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:
proving correctness of a mathematical statement
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

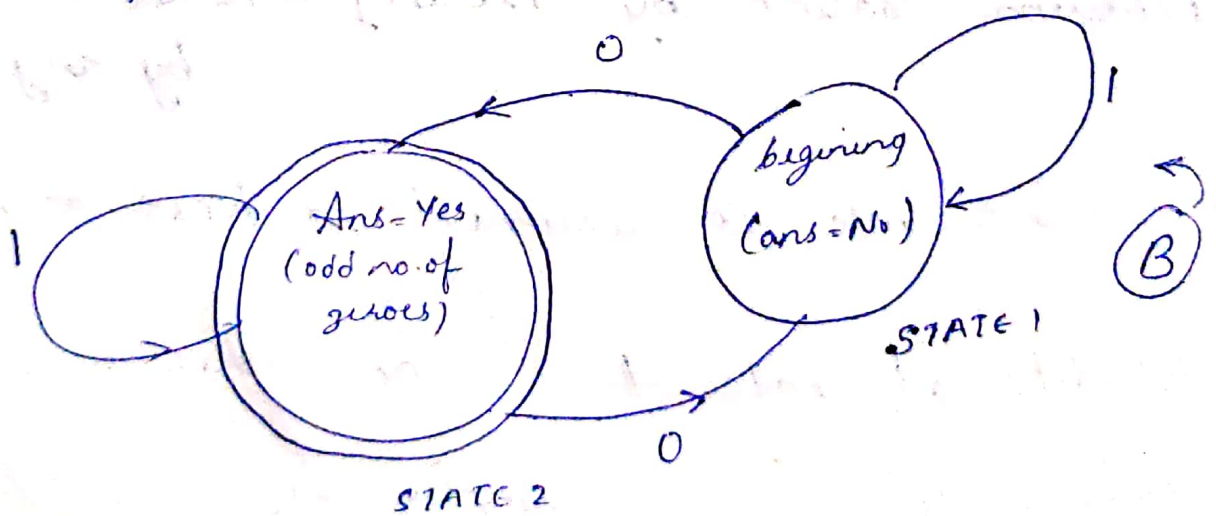
simple m/c	complex m/c
calculator (solve some problem)	Turing m/c.

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 zeroes.

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



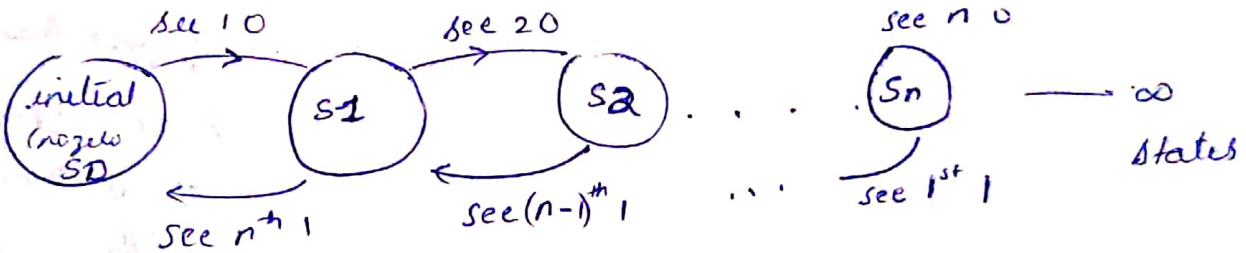
— finite automata

here m/c B "decides" language A

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

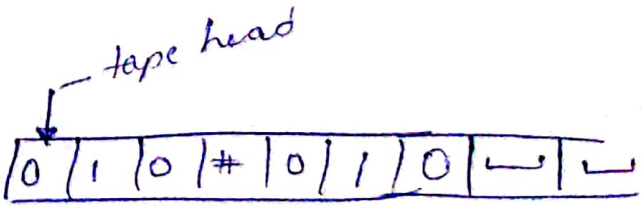
∴ let A' = lang has string of the form w#w
w is string with 0, 1 or #

If the m/c doesn't need finite no. of state,



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

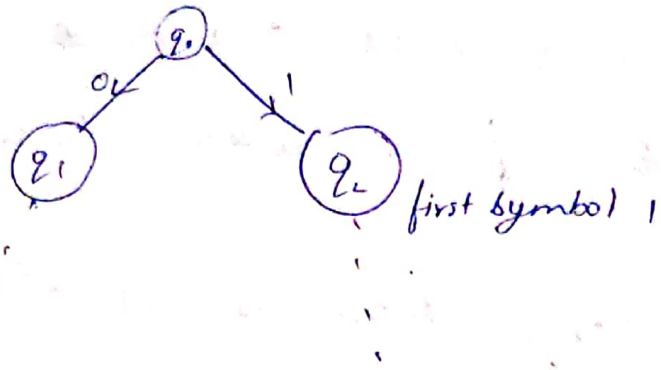
A Turing m/c has



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

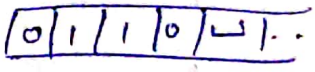
Once you see has
compare next cha
with 1st one
if same cross both els
say ans = no
if yes then compa
next char after
without cross
with 1st symbol
from 1st without
cross

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 $\{0, 1, ␣\}$

i) $\Sigma \subseteq \Gamma$ & $␣ \in \Gamma$

⑥ Finite set of state (works for any I/P)

⑦ q_0 - initial state
q_{accept} - accept "
q_{reject} - reject state
$\} \in Q$ $q_{accept} \neq q_{reject}$

⑧ transition fn (or "moves" / "steps")

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

q → current state
a → current symbol
r → new state
b → new symbol
o → direction of movement

Q] $Q = q_1, q_2, q_3, q_{acc}, q_{rej}$

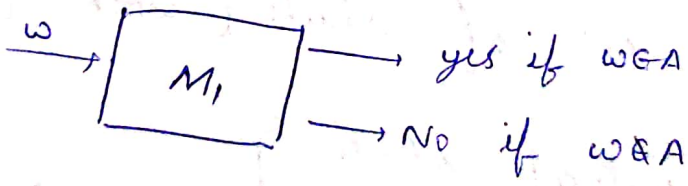
$\Gamma = \{0, 1, x, \sqcup\}$ — symbol that can be seen in tape

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

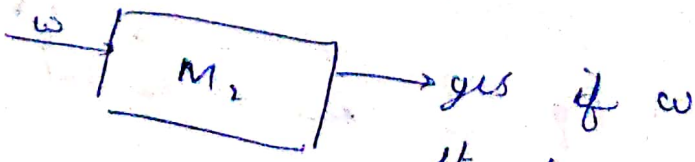
we need to define transition for each state & symbol. If not its assume to stop there & go to q_{rej}.

(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.



It doesn't say yes if w∉A
(i.e, it doesn't say no)

here M_1 "recognise" lang A

i.e. If $w \in A$, it may enter ∞ loop i.e. donot stop.

⇒ lang A is Turing decidable if ∃ M which decides A (recursive)

& lang A is Turing recognisable if ∃ M' which recognizes lang A (recursively enumerable)

---

set of Turing decidable ⊆ 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 $0^n1^n = A$.

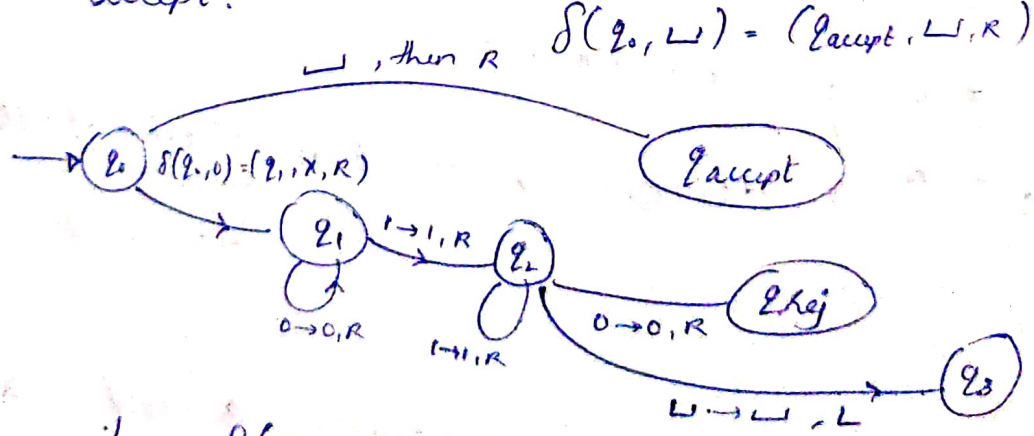
ⓐ scan from left → right

ⓔ 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 $0 \rightarrow x$ to know tape head

