/c

→ Turing m/c is assumed to have ∞ memory

: Models of m/c that can solve a problem

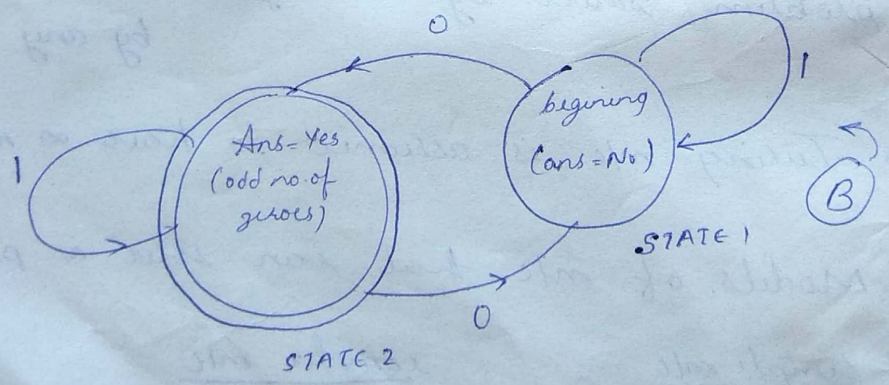
simple m/c	complex m/c
calculator (solve some problem)	Turing m/c.

Automata : Types of mathematical models of m/c

→ Design a m/c that gives a binary string it gives 'yes' if it has odd no. of zeros and otherwise 'no'.

A: Here lets consider lang A as which consists of all binary strings with odd number of zeros.

beginning ans = No
At 0's ans changes
At 1's ans remains same



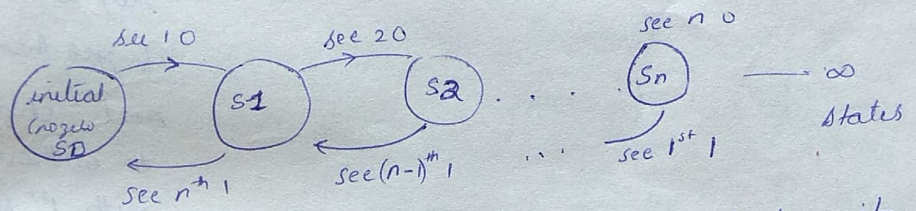
— finite automata

Lets m/c B decides language

. Given a string has 0, 1 or #. Check if its of the form w#w (∵ w - any string)

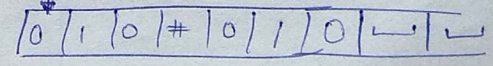
A: let A' = lang has string of the form w#w?
w is string with 0, 1 or #

If the m/c doesn't need finite no. of states;



if m/c returns to initial state, it has same no. of 0's and 1's in the form (00...111...)
0's 1's

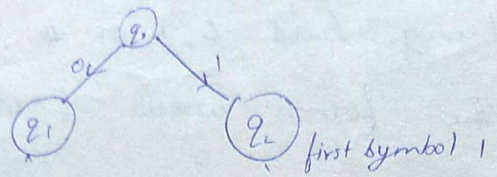
A Turing m/c has



A ∞ tape with initially the input followed by 'blank' symbols.

(i) # alone i.e. #␣␣ ⇒ ans = yes
(ii) NO # ⇒ ans = no (traverse one)

(iii) 0...#...



Sipser

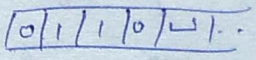
Once you see has
compare next cha
with 1st one
if same cross both els
say ans = no
if yes then compa
next char after
without cross
with 1st symbol
from 1st without
cross.

Turing m/c has:

① An input tape : An input string followed by blank symbols (␣)

② tape head : initially points to beginning to tape.

③ At a time tape head moves left or right by 1



④ Input alphabet Σ (here 0,1)

⑤ Tape alphabet Γ here $\{0,1,␣\}$

$\Sigma \subseteq \Gamma \quad \& \quad ␣ \in \Gamma$

⑥ Finite set of state (works for any i/p)

⑦ q_0 - initial state
q_{accept} - accept "
q_{reject} - reject state

$\in Q \quad q_{accept} \neq q_{reject}$

⑧ transition fn (or "moves"/"steps")

$\delta(q, a) \longrightarrow (r, b, D)$

q : current state
a : current symbol
r : new state
b : new symbol
D : direction of movement

Q] $Q = q_0, q_1, q_2, q_{acc}, q_{rej}$

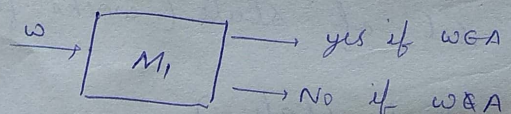
$\Gamma = \{0, 1, x, \sqcup\}$ — symbol that can be seen in tape

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times D$

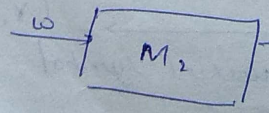
we need to define transition for each state & symbol. If not its assume to stop these & go to q_{rej}.

(Once rejected tape symbol & direction & are not important)

M_1 decides lang A



Now, in some cases this type of m/c may not be designed even if its computable. For those cases:



yes if $w \in A$

It doesn't say 'yes' if $w \notin A$

(i, it doesn't say no)

i.e., If $w \in A$, it may enter ∞ loop i.e. do not stop.

$\Rightarrow$ lang A is Turing decidable if $\exists$ M which decides A (recursive)

& lang A is Turing recognisable if $\exists$ M' which recognizes lang A (recursively enumerable)

---

set of Turing decidable lang $\subset$ set of Turing recognisable lang

---

Q. ① $w \# w$
② $0^N 1^N$
③ 0^{2N}

$w \in \{0,1\}$

design m/c that checks these lang. decides

We say a turing machine accepts a lang. if it reaches q_{accept} at end of tape

turing m/c stop @ q_{reject}/q_{accept} i.e. no moves from q_{rej}/q_{accept}

if a transition is not defined, it usually moves to q_{reject}

② $0^n1^n = A$.

