Sudeep Sir

Not solvable using a computer

Eg: Given a mathematical statement, check if its true or false.

A problem can be solved using a computer ⇒ it can be solved using a turing machine.

Turing Machine - A mathematical model of a computer.

↳ we assume that it has infinite memory.

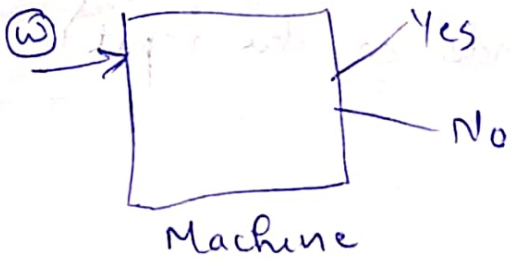
// What are different models of machine?

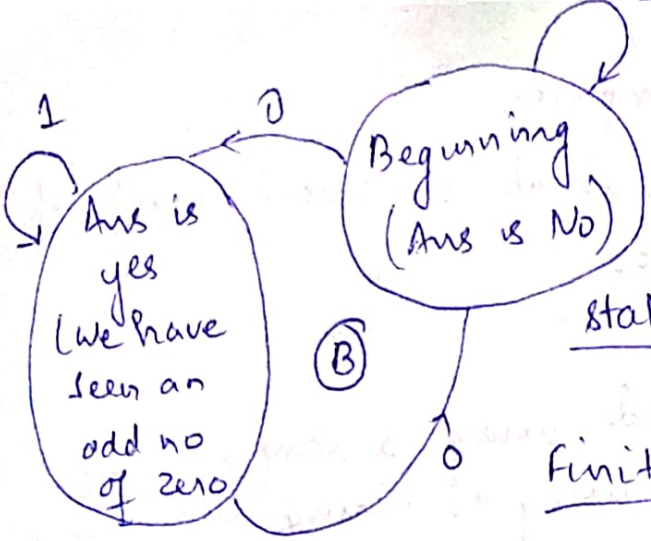
simple machine (calculator) < complex machine (Turing Machine)

Automata Theory → deals with different types of mathematical model.

Input: 0001101 Q) Does the string have an odd No of zeroes.

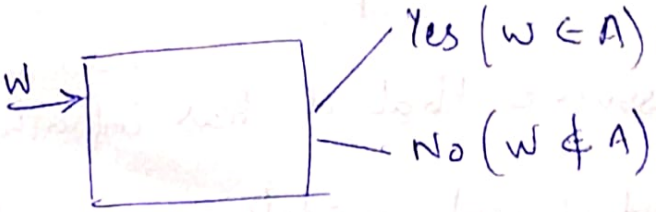
Language:- A consists of binary string with odd no of zeroes





Finite Automata

// Machine B "decides" language A.



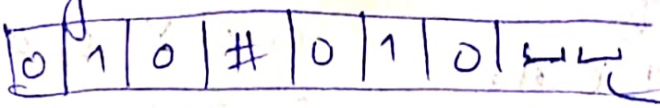
Given a string (containing 0, 1, or #). check if it is of the form w # w (same string repeated)

$A = \{ 0\#0,\ 1\#1,\ 010\#010,\ 0010\#0010 \}$

$A = \{ 0^n 1^n : n \geq 0 \}$

// We need to remember "stuff" that may be long (how long, depends on the input)

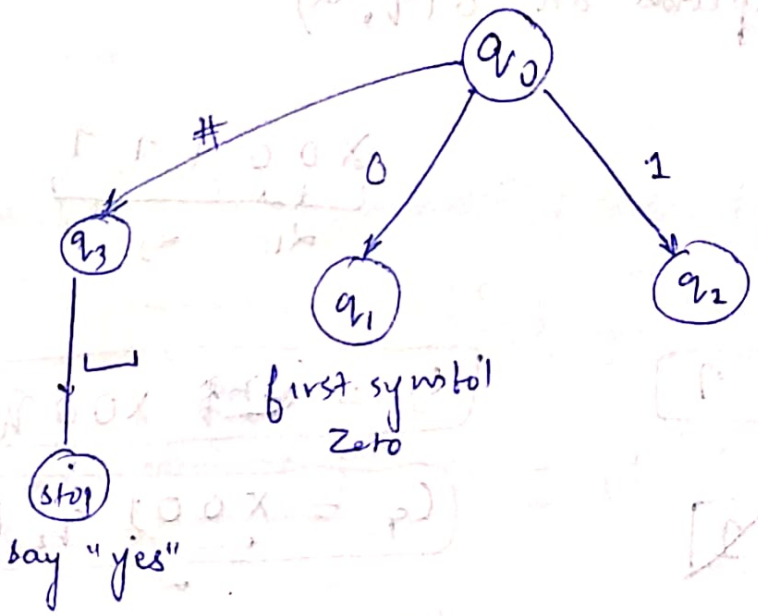
// A turing machine has an "infinite tape"



(initially contain input)

- initially contain input, followed by 'blank' symbol.

Tape Speed :- initially at the first symbol.

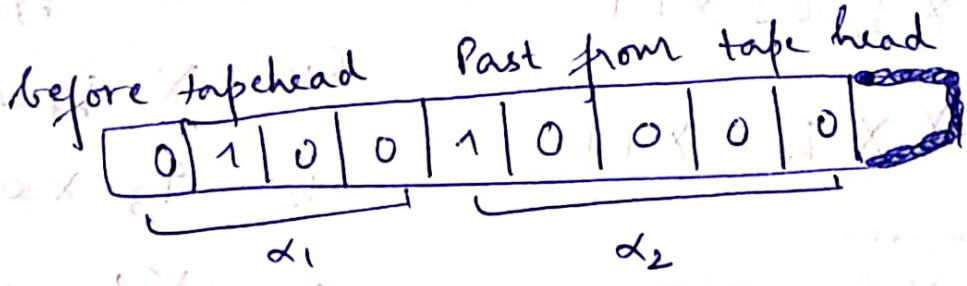


// After #, match the symbol with first symbol.