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

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

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

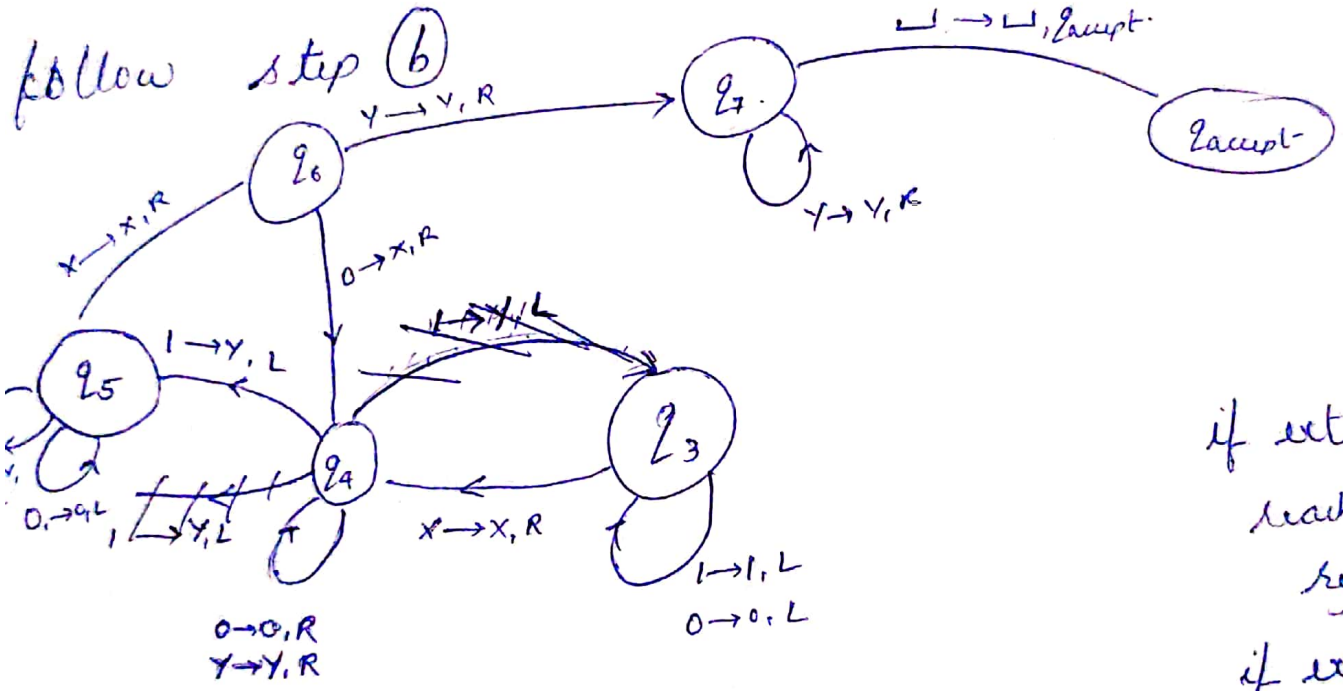
$\delta(q_1, 0) = (q_1, 0, R)$

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

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

Undefined transition $\rightarrow q_{reject}$

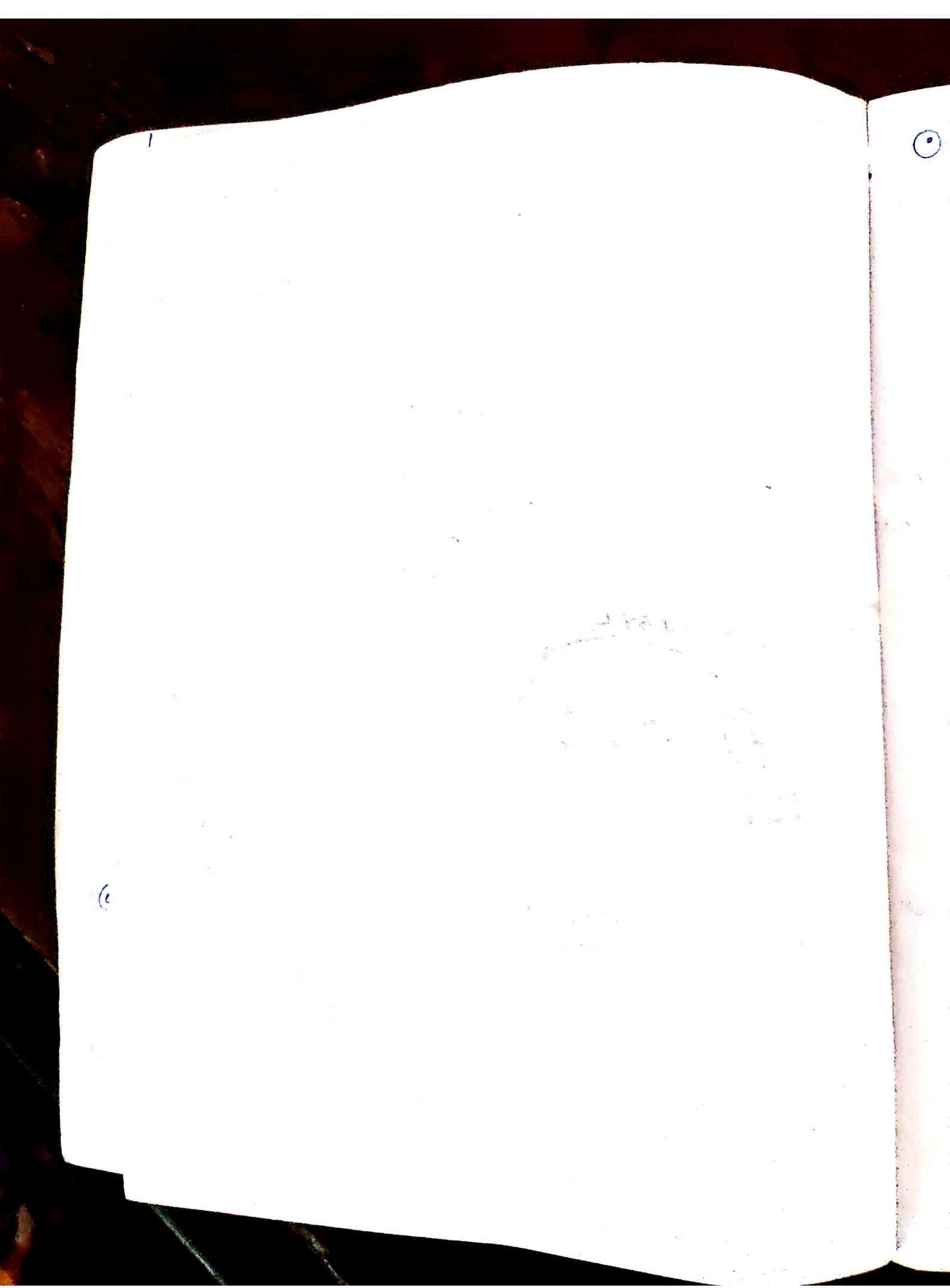
Now, one it reaches $\sqcup$ in q_2 we have to move back and follow step (b)



if extra 1, reach q_7 & reject

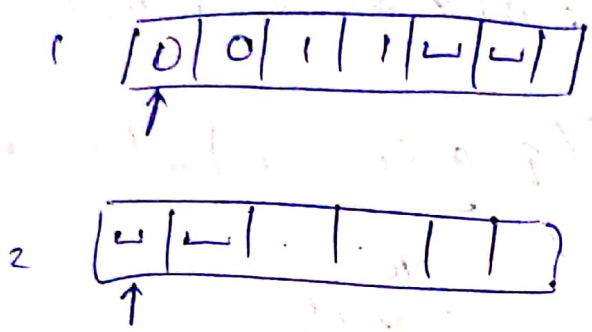
if extra 0, reject at q_4

$O(n^2)$



⊙ We considered Single tape deterministic turing m/c.

Now what if we had 2 tapes with input & 1 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.

$\Rightarrow O(n)$

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

Here: ⊙ Tape 1 has i/p followed by ␣

⊙ Tape 2-k has ␣ initially

⊙ set of states Q, $(q_{accept}, q_{rej}, q_0) \subseteq Q$ — states same

Now, transition fn / moves,

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

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

(r: new state; $b_1 b_2 \ldots b_k$: new symbol; $D_1 D_2 \ldots D_k$: direction; q: cur. state; $a_1 a_2 \ldots a_k$: 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

(Γ - tape alphabet)

$$\delta(q, a_1, a_2) = (r, (b_1, D_1), (b_2, D_2))$$

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

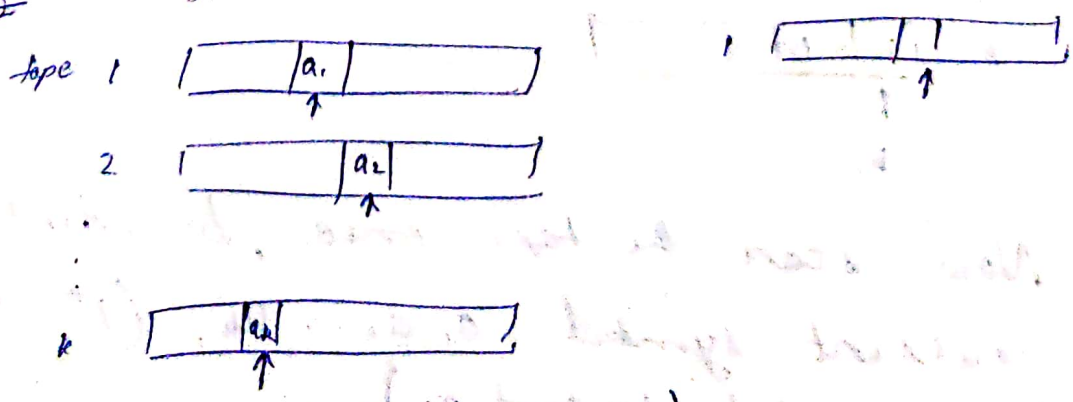
Now, Are multi tape turing m/c better

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)



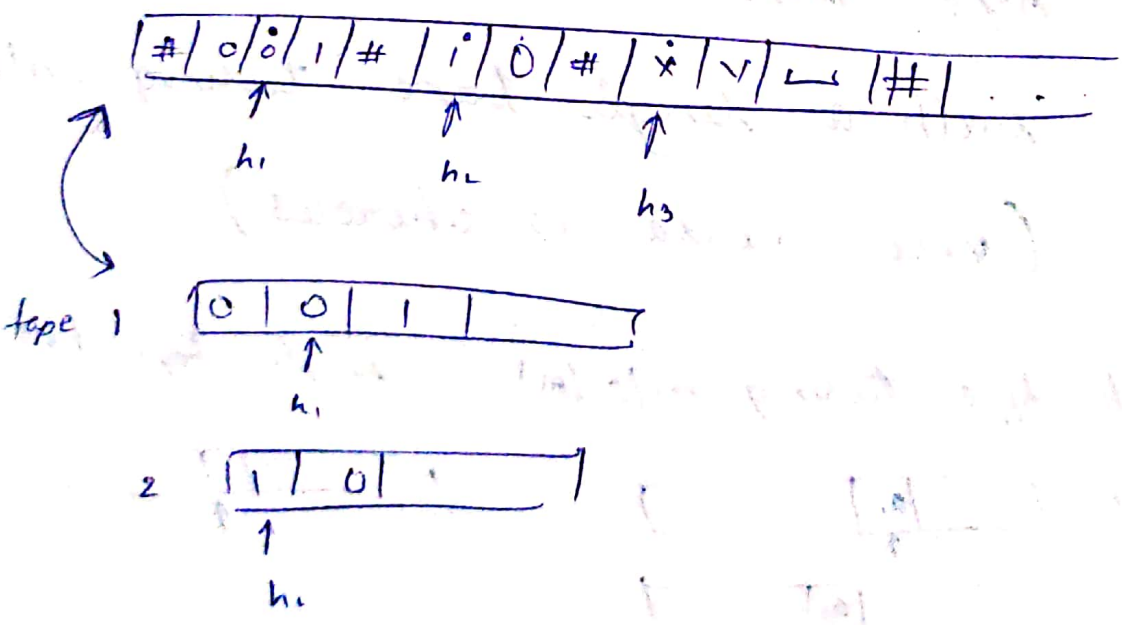
let $\delta(q, a_1, a_2 \ldots a_k) = (\gamma, b_1, b_2 \ldots b_k, D_1, D_2 \ldots D_k)$

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

Now, how to keep 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

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

∴ many moves are required

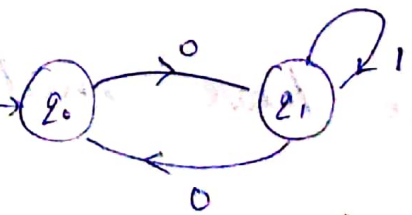
$O(f(n)) \longrightarrow O(f(n)^k)$

$O(f(n))$: k tape

$O(f(n)^k)$: single tape

Non-deterministic turing m/c

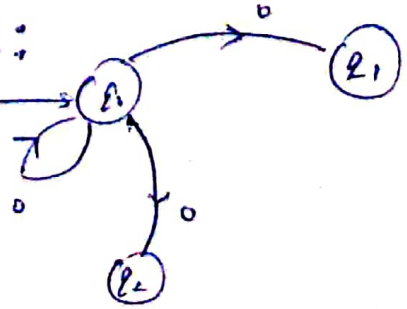
deterministic :



1 move per symbol ∴ ∈

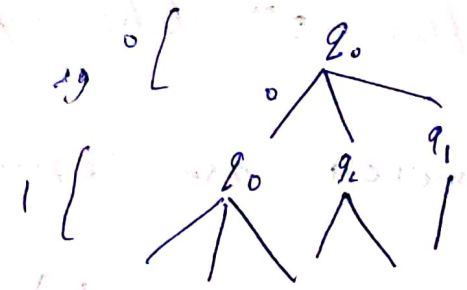
non-deterministic FA:

multiple moves per situation



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

i.e. m/c 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

$$\delta(q, a) = \{\underbrace{(r_1, b_1, D_1), (r_2, b_2, D_2)}\}$$

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

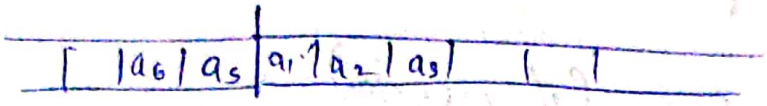
let $< q_1, q_2, \ldots q_n >$ be a single state

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

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

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

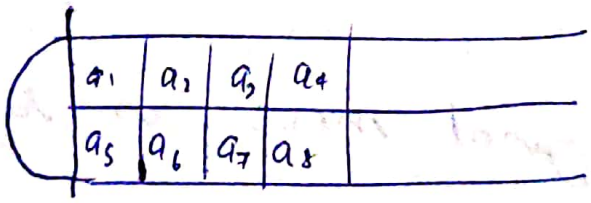
single



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?