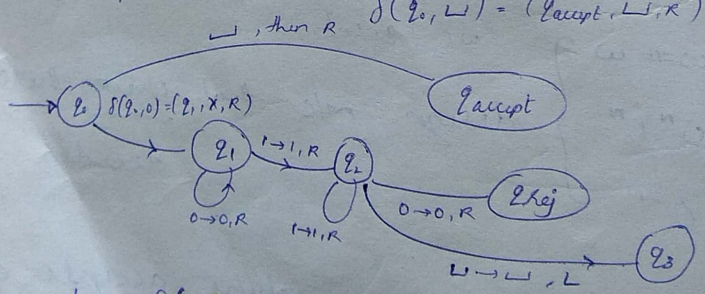
ⓐ scan from left → right

ⓑ if ∄ 0 after 1st 1, reject

ⓒ Come back to left and cross off 0 and go right & cross of corresponding 1.

If they both end up at same time, accept.

ⓐ



Now, if $\delta(q_0, 0) = (q_0, 0, R)$ then if i/p is 0 0 ␣ it accepts.

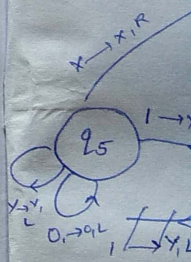
change

Undefin

Now, or

we ha

follow



change $0 \rightarrow x$ to know tape head

$\delta(q_0, \sqcup) = (q_{accept}, 0, R)$

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_1, 1) = (q_2, 1, R)$

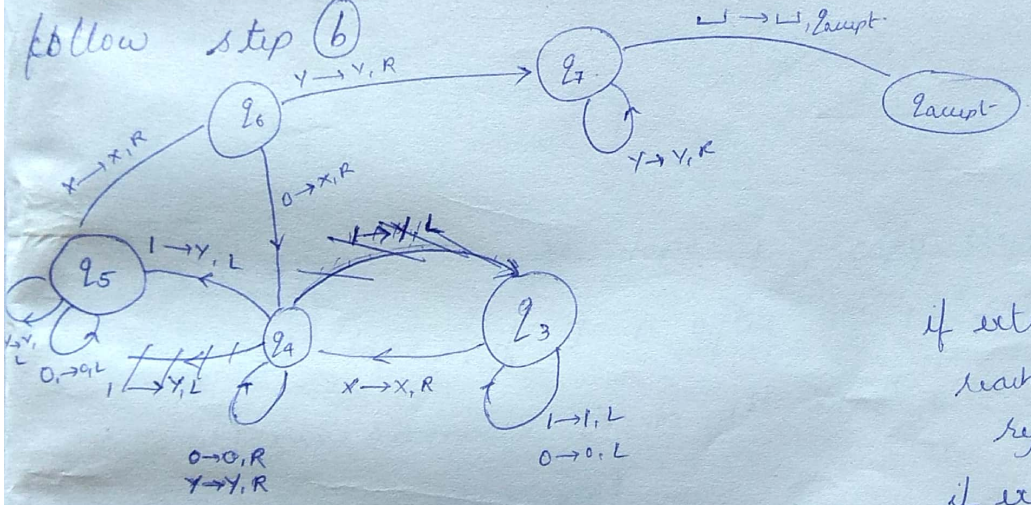
$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_2, 1) = (q_2, 1, R)$

$\delta(q_2, 0) = (q_{rej}, 0, R)$

Undefined transition $\rightarrow q_{reject}$

Now, once it reaches $\sqcup$ in q_2 we have to move back and follow step (6)



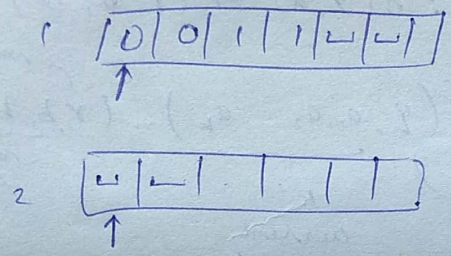
if extra 1, reach q_7 & reject

if extra 0, reject at q_4

$O(n^2)$

⊙ We considered single tape deterministic turing m/c.

Now what if we had 2 tapes with input ϕ, blank.



(1) scan through tape 1, copy 0s to 2nd tape. if ∄ 0 after 1st 1, reject

(2) now compare 1's in 1st tape (forward) with 0s in 2 (backward)

If both finish together, accept.

else reject.

⇒ O(n)

this is called multi tape turing m/c (k-tape for k tapes)

Rule: ⊙ Tape 1 has i/p followed by ␣

⊙ Tape 2-k has ␣ initially

⊙ set of states Q, $(q_{accept}, q_{rej}, q_0) \in \Gamma$ — states ⊂ sum

Now, transition fn/ moves,

single tape : $\delta(q, a) \to (r, b, D)$

k-tape : $\delta(q, a_1, a_2 \ldots a_k) = (r, b_1 b_2 \ldots b_k, D_1 D_2 \ldots D_k)$

new state; new symbol; direction

cur. state; k current symbols

$a_i, b_i \in \Gamma \quad \forall i = 1 \to k$

$\Rightarrow \delta : Q \times \underbrace{\Gamma \times \ldots \Gamma}_{\Gamma^k} \longrightarrow Q \times \underbrace{\Gamma \times \Gamma \times \ldots \Gamma}_{\Gamma^k} \times D^k$

Q it can also be seen as

$\delta(q, a, a_2) = (r, (b_1, D_1), (b_2, D_2))$

$\Rightarrow \delta : (Q \times \Gamma^k) \to \left(Q \times (\Gamma \times \{L, R\})^k\right)$

Γ - tape alphabet

Now,

A: N

Theore

Proof

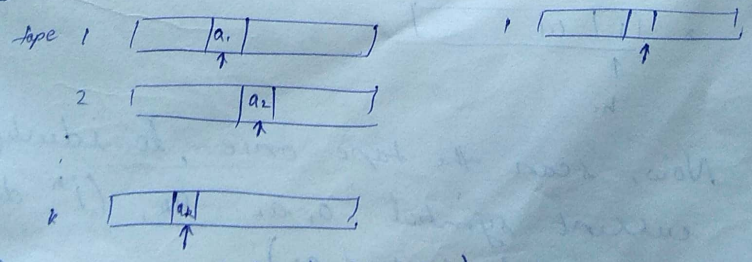
let δ(

Now, Are multi tape turing m/c better in terms of computational power? (i.e. do something in multi tape that can't be done in single tape?)