$\in Q$

$\star$: give it along Q i $(q, 0)$

right

$\therefore$ new states $Q' = Q \times \{0, 1\}$

left

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

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

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

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

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

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

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

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

here let symbol with '.' above (eg: $\dot{a_1}$) represent element at the begenning of left / right track.

eg $\delta'\left((q_2, 0), \begin{pmatrix} a_2 \\ a_3 \end{pmatrix}\right) = \left((r, 0), \begin{pmatrix} \dot{a_1} \\ a_3 \end{pmatrix}\right)$

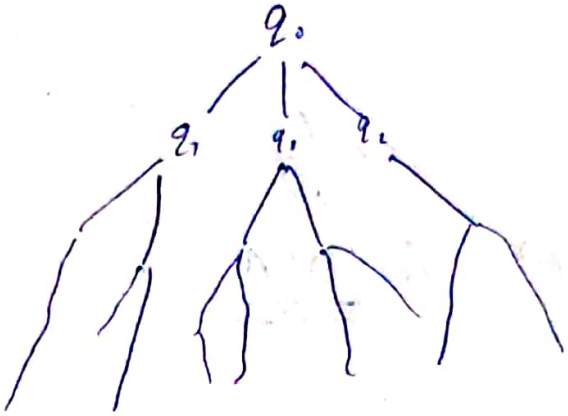
(still on right half (but at left-most end as element is $\dot{a_1}$

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

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

A. Let '.' as symbol above elmt in rightmost cell & * 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 pull(s) as direction.

A Recognizes only "regular Language" i.e. 'M' to DFA

Qui a proof: DFA ⇔ above turing m/c both sides

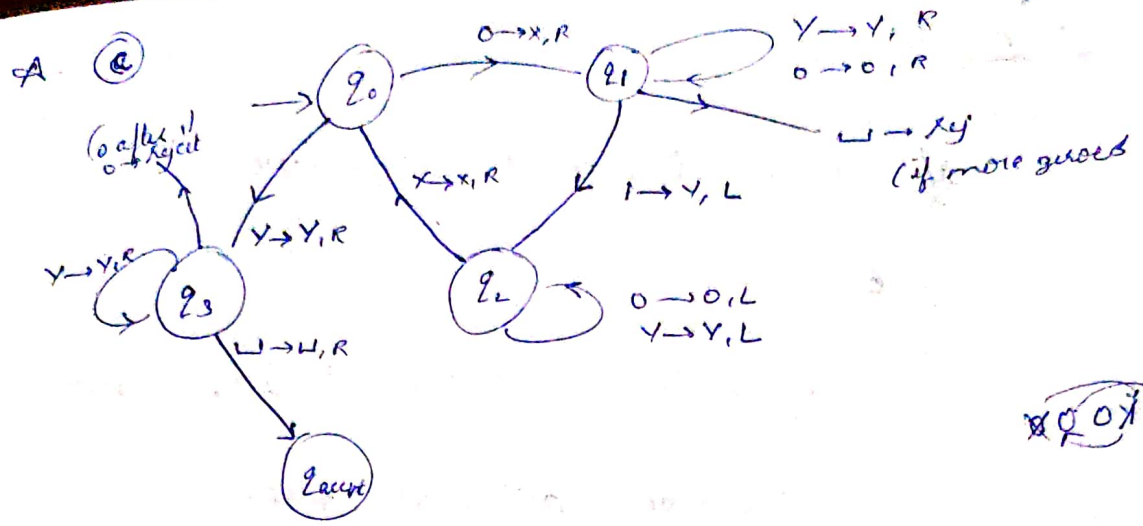
Q. Design a turing m/c to decide following languages

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

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

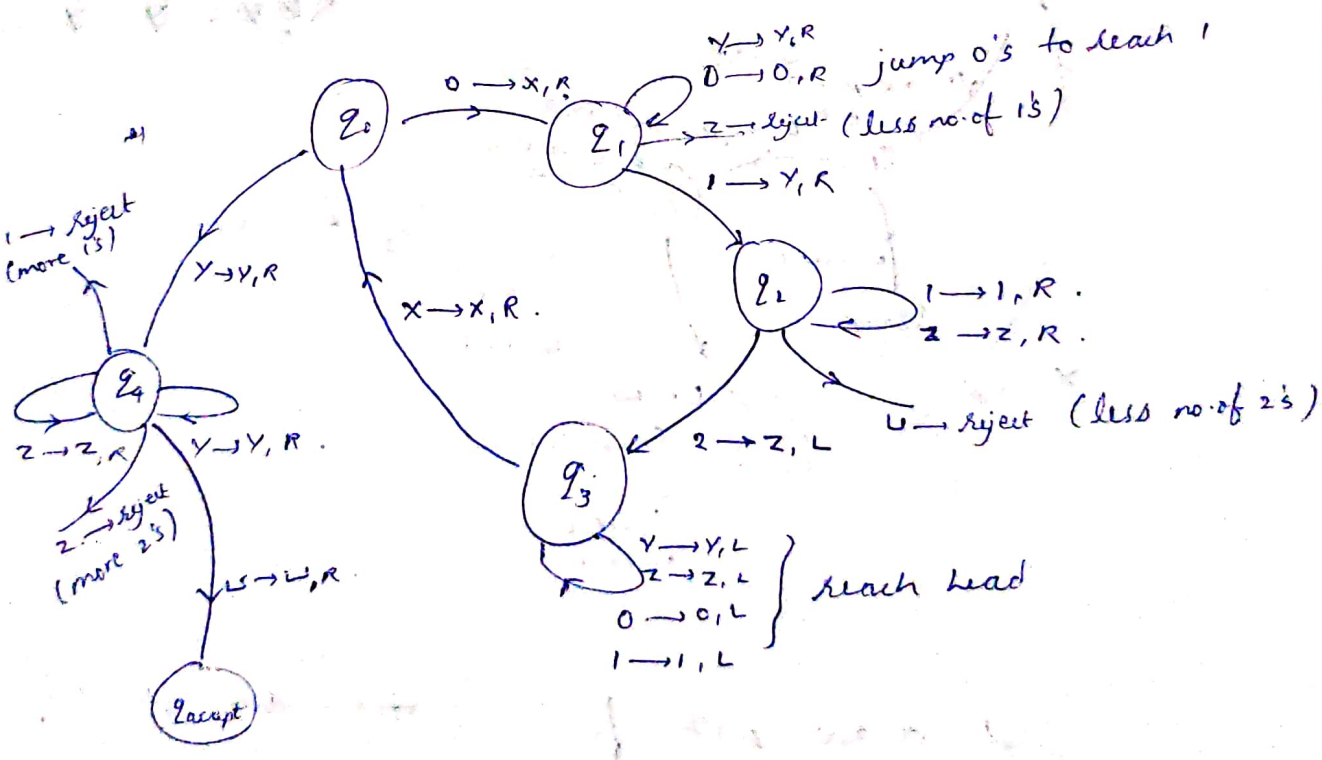
(c) $0^N 1^N$ $N \geq 0$

(d) String is palendrome



$0^n 1^n$

∴ for $0^N, 1^N 2^N$ add extra symbol



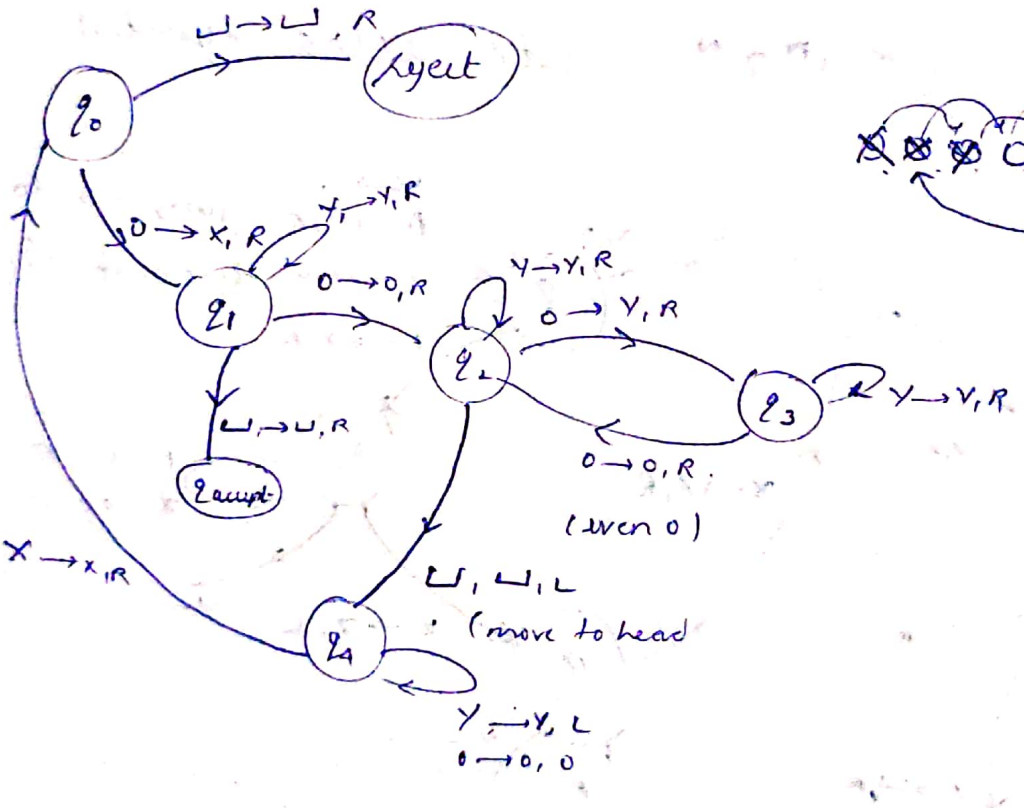
(a) Repeat:

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 ~~then for left~~ round.

then for next

(2 rounds alternating → multiple of 4 at end)

Non - deterministic turing m/c.

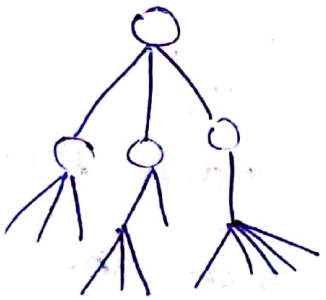
— recognize lang A

— If N is a non. det. turing m/c, ∃ an equivalent determin. turing m/c M that recognize 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

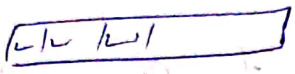
$$\max |\delta(q,a)| = b = |Q| \times |\Gamma| \times 2$$

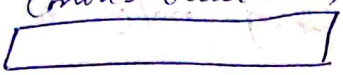
as $\delta(q,a) = \underbrace{Q \times \Gamma \times \{L,R\}}_{|Q| \quad |\Gamma| \quad 2}$

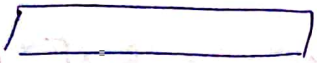
/ possibility

M - deterministic

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

tape 1 :  → input tape
→ never changed.

tape 2 : (moves occur here)  → simulation / execution tape

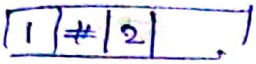
tape 3 :  → current branch of (in NFA) execution tape / address tape

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 there

∴ in k^{th} level we have

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

⇒ 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. And if any move (as tape 3 has choices from 1st move to current move sequentially) reaches qaccept, then stop & accept

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

Here the proof:

Actual no. of branches / choices for a move depends on I/P

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

$\delta(q_0, 0)$ " " 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. $s = 11111$ (unary representation) → for sq. of number easier

0101 (binary ").

Undecidability

→ What lang are turing decidable

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

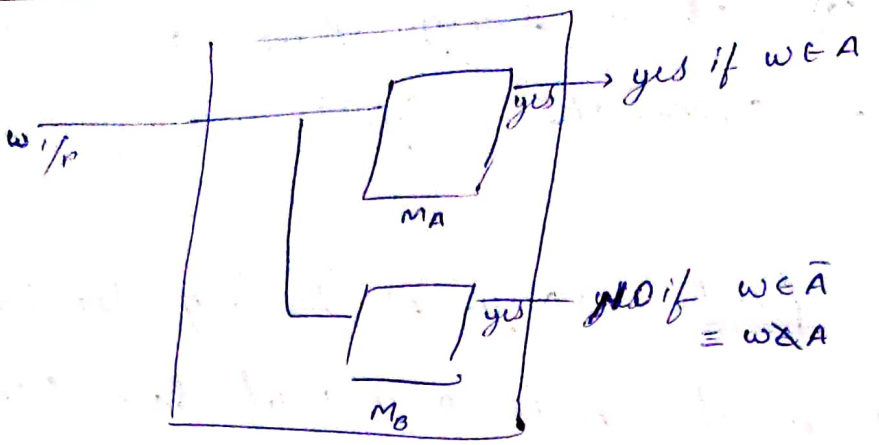
→ Are there lang. that are not recognisable (yes)

∴ 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 accpt state 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

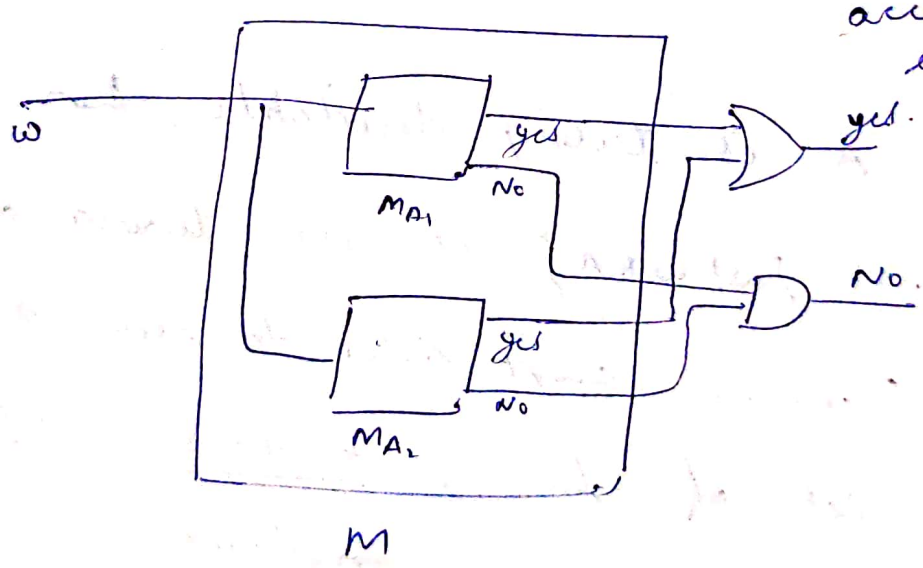


M decides A

M_A & M_B recognise A

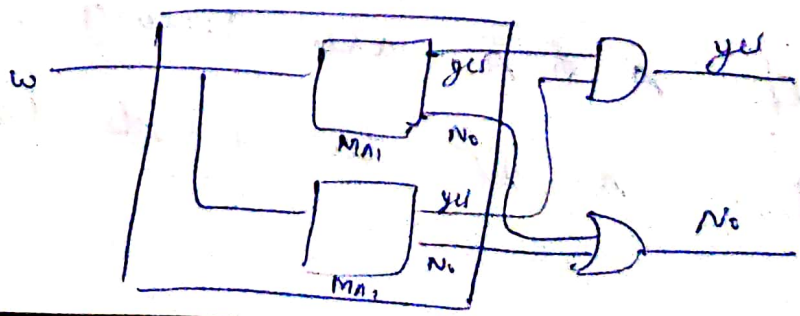
⇒ If A_1 & A_2 are decidable lang.

⊙ then $A_1 \cup A_2$ is turing decidable



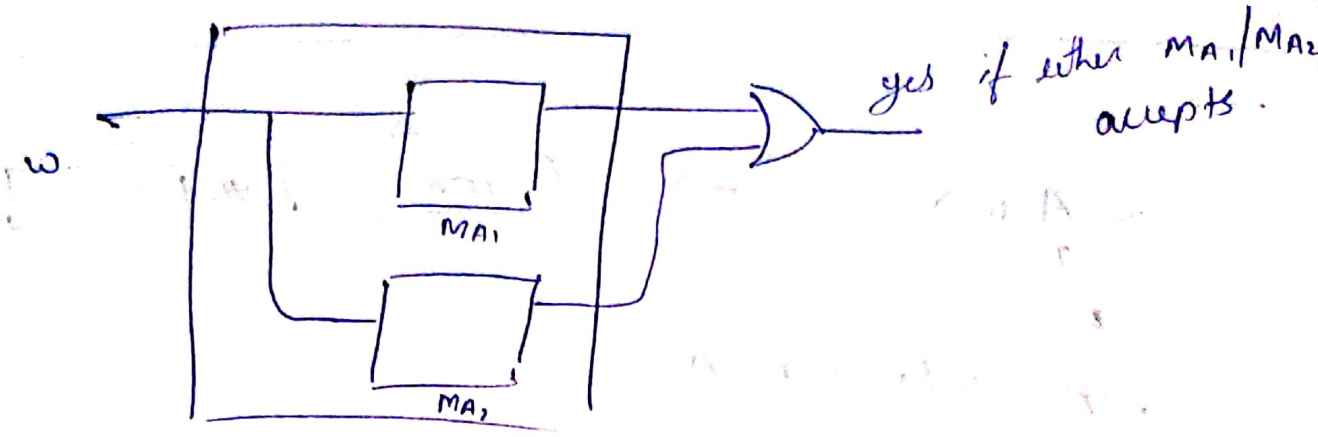
M

⊙ 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

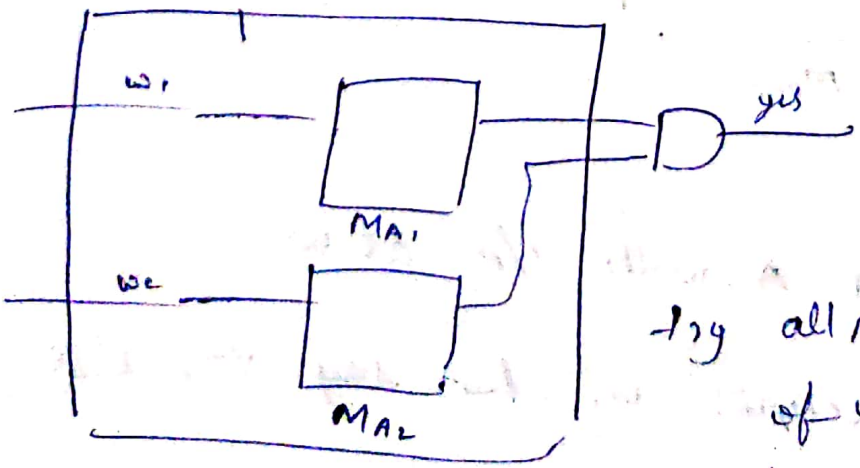


⊙ $A_1 \cap A_2$ is turing recognisable

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

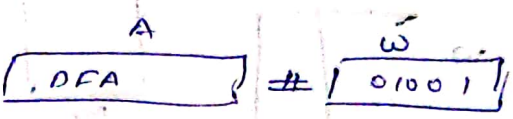
⇒ If A_1 & A_2 are decidable / recognisable

A_1 concatenated with A_2 ($A_1 \cdot A_2$) 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]

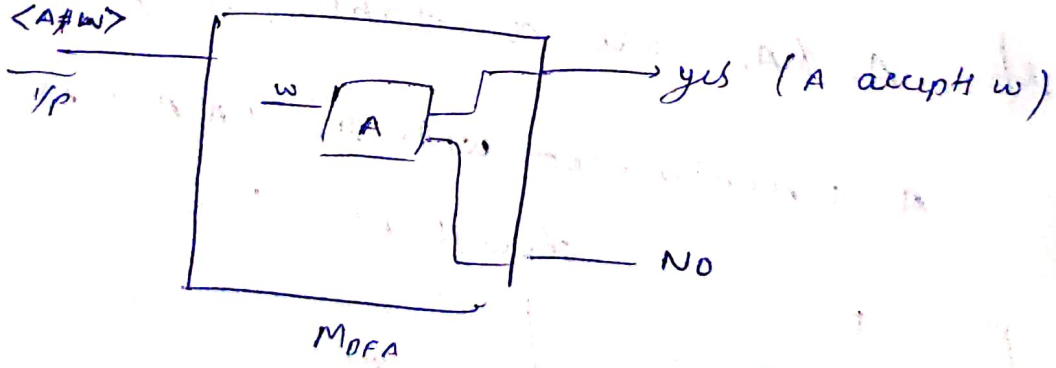
$\langle A, w \rangle$ ⇒ 

↓

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

$L_{TM} = \{ \langle M, w \rangle : M \text{ accepts } w \} \rightarrow$ universial 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 defini 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

y: 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

$n \Rightarrow \lfloor \log_2(i+1) \rfloor$ (no. of alphabets)

$(i-1) - \left(\sum_{j=1}^{n-1} 2^j\right)$

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

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

(if i starts from 0)

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

i rename states s.t. q_1 q_2 q_3 q_4 ...

q_1 — start state; q_2 — accpt state

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

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

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

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

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

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

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 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 1 0 0

∴ We write code for turing m/c as

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

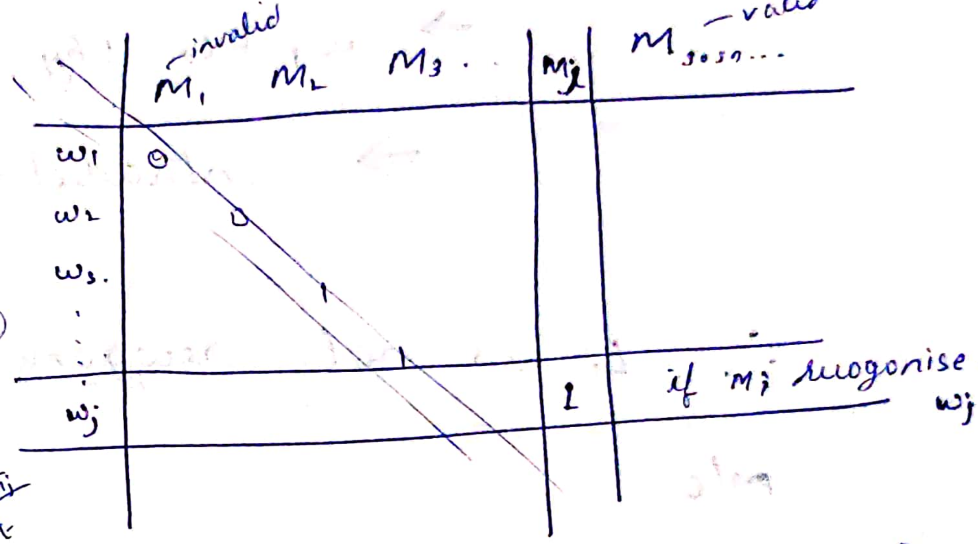
2 1's as delimiter

∴ 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.