A: No, Anything solved using multi tape can be solved in a single tape.

Theorem: If a lang. A is decidable (recognisable) with a k-tape turing m/c, then A is decidable (recogni-) with a single tape turing m/c (vice-versa is obvious)

Proof: k type turing m/c (M) single tape (S)



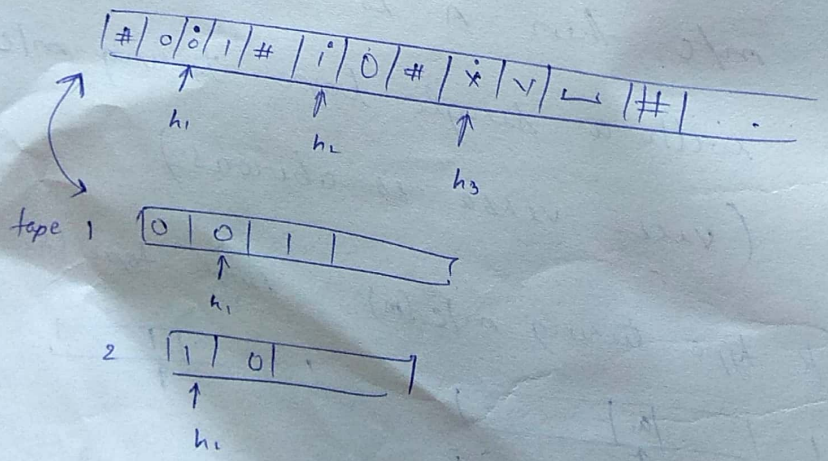
let $\delta(q, a_1, a_2 \ldots a_k) = (\gamma, b_1, b_2 \ldots b_k, D_1, D_2 \ldots D_k)$

Now, we can combine all the k tapes into a single tape one after another seperated by a delimiter # perform transition on it.

Now, how to kep track of k tape heads in single tape., ans: use dotted symbol.

if $\Gamma = \{a, b, x, y, \sqcup\}$

then $\Gamma' = \{a, b, x, y, \sqcup, \dot{a}, \dot{b}, \dot{x}, \dot{y}, \dot{\sqcup}\}$



Now, scan the tape once, to identify current symbol, $a_1, a_2 \ldots a_k$, (i^{th} dotted symbol denoted a_i)

Now, Making changes in symbols or adding symbols may require more than 1 moves in single tape

∴ if we add a symbol to tape 2, then in single tape, we need to shift all symbols after 2nd dotted symbol to right

∴ many moves are required

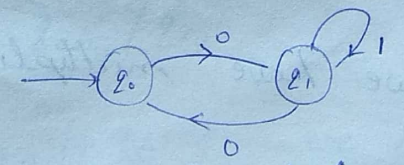
$O(F(n)) \longrightarrow O(F(n)^k)$

$O(F(n))$ — k tape

$O(F(n)^k)$ — single tape

Non-deterministic turing m/c

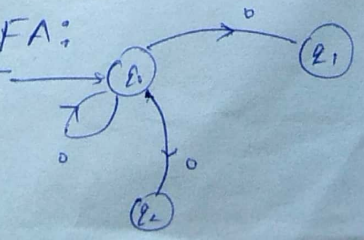
deterministic :



1 move per symbol ✓

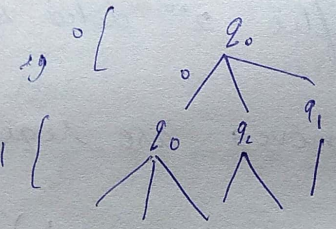
non-deterministic FA:

multiple moves per situation



then how do we say a string is accepted or not in non-deterministic

i.e. M^r to making all possible choice parallely



so, if atleast 1 of the process reach accept state, we accept the string

∴ In non-deterministic turing m/c we have multiple choice of moves

$$\delta(q, a) = \{\underbrace{(r_1, b_1, D_1), (r_2, b_2, D_2)}_{\text{2 choices for this situation}}\}$$

In turing m/c if i/p is 0001, we can't say whether its accepted or rejected by looking at states at 4th move as turing m/c may take multiple moves for a single operation

∴ If any path leads to / ends at q_{accept}, then we accept the string.

But can't say, if any path leads to q_{rej} ⇒ string is rejected

since some path may never terminate, hence we can't say if it eventually reach q_{acc} / q_{rej}

∴ If all path leads to q_{rej} we reject the string

If we has set of states & each tape have its own fns?

Q_1 tape1 ▭ $\delta_1(q_1, a_1) \rightarrow (r_1, b_1, D_1)$

Q_2 tape2 ▭ $\delta_2(\quad) \rightarrow (\quad)$

⋮

Q_n tapen ▭ $\delta_n(\quad) \rightarrow (\quad)$

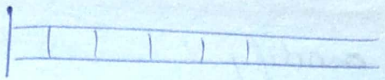
⇒ Now, can we convert it into normal multi tape turing machine? Yes

Let $\langle q_1, q_2, \ldots q_n \rangle$ be a single state

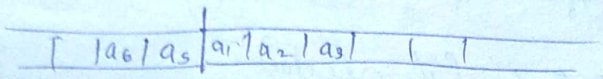
$\in (Q_1 \times Q_2 \times \ldots Q_n)$

$\therefore \delta(\underbrace{\langle q_1, q_2 \ldots q_n \rangle}_{\text{n states assumed as 1 state of } Q = Q_1 \times Q_2 \times \ldots Q_n}, a_1, a_2 \ldots a_n) = (\langle r_1, r_2 \ldots r_n \rangle, b_1 \ldots b_n, D_1 \ldots D_n)$

Also anything done in 2-way infinite tape turing m/c can be done in single tape. how?



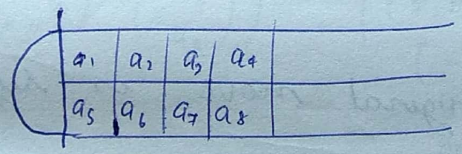
single



how to make
effects on this half reflect in single tape
without affecting other half.

- A raw 2 way tape can be wrapped around to make a 2-track tape that carry 2 symbol in one cell.

new $T' = \Gamma \times \Gamma$



$\therefore$ δ of the form $\delta(q, (a_1, a_5)) =$

Now, how to modify

① Transition fn δ

② Set of states (if needed)

① Now how do we know if had made a move in right half or left half of 2-way ∞ tape?