Set of all strings that are not accepted by turing m/c from $M_1 - \infty$ (recheck)
∴ ~~if we take any m/c M_j ... it has a string that is not accepted by M_j~~ in it

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. $\in (M_1, M_2 \ldots M_\infty) \rightarrow$ set of all turing 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 recognises L_d $\Rightarrow M_j$ accepts w_j $\Rightarrow$ contradiction

(c) 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 $w_j \notin L_d$

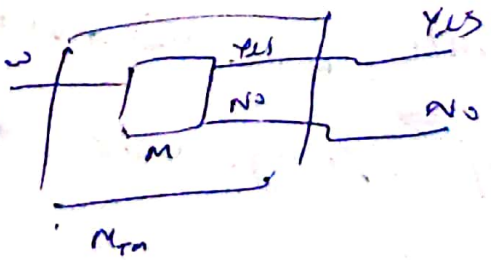
$\Rightarrow w_j \in L_d$ (by defn of L_d)

$\Rightarrow$ contradiction

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

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

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

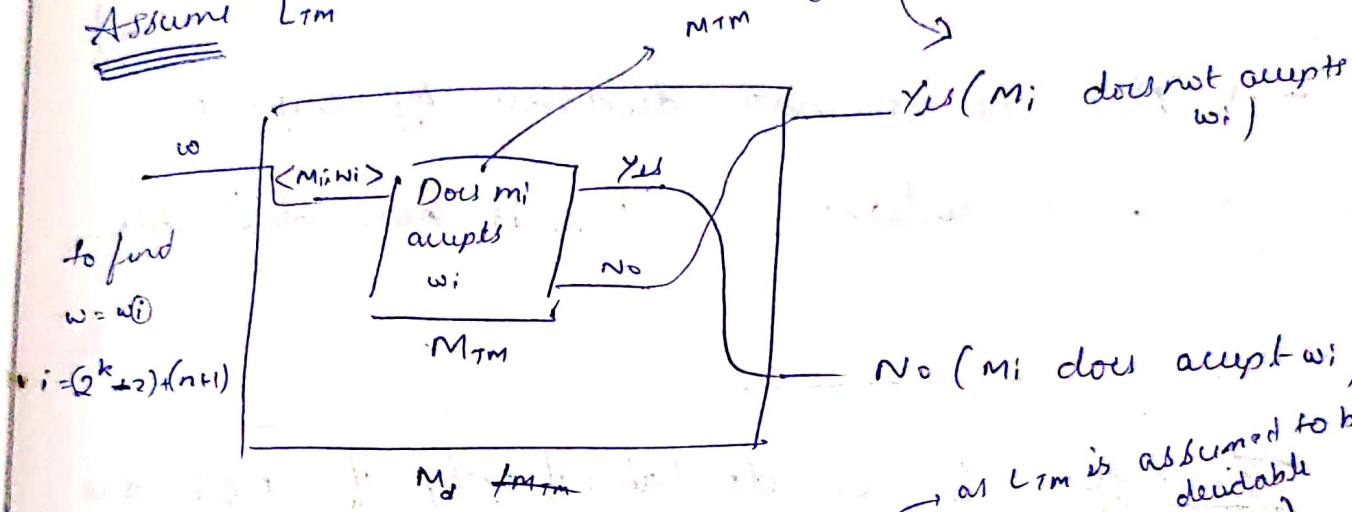


$L_d = \{ w_i : M_i \text{ does not accept } w_i \}$

Century $\langle i, i \rangle = 0$

Algorithm (T/M) for L_d

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



→ as L_{TM} is assumed to be decidable

∴ Algorithm is: (Assume M_{TM} decides w_i) $\forall w_i \in L_d$.

— 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:

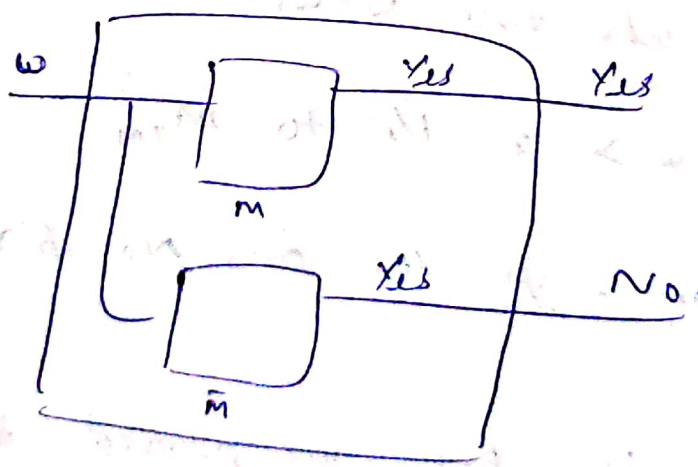
→ Assume L_{TM} is decidable. then

→ 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



if $\bar{L}_{TM}$ = {<M,w> / M does not accept w}
if $\bar{L}_{TM}$ is recognisable
L_d is also recognisable
not possible
↳ ∴ $\bar{L}_{TM}$ not recognisable

converse is obvious

→ since L_{TM} is undecidable, $L_{\bar{TM}}$ is not

HALTING PROBLEM

Given a string m/c M & string w

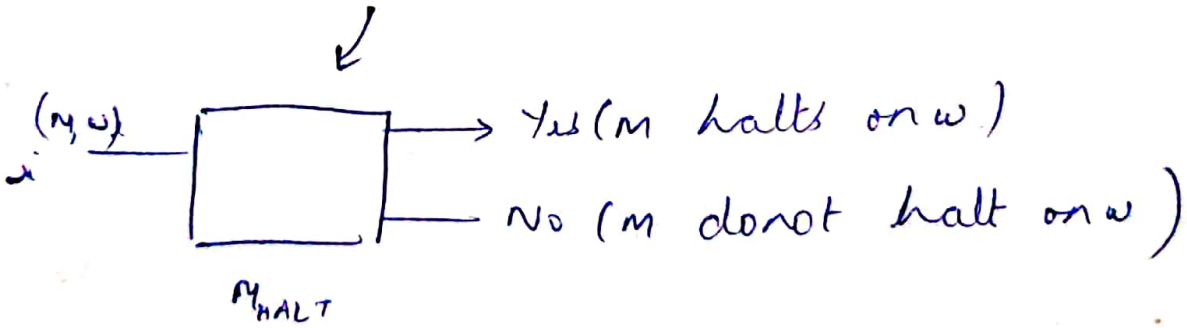
Does M halt on w (i.e., goes to accept/reject)

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

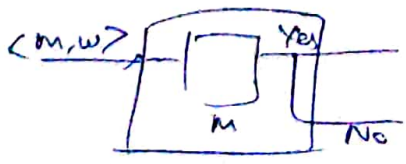
★ **Theorem:** L_{HALT} is not turing decidable

Proof: Assume L_{HALT} is decidable

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

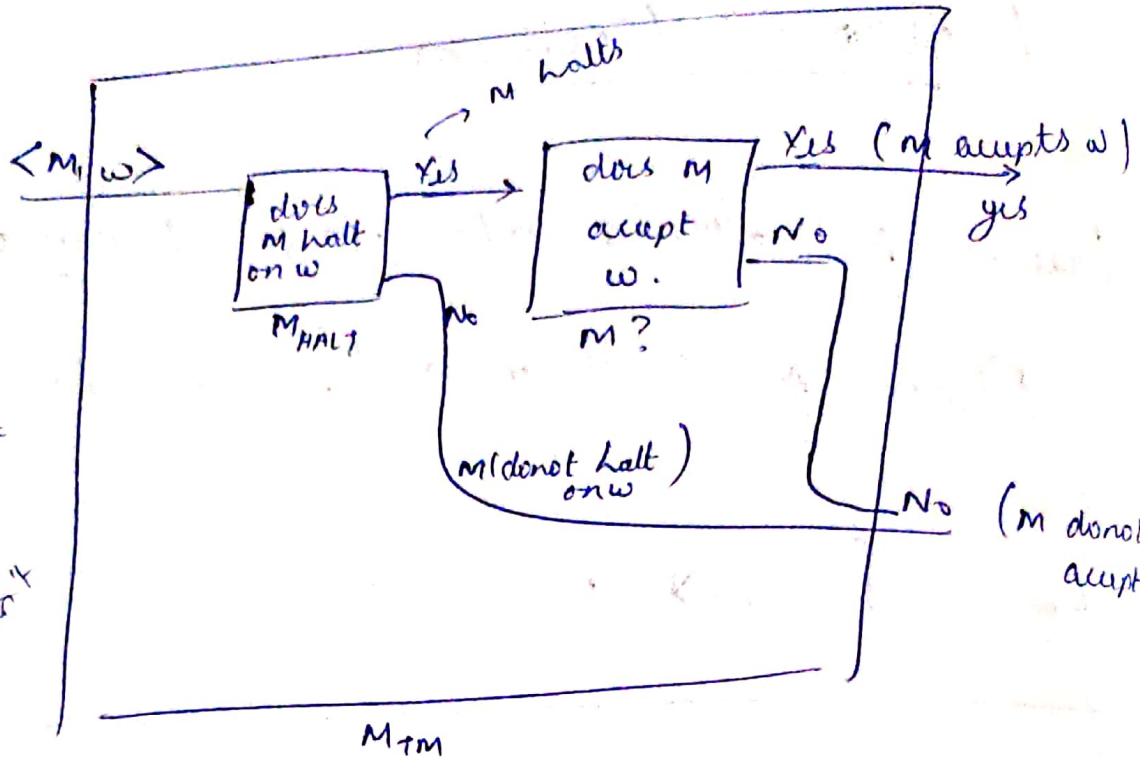


Then we can make a TM that decides L_{TM}. (it is undecidable as we couldn't say if M stops or not)



→ not possible as we don't know if M halts or not

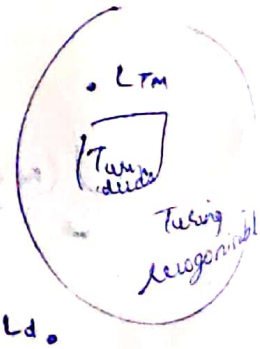
← if M halts, its possible to make this



M decides whether M accept w or not then what is M_{TM}? we can't but intially we can't

But L_{TM} is undecidable (i.e no TM decides L_{TM})

$\Rightarrow$ M_{HALT} is undecidable



Q. Given a TM M, is the lang recognised by M i.e., L(M) regular.

i.e. $L_{REG} = \{ M : L(M) \text{ is regular} \}$

eg: • M_1 = accepts - string of the form 0^n1^n

M_2: accepts string with odd no of 0s

M_3: Accepts w#w

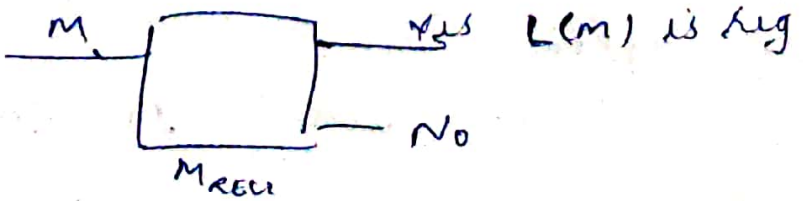
M_4: accepts all string ...

Here $M_2 \& M_4 \in L_{REG}$ can make DFA for them.

[pumping lemma can be used to check if a lang is regular.]

S.T L_{REG} is undecidable

Proof: Let M_{REG} decide L_{REG}.