* give it along Q i.e. $(q, 0)$ $\in Q$

∴ new states $Q' = Q \times \{0, 1\}$ (0: right, 1: left)

Case (i) : original move in right half to right

i.e. $\delta(q, a_1) = (r, b_1, R/L)$

then new fn $\Rightarrow \delta'((q,0), \begin{pmatrix} a_1 \\ a_2 \end{pmatrix})$

$= \left((r,0), \begin{pmatrix} b_1 \\ a_2 \end{pmatrix}\right)$

Case (ii) : original move in left half towards left

$\delta(q_2, a_2) = (r, b_2, L)$

$\Rightarrow \delta'\left((q_2, 1), \begin{pmatrix} a_2 \\ a_3 \end{pmatrix}\right) = \left((r, 1), \begin{pmatrix} b_2 \\ a_3 \end{pmatrix}\right)$

case (iii): a] Original move in right half, Not in left most cell, dis. left.

Here let symbol with '.' above (eg: $\dot{a}_i$) represent element at the beginning of left / right track

eg $\delta'\left((q_2, 0), \begin{pmatrix} a_2 \\ a_3 \end{pmatrix}\right) = \left((r, 0), \begin{pmatrix} \dot{a}_i \\ a_3 \end{pmatrix}\right)$

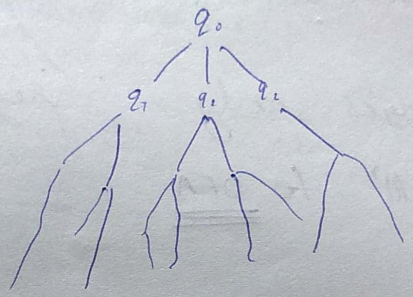
(still on right half (but at left-most end as element is $\dot{a}_i$

n^rly case iv: a] Original move in left half not in rightmost end, to right

For case (iii) b ϕ (iv) b, to move from left half to right ϕ vice versa, how to represent that move using new transition fn?

A: let '.' as symbol above dnt in rightmost cell ϕ * as that above elmnt in leftmost cell.

Now, if our non det turing m/c is like



how to traverse each process.

DFS / BFS ?

Can't use DFS because if the process we chose never ends, the traversal also never ends & we don't get O/P.

If we use BFS, we go level by level. & if any of the states in a level is q_{accept}, we accept it or else we go to next level.

→ A turing m/c with right & stay pulse as direction.

A: recognizes only "regular languages"

i.e., m/c to DFA

Give a proof: DFA ⇔ above turing m/c (both sides)

Q. Design a turing m/c to decide following languages

(a) 0^{2^n} $(n \geq 0)$ i.e., 0
00
0000

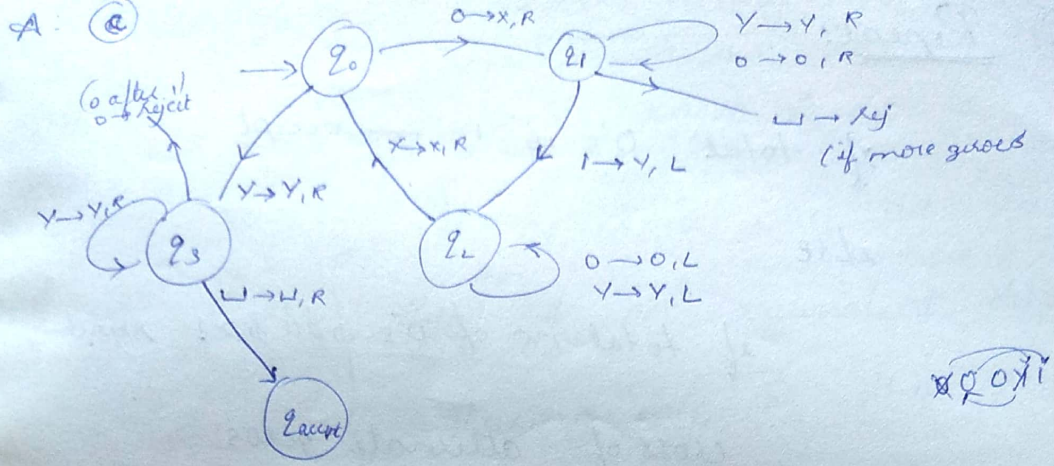
(b) w reflected by w' eg 001/100

(c) $0^N 1^N$ $N \geq 0$

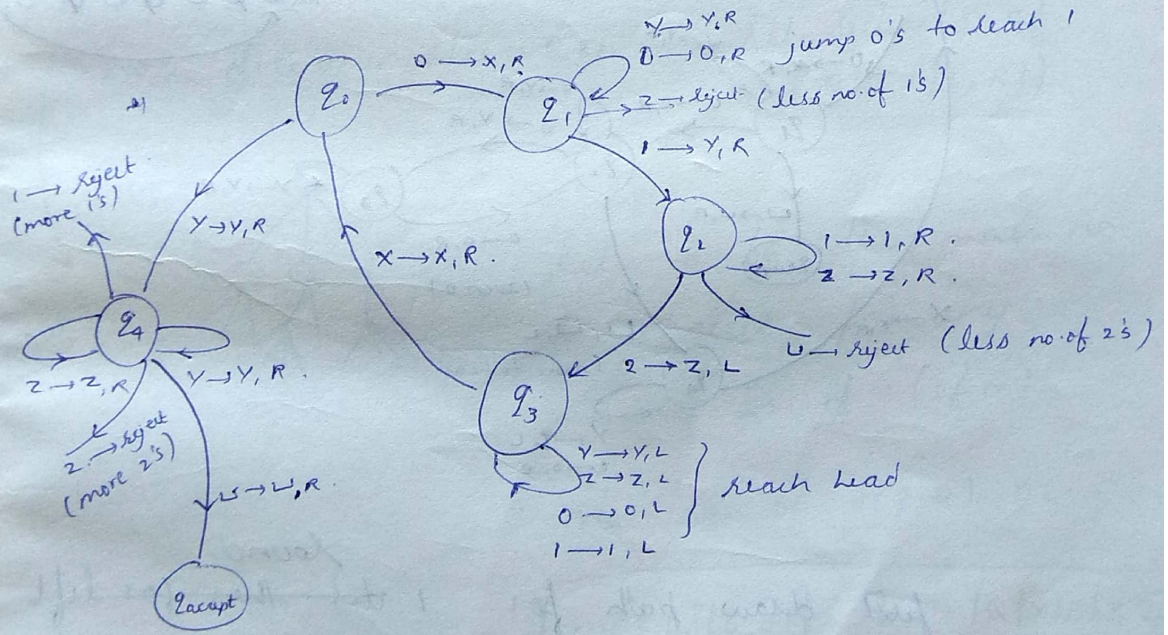
(d) string is palindrome

Scanned by CamScanner

A. (a)



∴ for $0^N 1^N 2^N$ add extra symbol



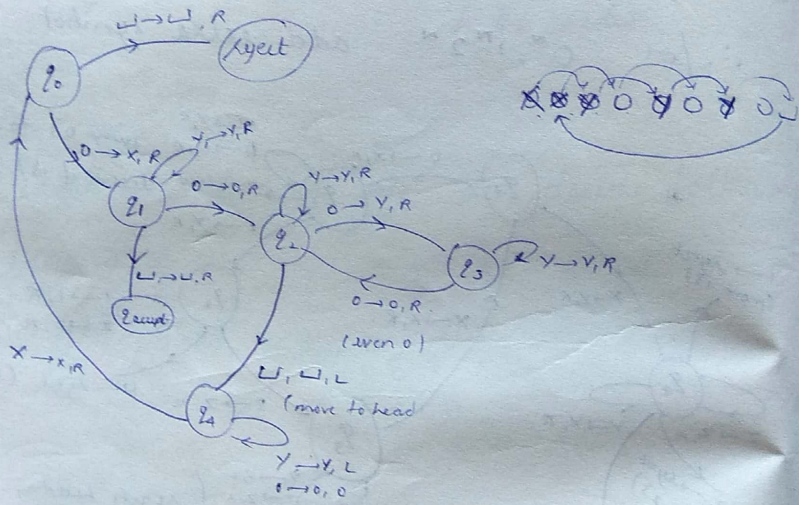
(a) Repeat:

if total 0's is 1 → accept

else

if total no. of 0's odd & > 1 reject

cross of alternate zeros.



first draw path for 1 ~~till then for left~~ found,

then for next

(2 sounds alternating → multiple of 4 at end)

Non-deterministic turing m/c

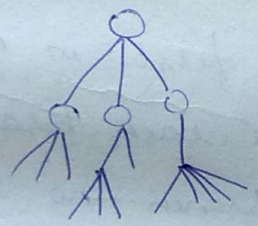
— recognize lang A

— If N is a non-det. turing m/c, ∃ an equivalent determ turing m/c M that recognizes the same lang A.

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

non-deterministic : $Q \times \Gamma \longrightarrow$ all subsets of $(Q \times \Gamma \times \{L, R\})$

PROOF:



Let 'b' be the max. no. of choices for any move in d (of N)

∴ Max size of $\delta(q,a)$ or

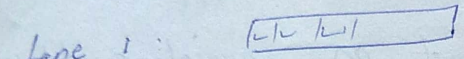
$$\boxed{\max |\delta(q,a)| = b = |Q| \times |\Gamma| \times 2}$$

as $\delta(q,a) = \underset{|Q|}{\dot{Q}} \times \underset{|\Gamma|}{\Gamma} \times \underset{2}{\{L,R\}}$

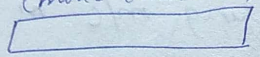
↓ possibility

M - deterministic

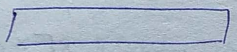
3-tape turing m/c $b \in z^+$



tape 1 : → input tape
→ never changed



tape 2 : → simulation / execution tape



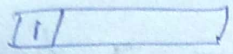
tape 3 : → current branch (in NFA) of execution tape / address tape

Assume we go breadth first in NFA & give number 1,2 for each branch from a state. This branch no. is in tape 3

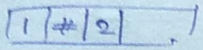
To represent 2nd level branch we use (1,2) like representation

∴

∴ tape 3 is like a dictionary with alphabets ∈ {1, 2 ... b}



⇒ 1st branch 0th level



⇒ move in 1st branch, then follow 2nd branch from these

∴ in k^{th} level we have

$(i_1, i_2 \ldots i_k)$ $i_j \; \forall j \in \{1 \ldots k\} \in \{1, 2 \ldots b\}$

$\Rightarrow i_j^{th}$ choice in j^{th} move.

Now based on tape 3 we make moves in 2 after each move we make value of tape 2 to input using tape 1 to carry out the next choice of move. (as tape 3 has choices from 1st move to current move sequentially) And if any move reaches q_{accept}, then stop & accept

∴ Non-det turing m/c can be implemented using a 3 tape det. turing m/c which inturn can be done using single tape turing m/c.

Hence the proof:

Actual no. of branches / choices for a move depends on i/p

eg: $\delta(q_0, 1)$ can have 2 branch while

$\delta(q_1, 0)$ " " 5 "

Now, if a move for tape 3 is not there (tape 3 updates through all combination of branches) just jump to next move

---

Note: We can use any representation in a turing m/c

eg: 5 = 11111 (unary representation) → for sq. of numbers easier

= 0101 (binary ").