Build M'_w s.t based on whether M'_w is regular we can say M accepts w or not

M'_w : Accept non regular language (eg: 0^n1^n) if M does not accept w

Accept: regular lang if M accept w

If M'_w : $\begin{cases} x \neq w & \text{reject} \\ x = w & \text{run M with w} \end{cases}$

no irregular lang:
- if M accepts w, is M'_w accept regular lang $L(M'_w) = \{w\}$
- if M does not accept w, M'_w accept no string $\{\}$

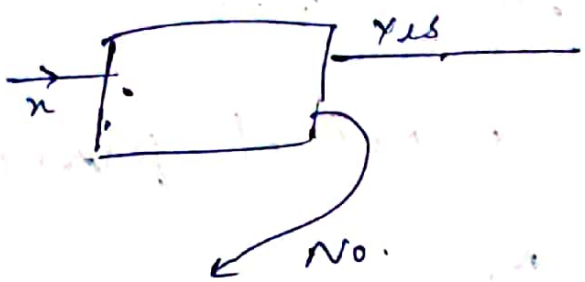
If let, if M accept w, M'_w accept all string

else

it accept strings of the form 0^i1^i not regular

Step 1 :

→ Accept all strings that are of the form 0^n1^n

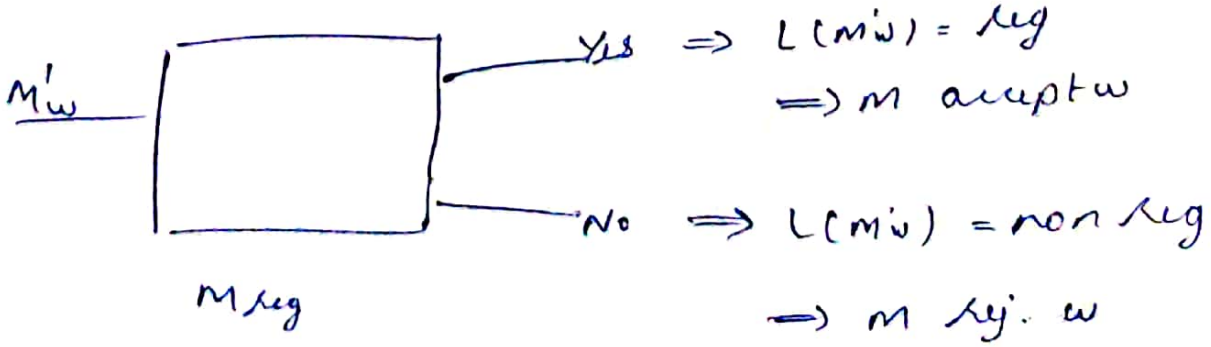


→ Step 2

if $x \neq 0^n1^n$, run w on M, ⟶ Yes accept x

∴ if m accepts w, then M_w accepts all string ⟶ regular.

if m does not accept w. it accepts only strings of the form 0^n1^n ⟶ not regular



Yes ⇒ L(M'w) = reg
⇒ M accept w

No ⇒ L(M'w) = non reg
⇒ M rej. w

$M_{1\_w_1}$: if $x = 0^n 1^n$ accept

if not

run M with w_1.

if x NOT of form $0^n 1^n$

M_1 :

$L(M_{1\_w_1})$: $0^n 1^n$ + all other strings → all string

M_2 accept all string either in step 1 or in step 2.

if

Time complexity of TM

NP — Non-deterministic polynomial

$P \subseteq NP$

As all det. TM can be seen as a non-deter. TM with exactly one choice (or no choice → move to Q_{rej})

eg: ⊙ checking if 2 nos are relatively prime $\in P, \in NP$.

⊙ $0^n1^n \in P, \in NP$

⊙ $ww \in P, \in NP$.

⊙ check if a given no. is prime $\in NP$; $\in P$ (complicated proof)

eg: check if graph is connected

$\langle G \rangle$ encoded as (matrix, adjacency list)

$size = O(n^2)$ n = # vertices

NP class (alternate defn)

A lang. $L \in NP$ if $w \in L$ can be

verified with the help of a certificate (whose size is polynomial to $|w|$) with a polynomial time soln.
to $|w|$

(Q1). does k-Colouring $\in NP$

$L = \{G$ that are 3-colourable$\}$

at present no known poly-time algo to check if a given $G \in L$

But given an assignment of colours to each vertex as certificate we can verify it in $O(m)$ m - # edges time.

(Q2) k-clique

certificate - clique vertices

verify - if all are connected

$L \in NP$, if $\exists$ non-det TM that recognises L in $O(|w|^c)$ i.e. it gives yes answer

in polynomial time. ('no' answer need not be in pol. time)

If 'no' answer can also be verified in polynomial time.

$L \in co\text{-}NP$ or $\bar{L} \in NP$

If $L \in P \Rightarrow L \in NP$ & $L \in co\text{-}NP$.
(decides L)

T.P the 2 defn for NP is same.

(1) We the path of non-det TM 'N' as our certificate. Now since N reaches Qaccept in pol. time, we can verify certificate in $O(|w|^2)$

(2) Guess the certificate & run the verifier algorithm. It guesses a path in the non-det TM. and combining all these path gives the Non-det TM

Travelling salesman problem

IIIrd to finding a least weighted Hamiltonian

cycle → decision version is $\in NP$

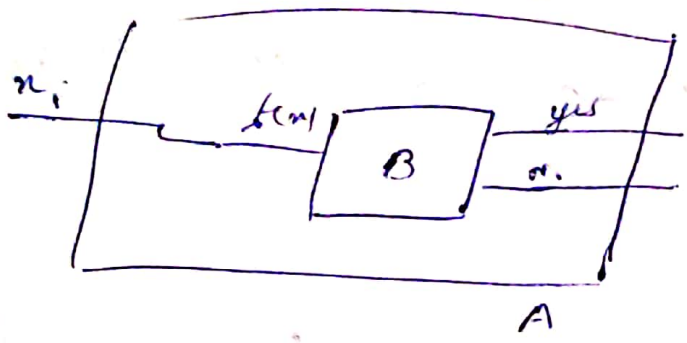
---

if prob A is not in P, then B is also not in P

$\Rightarrow$ B is in P $\Rightarrow$ A is in P

$A \leq_m^P B$

i.e., assume $B \in P$ ($\exists$ an algo in pol. time



$f(x)$ in $|x|^c$

CNF - SAT

It $\in NP$ as given a certificate we can verify it in polynomial time

$\Rightarrow$ if CNF-SAT $\in P$, then $P = NP$?

NP-H class

Problems that are atleast as hard

as any problem in NP

i $\forall X : X \in NP\text{-H}, X \leq_m^p L$ where $L \in NP\text{-H}$.

$\therefore$ if $L \in P$, then $X \in P \Rightarrow NP = P$!

---

(1) clique :- Is there a clique of size k

(2) independent set :- Is there an ind set of size k.

Assume clique $\in NP\text{-H}$. Then does ind.set $\in$ NP-H?

let (1) = L_1 & (2) = L_2

$\forall X : X \in NP, X \leq_m^p L_1$

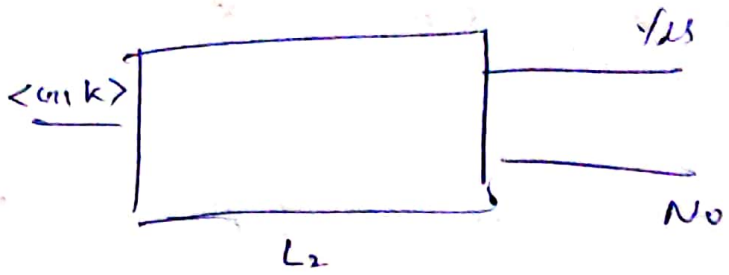
if $L_1 \leq_m^p L_2$ then $\forall X : X \in NP, X \leq L_2$

$\therefore L_2 \in NP\text{-H}$

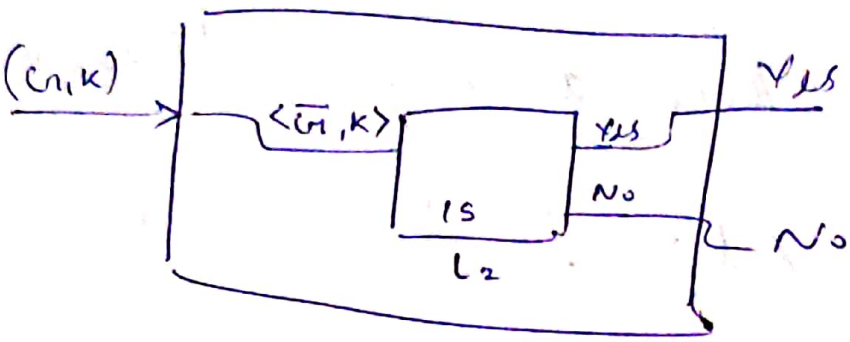
To show $L_1 \leq_m^p L_2$.

$\Rightarrow$ make L_1 using L_2 in polynomial time

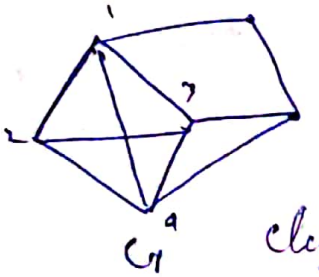
$\therefore$ just prove that if L_2 runs in poly. time, L_1 also runs in pol. time



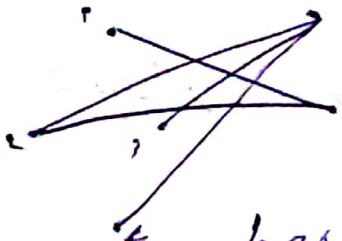
(IS)



clique



$\Rightarrow \bar{G}$ =



G_1 clique of size 4

has ind. set of size 4

∴ If $\bar{G}$ has clique
$\bar{G}$ has ind set

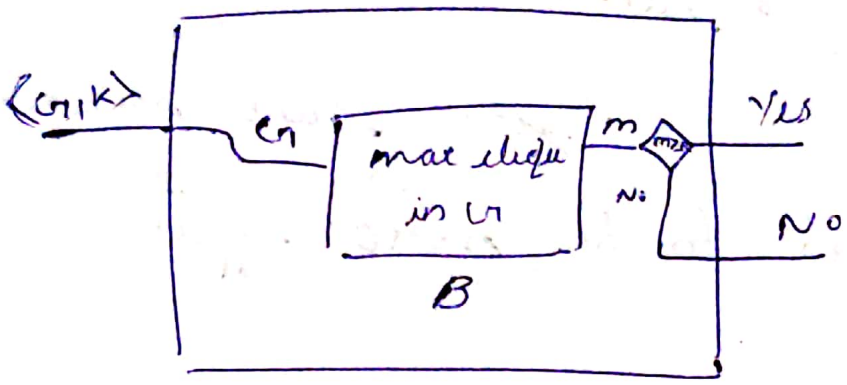
$f(G) = \bar{G}$ can be done in polynomial time

$L_1 \leq_m^p L_2$

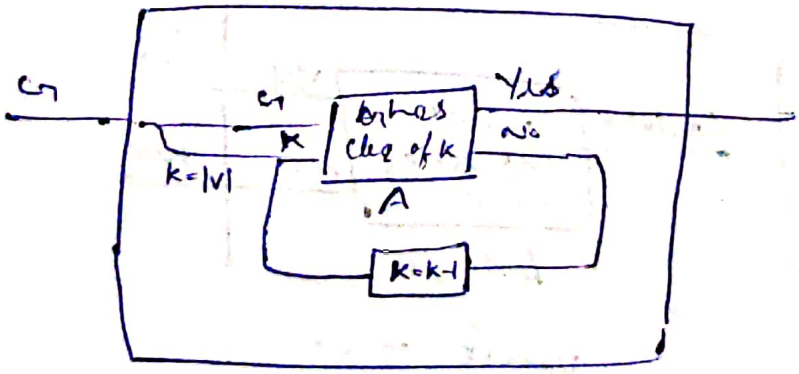
(A) CLIQUE (decision problem)

(B) CLIQUE (optimisation) - what is the max sized clique?

p.t that both are polynomialy equivalent



if B has $O(n^c)$ then A has $O(n^{c'})$



if A has $O(n^c)$ then B has $O(kn^c)$

(C) Find a biggest clique in G.
Remove an edge, check if it has

the same sized clique as before.