Undecidability

→ What lang are turing decidable

→ Are there lang that are not decidable? (yes)

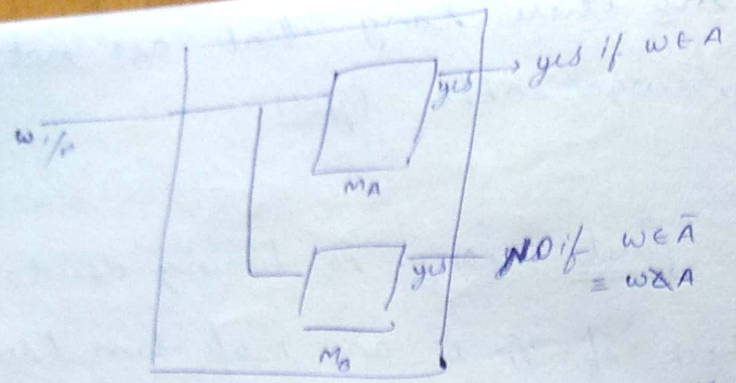
→ Are there lang. that are not recognisable (yes)

A Any finite lang. is turing decidable
(check if I/P is any 1 of them (recursively))
& recognisable also (recursively enumerable

⇒ A turing m/c that recognise A and halts on all I/Ps is turing decidable

⇒ If A is turing decidable, then
⊙ $\bar{A} = \{w \mid w \notin A\}$ is also turing decidable
(change accept states to reject & vice versa)
∴ set of turing decidable lang is closed under complement

⇒ If A & $\bar{A}$ are turing recognisable
then A is turing decidable

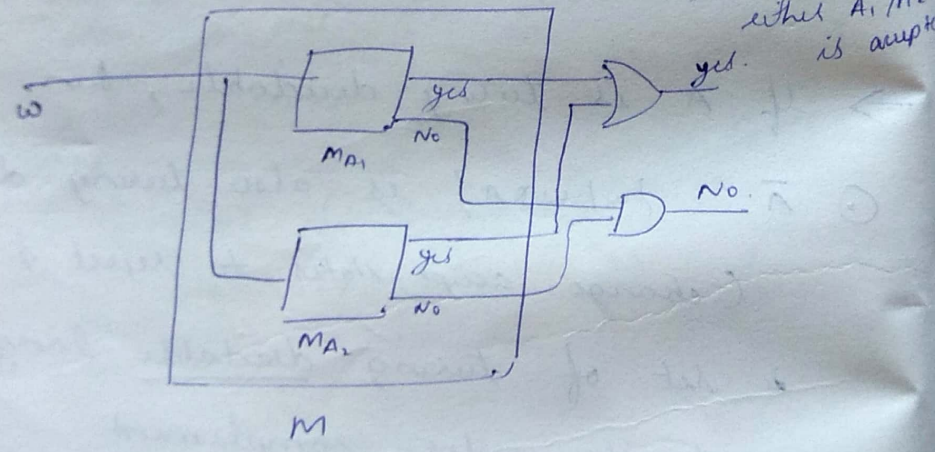


M decides A

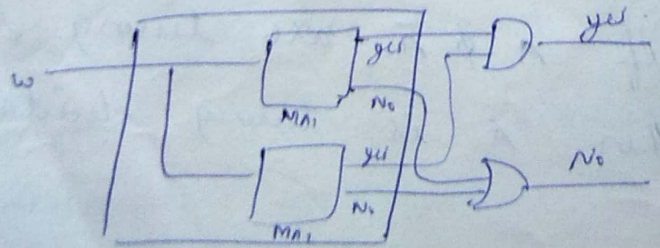
M_A & M_B recognise A

→ If A_1 & A_2 are decidable lang.

⊙ then $A_1 \cup A_2$ is turing decidable

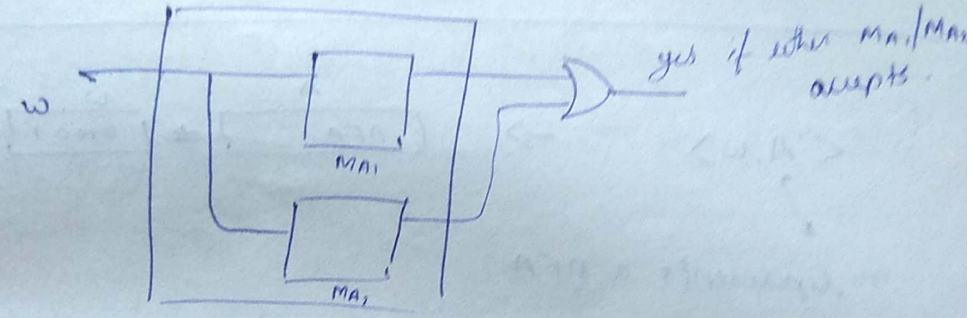


⊙ then $A_1 \cap A_2$ is also turing decidable



⇒ If A_1 & A_2 are turing recognisable

⊙ $A_1 \cup A_2$ is turing recognisable

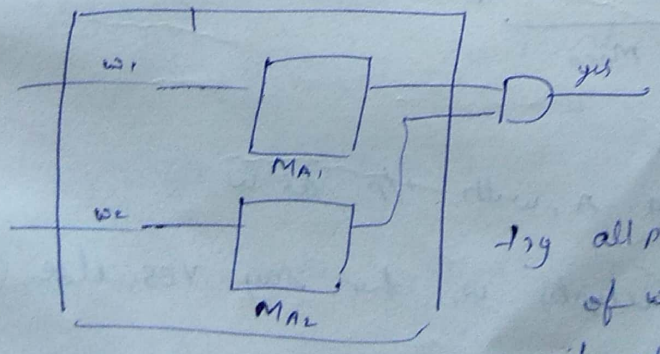


⊙ $A_1 \cap A_2$ is turing recognisable

i accept if M_{A_1} & M_{A_2} accepts
(don't worry about 'no')

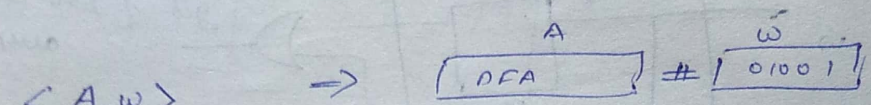
⇒ If A_1 & A_2 are decidable / recognisable

A_1 concatenated with A_2 $(A_1 A_2)$ is turing decidable / recognisable



Try all possible combination of $w_1 \cdot w_2 = w$ & check if atleast 1 gives yes

[pass the head of tape when M_A reaches q_{accept} to M_B & if both accepts, then accept w]



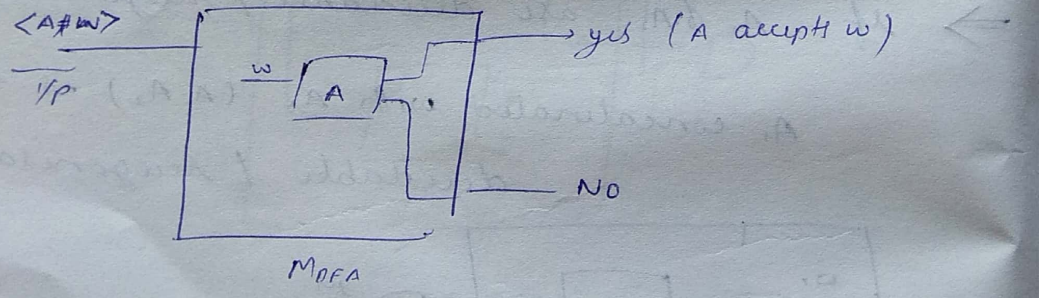
$\langle A, w \rangle$ ⇒

↓

Represents a DFA.

⇒ does A accepts w.

$L_{DFA} = \{ \langle A, w \rangle : A \text{ accepts } w \}$ ⇒ turing decidable



Algorithm

→ Run DFA A, with I/P as w

→ If A accepts w, then say YES, else NO

$L_{TM} = \{ \langle M, w \rangle : M \text{ accepts } w \}$ → universal lang

M is a turing m/c.

L_{TM} is not turing decidable

(M may be non terminating for some w)

Theorem : L_{TM} is undecidable (no way to design it)...

Proof: ① We define a lang such that there is no turing m/c that recognises that lang. (diagonalisation technique)

(if L_{TM} is decidable, then above statement is false)

Idea: ① "Number" all turing m/c s.t each number represent a turing m/c

② Number each I/P string (over {0,1})

How to number I/P string

eg: 0 10
00 1110
010 ?
0001 ?

let $w_1 = 0$ } length 1 string
$w_2 = 1$

$w_3 = 00$
$w_4 = 01$ } length 2 string
$w_5 = 10$
$w_6 = 11$

.
.
.
.

Exercise: Given a number i, what is w_i

$\Rightarrow \lfloor \underbrace{\log_2(i+1)}_{\text{no of alphabets}} \rfloor$ $\quad i -$

(b) Given a string w_i, (i we know its length : & number n) how to find i s.t $w = w_i$?

$$i = (2^k - 2) + (n+1)$$

How to number turing m/c.

eg: $\delta(q_0, 0) = (q_1, \sqcup, R)$

$\delta(q_1, 1) = (q_3, 1, L)$

$\delta(q_3, 0) = (q_{accpt}, 0, R)$

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

Find order of state

& rename states s.t, q_1 (start state), q_2 (accpt state), q_3, q_4 ...

$\rightarrow$ q_0 q_{accpt} q_1 q_3
$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$
q_1 q_2 q_3 q_4

$\Rightarrow \delta(q_1, 0) = (q_3, \sqcup, R)$

$\delta(q_3, 1) = (q_4, 1, L)$

$\delta(q_4, 0) = (q_2, 0, R)$

$\delta(q_1, 1) = (q_1, 1, R)$

$\rightarrow \delta(q_1, x_1) = (q_3, x_3, D_L)$

Also let $\{L, R\}$
$\downarrow$ $\downarrow$
D_1 D_L

$\{0, 1, \sqcup\}$
$\downarrow$ $\downarrow$ $\downarrow$
x_1 x_2 x_3

∴ if we have 5 number i, j, k, l & m

then it corresponds to the move

$\delta(q_i, x_j) = (q_k, x_l, D_m)$

⇒ "$0^i 1 0^j 1 0^k 1 0^l 1 0^m$" → code

↓

0 i times followed by 1 (as delimiter)

∴ $\delta(q_1, x_1) = (q_3, x_3, D_2)$

⇒ 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0

∴ We write code for turing m/c as

111 <mov1> 11 <mov2> 11 <mov3> 11 ...

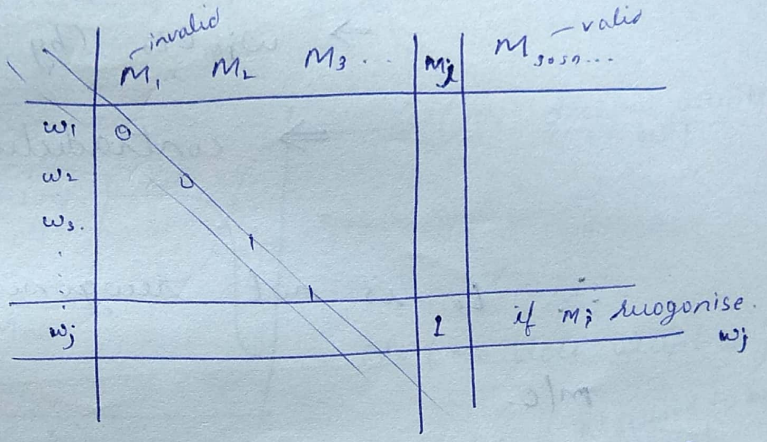
2 1's as delimiter

∴ if rearrange the moves, we could get diff number for same turing m/c

i.e. from a given no. we can get the moves of a turing m/c.

back to proof: Defining a lang. s.t there is no T.M that recognise that lang.

Put turing m/c & strings in order to get a matrix i.e.



Now let $L_{diagonal} = L_d = \{w_i : \text{entry } \langle i,i \rangle = 0\}$

or

$L_d = \{w_i : M_i \text{ does not accept } w_i\}$

Now, prove that L_d is not recognised by any m/c.

let M_j recognises the lang L_d.

Now, From defn of L_d, if $w_j \in L_d$,

case ① → M_j does not accept w_j

But our assumption is that M_j recognise

$L_d \Rightarrow M_j$ accepts $w_j \Rightarrow$ contradiction

② If $w_j \notin L_d$.

M_j recognise L_d only if it accepts strings of L_d only

$\Rightarrow M_j$ donot accept w_j as $w_j \notin L_d$

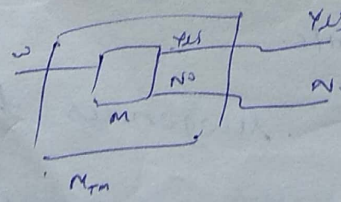
$\Rightarrow w_j \in L_d$ (by defn of L_d)

$\Rightarrow$ contradiction

$\therefore L_d$ is not recognised by any turing m/c.

$\Rightarrow$ Is L_{TM} turing recognisable ?