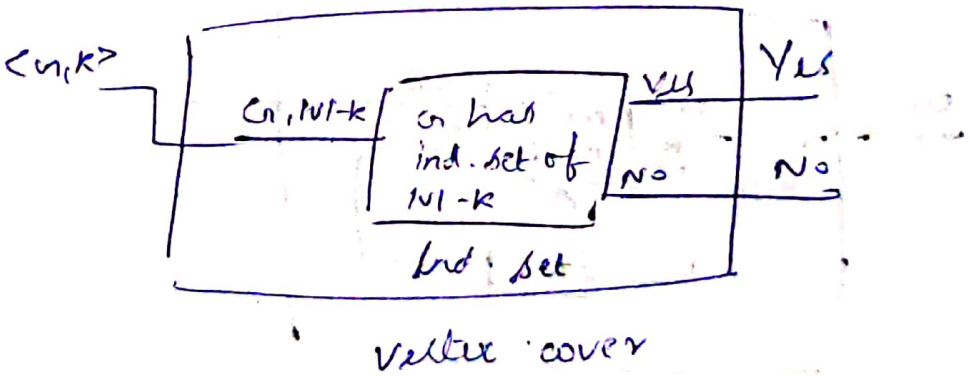
if NO. keep edge & remove another

if YES, remove another

repeat this till only clique edges remain

(or you can remove vertices to be more efficient)

Q. Vertex cover: set of vertices s.t all edges have atleast on vertex in it.



Vertex cover

if G has vertex cover of size K, then G also has ind. set of size |V|-K ?

& vice versa

NP-C

If $A \in NP$ & $A \in NP\text{-}H$, then

$A \in NP\text{-}C$

⊙ Show that clique $\in NP\text{-}H$ if

3-CNF_SAT $\in NP\text{-}H$.

$\Rightarrow \forall X : X \in NP \quad X \leq_m^p$ 3-CNF-SAT

TP 3-CNF_SAT $\leq_m^p$ clique

i.e. $\exists f(F)$ computable in $O(|F|^c)$ s.t

$F \in$ CNF-SAT $\Leftrightarrow f(F) \in$ CLIQUE

how to make $\underbrace{f(F)}_{\langle G, k \rangle}$? (make CNF-SAT using CLIQUE whose I/P is f(F))

for each clause,

C_i : we have 3 vertices in the graph G

let edges be s.t

$\rightarrow$ no edge b/w nodes in same clause

$\leftrightarrow$ edge b/w nodes in diff clauses except if both are compliment of

each other.

$f(F)$ is $O(|F|^2)$

Now to prove (i) $F \in CNF \Rightarrow f(F) = \langle G, k \rangle \in CLIQUE$

$F \in CNF \Rightarrow$ ∄ atleast one literal in each clause which is 1

let $X = \{x_1, x_2 \ldots x_k\}$ be three nodes from each clause. Now no $x_i \in X$ is a complements of another $x_j \in X$ (as all are 1 if F is sat)

$\Rightarrow$ ∄ edge b/w each $x_i \in X$

$\therefore$ ∄ $|X| = k$ (# clause)

$\therefore \langle G, k \rangle \in CLIQUE$

(b) $f(F) \in CLIQUE \Rightarrow F \in CNF\_SAT$

Let $X = \{x_1, x_2 \ldots x_k\}$ be clique each $x_i \in$ diff clause. (no edge b/w nodes in same clause)

Now since no edge exist b/w compliments, all $x_i \in X$ are not compliment of $x_j \in X$

$\therefore$ assign 1 to each literal corresponding nodes in X

$\Rightarrow$ $\exists$ atleast one literal with 1 in each clause

$\Rightarrow$ $F \in$ CNF-SAT (F is satisfiable)

Q S.T 3-SAT is NP-hard if (general) SAT is NP-H

A: SAT $\in$ NP-H $\Rightarrow \forall x: x \in NP, x \leq_m^p$ SAT

TP SAT $\leq_m^p$ 3-SAT

to make $f(F)$ in $O(|F|^c)$

In general sat, let

clause $C_2 = x_2 \vee \bar{x}_3$

$C_3 = x_1 \vee x_2 \vee x_3 \vee x_4$

$x_2 \vee \bar{x}_3 \equiv (x_2 \vee \bar{x}_3 \vee z_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{z}_4)$

$\Rightarrow$ if ϕ = True $\Rightarrow$ ϕ' is also true & vice versa

because $\phi' = (z_4 \vee \phi)(\bar{z}_4 \vee \phi)$

$\phi = T \Rightarrow z_4 \vee \phi = 1$ &

$\bar{z}_4 \vee \phi = 1 \Rightarrow \phi = 1$

III^{rly} $c_1 = x_1 \equiv (x_1 \vee t_1)(x_1 \vee \bar{t}_1)$

III^{r} to c_2 III^{r} to c_2

$\therefore$ make it 3

if $c_4 = (x_1 \vee x_2 \vee \bar{x}_3 \vee x_4)$

$\bar{x}_3 \equiv t_1$

$\equiv (x_1 \vee x_2 \vee t_1) \wedge (\cancel{x_1 \vee x_2} \vee \bar{t}_1 \vee \bar{x}_3 \vee x_4)$

in the sense satisfiable iff always 1

but if t_1 is independent var,

$(x_1 \vee x_2) \vee (x_3 \vee x_4 \vee x_5)$

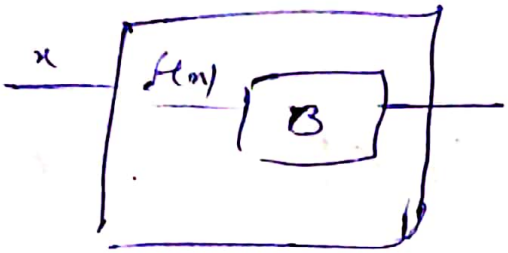
$(x_1 \vee x_2 \vee t_1) \wedge (\bar{t}_1 \vee x_3 \vee t_2) \wedge (\bar{t}_2 \vee x_4 \vee x_5)$

if ϕ is sat, we say ϕ' is also sat by some assignment of values

for k literal clause.

$(x_1 \vee x_2) \vee (x_3) \vee (\ldots) \cdots (x_{k-1} \vee x_k)$

$A \leq_m^p B \Rightarrow$ Algo for A in $O(|x|^a)$ if B runs it in $O(|x|^c)$ time



Q. a) Hamiltonian path: Path that goes through all vertex → directed & undirected

b) Hamiltonian cycle: cycle that covers all vertices

c) Is there a ham. path from u to v $\langle G, u, v \rangle$

use c) to build b).

instead of just adding edge b/w u & v, add a vertex w & 2 edge $\langle v, w \rangle$ & $\langle w, v \rangle$

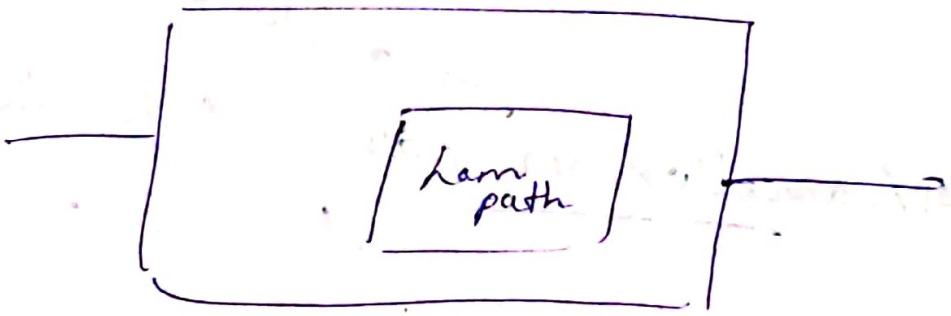
Now, if ∃ a hamiltonian cycle

then ∃ a path from u to v

else

there does not exist path from u to v

hamiltonian cycle $\leq_n^p$ ham path.



if it was known that is there a path from u to v to be in P.

then

remove edge u,v

Is there a path from u to v

if YES G has ham. cycle.

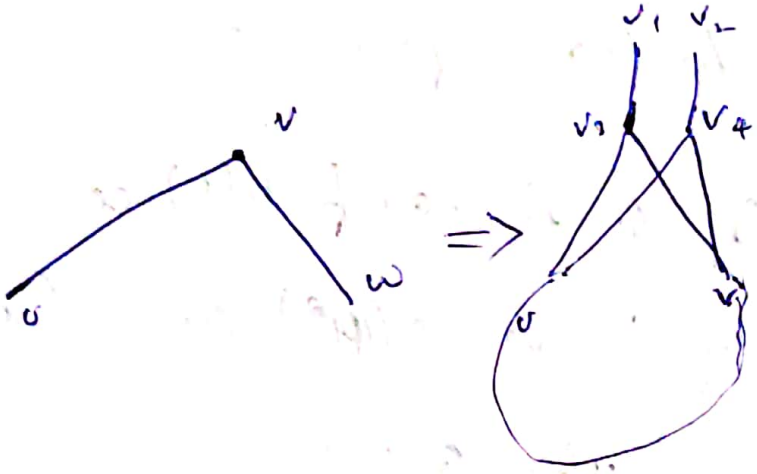
else

pick another edge u,v

Now, what if it was a general hamiltonian path problem?

(can't take complement & give it to ham path eg: if G was complete graph $\bar{G}$ = no edge ⇒ no ham path

$\not\Rightarrow$ G has no ham cycle



eg: Set cover: Given a set $\{e_1, e_2, e_3 \ldots\} = U$

& subsets $S_1, S_2 \ldots \subseteq U$

Given k, is there a k subset whose union is U

(prove using vertex cover algo,

U = set of edges

S_1 S_2 .. are edges incident on vertex 1, 2 .. k

Q P.T. CNF-SAT $\in$ NPC

equivalent to
SAT $\in$ P iff P = NP.

(1) SAT $\in$ NP

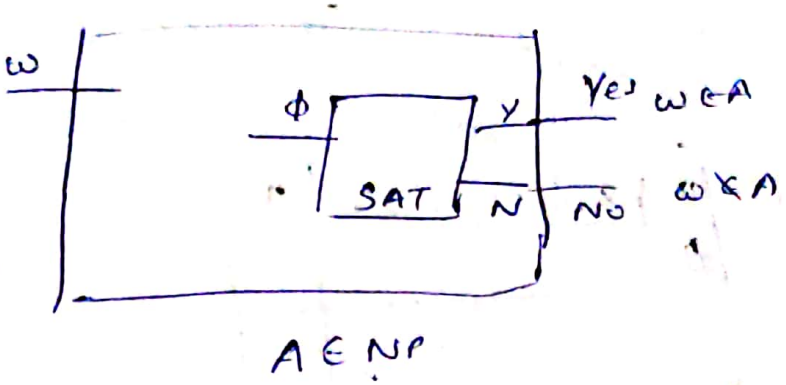
Input: boolean formula.

Yes (if ϕ is satisfiable) can be verified in $O(|\phi|^2)$ time (certificate truth assignments)

If sat has a poly. time algo, then all problems $A \in NP$ also has a polynomial time algo

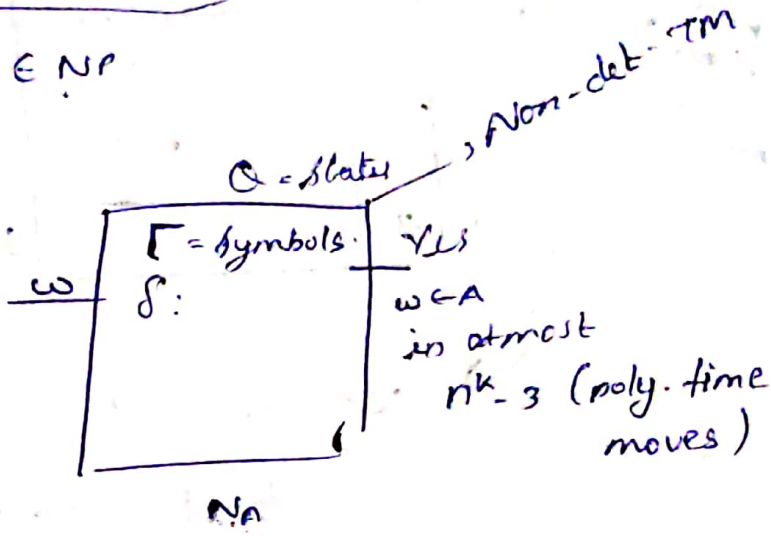
IInd to saying $\forall X : X \in NP$, $X \leq_m^P SAT$

if SAT $\in P$, $X \in P$



$A \in NP$

N_A is a TM s.t.



also N_A answers in n^k steps $\equiv$ if it moves beyond n^k steps $w \notin A$

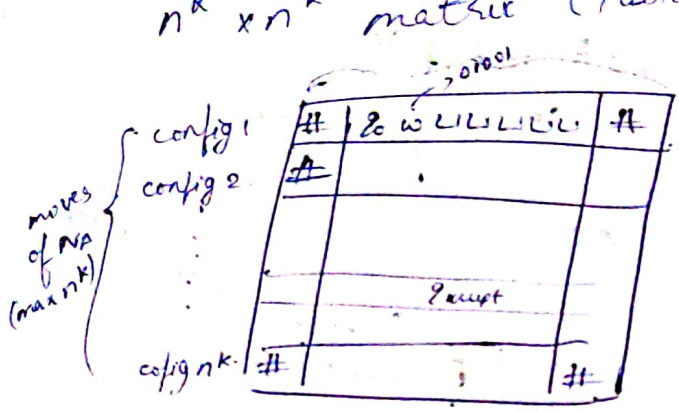
$A \leq_m^P SAT$

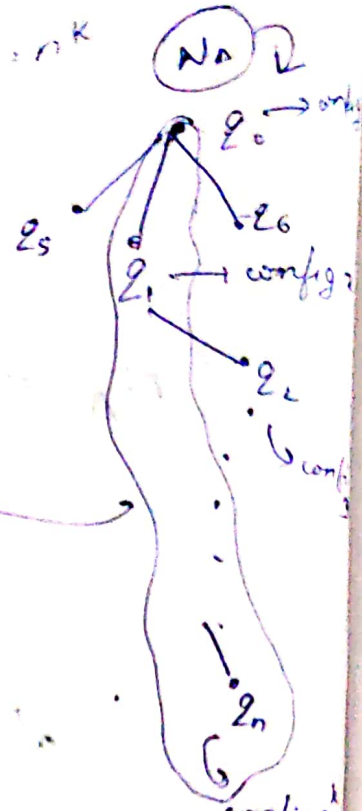
$\Rightarrow w \in A \iff f(w) = \phi \in SAT$

and $f(w) = O(|w|^c)$

N_A ~~has~~ is a non-det TM (we know it exist as $A \in NP$)

$n^k \times n^k$ matrix (Tables)





configuration of a branch ~~that reaches q_{accept}~~

ϕ: should be satisfiable iff

∃ atleast one branch of computation in N_A that reaches q_{accept} (w as I/P)

which means, ∃ atleast one such tablau representing first n^k moves of N_A that eventually goes to q_{accept} (i.e one config has q_{accept} in it)

Now, we have made N_A. to make it ϕ, what are variables & clauses

Possible set of symbols in C.

$$|C| = |\underbrace{\{\#\} \cup \Gamma \cup Q}_{\text{const. number}}| = m$$

In normal case $\underbrace{x_{i,j}}_{\text{var.}} = s \Rightarrow$ matrix has symbol s at (i,j)

but here variable can take only boolean values.

make a bool. var. that says matrix has s at(i,j)

$\Rightarrow x_{i,j,s} = True \Rightarrow$ Matrix has s at i,j
(s: symbol)

$\therefore$ No. of variables $= n^k \times n^k \times \frac{m}{|C|}$

$= poly.$ in n $\quad O(n^{2k})$

$\therefore x_{i,j,0} = True$

$\left(\Rightarrow x_{i,j,\#} = False \quad x_{i,j,\Gamma} = F \quad x_{i,j,q} = F \quad x_{i,j,q_0} = F\right)$ } one cell has only 1 symbol in it

ϕ will capture :

(1) It represents such a Tableau $\rightarrow n^k \times n^k$

ϕ_{cell} symbols from C.

$$X_{i,j,s} = True \iff X_{i,j,t} = False \quad \forall t \neq s$$

(2) It begins with config $q_0 w \sqcup\sqcup\sqcup\ldots$

ϕ_{start} (initial config with w as input

(3) Row i to Row(i+1) is a valid move

ϕ_{move} is δ of N_A

(4) It goes to q^*_{accept} within n^k rows

ϕ_{accept} (n^k config)

$$\Rightarrow \phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

---

imp { From w we construct ϕ. And ϕ is satisfiable iff N_A has atleast 1 branch that accepts w in atmost n^k moves

ϕ_{cell} : i [S]

• cell(i,j) has only 1 symbol in it

$x_{i,j,s} = true \iff x_{i,j,t} = false \quad \forall t \neq s$

$a \iff b$

$\rightarrow (\neg a \Rightarrow b) \wedge (\neg b \Rightarrow a)$

ϕ_{accept} : $x_{i,j}, q_{accept}$ = true for some i,j

$\Rightarrow \phi_{accept} \bigvee_{\substack{i=1 \text{ to } n^k \\ j=1 \text{ to } n^k}} x_{i,j,q_{accept}}$

ϕ_{start} : $x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \bigcirc x_{1,3,1} \ldots x_{1,n^k,\#}$ $\quad \underset{\in \omega}{}$

ϕ_{cell} : $(x_{i,j,s} \wedge \overline{x_{ij s_1}}_{s_1 \neq s} \wedge \overline{x_{ij s_2}}_{s_2 \neq s} \ldots\ldots) \vee$

$(\overline{x_{ijs}} \wedge x_{ij s_1} \wedge \overline{x_{ij s_2}}_{s_2 \neq s_1} \ldots) \vee$

$\Rightarrow \phi_{cell} = T$ iff ~~to~~ can have only 1 value

$\equiv \left(X_{ijs} = \text{True for some } s \right) \wedge \left(X_{ijs} \text{ and } X_{ijt} \text{ are both not true together if } s \neq t \right)$

for fixed pair s, t

$\Rightarrow$ [a cell has atleast 1 symbol s in it and 2 symbols cannot appear together in a cell]

$$\equiv \underbrace{\left(\bigvee_{s \in c} X_{ijs} \right) \wedge \left(\bigwedge_{\substack{s,t \in c \\ s \neq t}} \overline{X_{ijs} \wedge X_{ijt}} \right)}_{\text{for one cell}}$$

for all the cell,

$$\phi_{cell} = \bigwedge_{\substack{i=1 \text{ to } n^k \\ j=1 \text{ to } n^k}} \left(\left(\bigvee_{s \in c} X_{ijs} \right) \wedge \left(\bigwedge_{\substack{s,t \in c \\ s \neq t}} \overline{X_{ijs} \wedge X_{ij}} \right) \right.$$

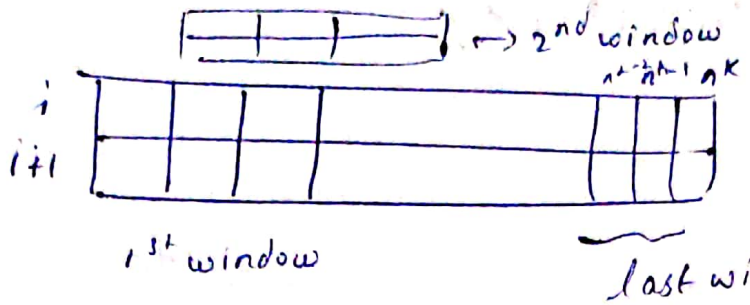
ϕ_{move}: what are the legal moves in N_A

we can check the $i+1^{th}$ row follows

how i by some legal move in $\mathcal{S}$

just by looking at all

2×3 windows in 2 rows



for fixed i

Claim: if we can ensure all such 2×3 windows are legal

i for $i = 1$ to $n^k - 1$

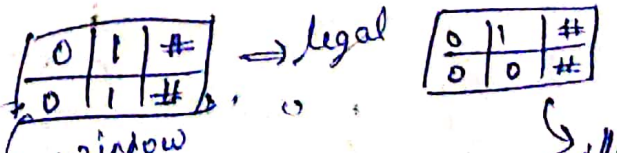
$j = 1$ to $n^k - 2$

then way i to i+1 is a legal move

Proof: config can be $\# q_0 \times 1 1 \sqcup \sqcup \sqcup \#$ or

$\# 0 0 q_1 \times 0 1 \sqcup \sqcup \#$

At each move we can only move 1 pos left or right

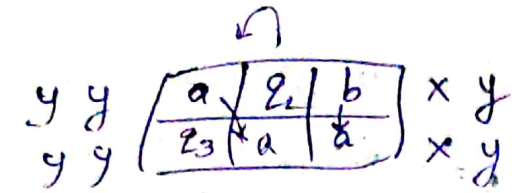


it can have $q_1, q_2 \ldots$

window

only 1st can be changed

eg. $\delta(q_2, b) = (q_3, a, L)$



case

(when stage as a middle symbo

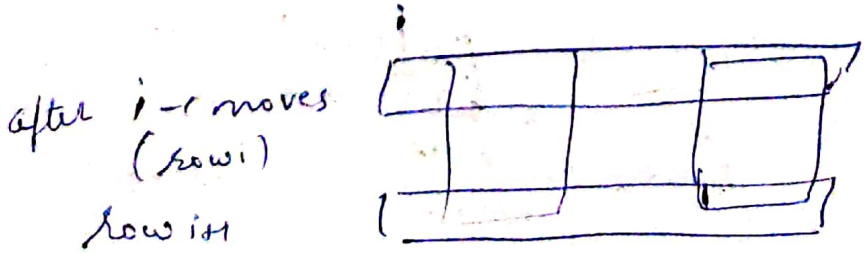
For a fixed 2×3 window, we can check if its legal or not in

time: O(# moves with state q) = co

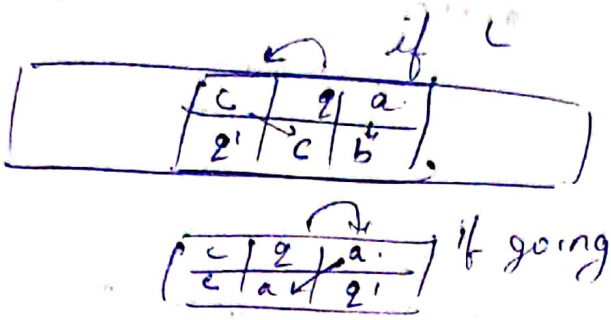
(if q is center symbol in top row)

In rest of the cases, we can directly say if its legal or not

claim: If all windows are legal, then for each row i & i+1, row i+1 follows legally from row i



we just need to focus on the window with state q in top center cell



} computed in const time

for the rest, just make sure only 1st is changing (if so)

if this window is legal, row $i+1$ follow s

This means the n^k rows corresponds to $n^k - 1$ moves (max $n^k - 3$ moves in worst case

$n^k - 3$ moves if all δ moves to right

$\phi_{move} \equiv$ (for every 2×3 window)(window (i,j) is legal)

$= \bigwedge_{\substack{i=1 \text{ for } n^k-1 \\ j=1\ n^k-2}}$ window (i,j) is legal.

depends on δ:

In general.

$O(n^{2k})$ ϕ_{cell} = (atleast 1 symbol in i,j) $\wedge$ (no 2 symbol in 1 cell)

$O(n^{2k})$ ϕ_{move} = $\bigwedge_{\substack{i=1 \text{ to } n^k-1 \\ j=1 \text{ to } n^k-2}}$ window ij is legal

constant

$O(n^k)$ ϕ_{start} = first row is starting config of TM N_A with w as input

Let $w = w_1 w_2 \ldots w_n$ (n symbols)

ϕ_{start} = $(x_{11\#}) \wedge (x_{12} q_0) \wedge (x_{13} w_1) \wedge \ldots$

$(x_{1n^k\#})$

$O(n^{2k})$ ϕ_{accept} = $\bigvee_{\substack{i=1 \text{ to } n^k \\ j=1 \text{ to } n^k}}$ $x_{ij} q_{accept}$