A. Yes. $L_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$

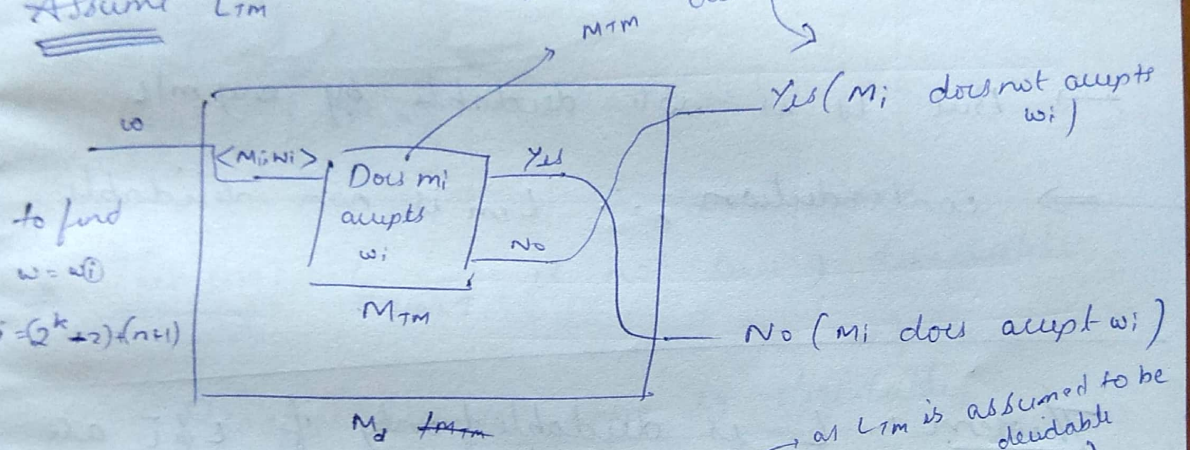


$L_d = \{ w_i : M_i \text{ does not accept } w_i \}$

Cantor $\langle i, i \rangle = 0$.

Algorithm (T/M) for L_d

Assume L_{TM} is decidable by M_{TM} decides L_{TM}



∴ Algorithm is: (Assume M_{TM} decides w_i → as L_{TM} is assumed to be decidable)

$\forall w_i \in L_d$.

— Find i such that $w = w_i$

— Give $\langle M_i, w_i \rangle$ as I/P to M_{TM}

If M_{TM} says yes, ans No & vice versa.

⇒ M_d decides L_d but we proved that

No T/M decides L_d

∴ contradiction ⇒ M_{TM} is not decidable ∀ L_{TM}

* what we have done is:

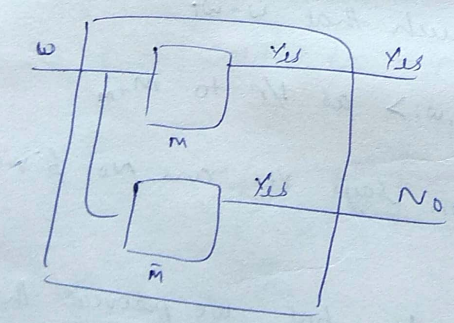
→ Assume L_{TM} is decidable. then

→ show that L_d is also decidable (using a m/c that decides L_{TM})

→ But L_d is not decidable by any m/c

⇒ contradiction ∴ L_{TM} is non decidable.

Theorem: L is decidable if only if $L \& \bar{L}$ are recognisable



converse is obvious

---

* ⇒ since L_{TM} is undecidable, $\bar{L_{TM}}$ is not recognisable //

HALTING PROBLEM

Given a string m/c M & string w

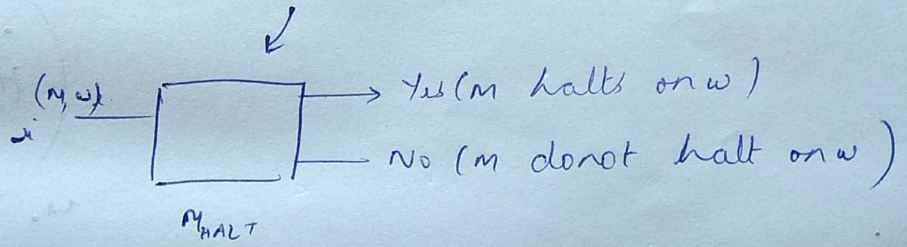
Does M halt on w (i.e., goes to accept/reject

$L_{HALT} = \{ \langle M, w \rangle : M \text{ halts } w \}$.

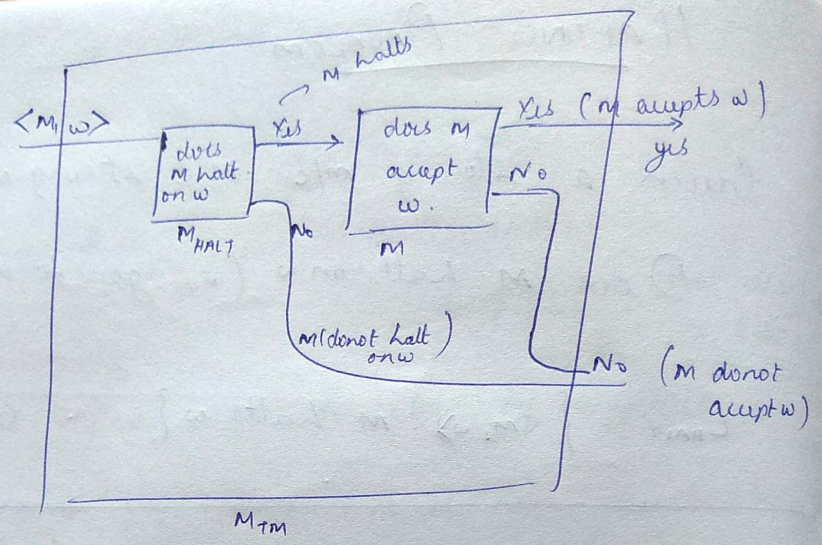
★ Theorem: L_{HALT} is not turing decidable

Proof: Assume L_{HALT} is decidable

(i.e., ∃ a TM M_{HALT} that decides L_{HALT})



Then we can make a TM that decides L_{TM}. (i.e undecidable as we couldn't say if M stops or not)



But L_{TM} is undecidable (∵ no TM decides L_{TM})

$\Rightarrow$ M_{HALT} is undecidable