(Notes incomplete)

" Configuration of a turing Machine :

- what is the content of tape.
- where head points to
- what is the current state.



current state = q

$w = 000111 \in A$.

001X0

$C =$ | 0 1 0 0 | q | 1 0 0 |

α_1 α_2

$C_i = \alpha_1 q \alpha_2$ is the "current configuration".

Next config, C_{i+1} = depends on $\delta(q, a)$

$C_2 = \alpha_1 q_1 \alpha_2$

$= X0\ q_1\ 0111$

$C \Rightarrow X00111$

$C_4 = X00q_1 11$

$C_{14} = X0q_4 111$

$C_{15} = X00q_4 111$

$C_{16} = X0q_5 0Y11$

$C_{17} = Xq_5 00Y11$

$C_{18} = \$q_5 X00Y11$

$C_{19} = Xq_6 00Y11$

$C_{20} = XXq_4 0Y11$

after few moves

$C_0 = q_0 000111$

$C_1 = X q_1 0 0111$

$C_2 =$

(first symbol of α_2 is 'a')

X00111

α_1 α_2

$C_3 = X00q_1 111$

$C_4 = X001 q_2 11$

α_1 α_2

$C_5 = X0011q_2 1$

$C_6 = X00111q_2$

$C_7 = X001q_3 1$

C_8 X001q3 11

C_9 X00q3 111

C_{10} X0q3 0 111

C_{11} Xq3 00 111

C_{12} q3 X00 111

C_{13} Xq4 0011

X X X Y q_7 Y Y

X X X Y Y q_7 Y

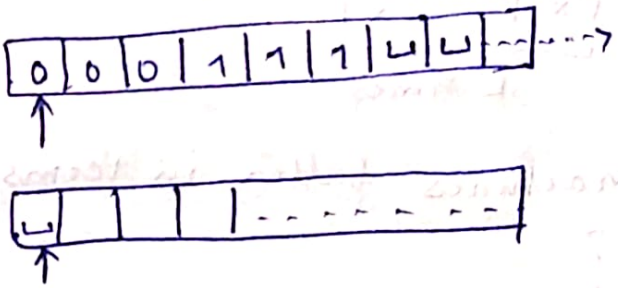
X X X Y Y Y q_7

→ X X X Y Y Y ␣ q_{accept} Turing Machine steps

Design a turing machine ① w # w

② ~~0^N~~ w(0^{2^N})

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



Multitape turing machine.

(k-tape for a constant k).

① Scan through tape 1, copy zeroes into second tape if there is a 0 after first 1, reject.

② Cross off the zeroes on tape 2 & corresponding 1s on tape 1. If both finish together (cut the same time), accept else reject.

// Tape 1 contains input w initially, followed by blanks.

// Tape 2 contains all blanks

// Set of states Q, {q_{accept}, q_{reject}, q_0 (q_{start})}, Σ

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

↑ current state — k current symbols

where $D_i \in \{L, R\}$ & $b_i \in \Gamma$ $\quad i \in 1 \ldots k$

$a_i \in \Gamma$

M:

$$\delta : Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R\}^k$$

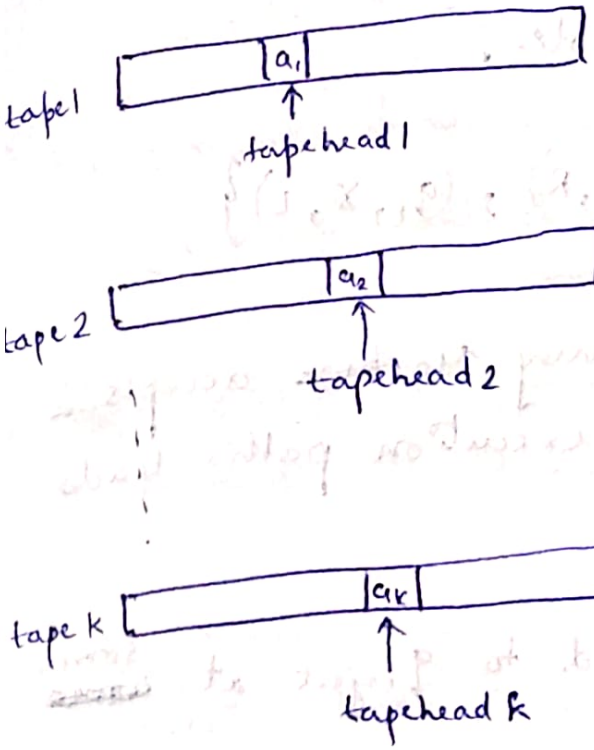
$(a_1, a_2 \ldots\ldots a_k) \in \underbrace{\Gamma \times \Gamma \ldots\ldots \times \Gamma}_{k \text{ times}}$

Q) Are Multiple turing machines better in terms of computational power ?

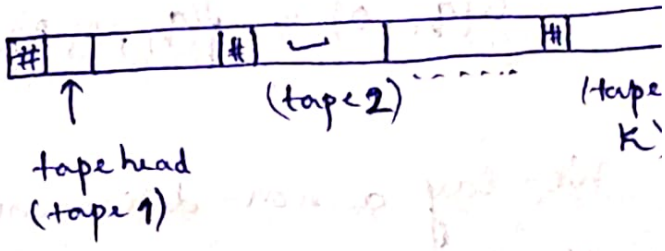
A) No, anything that can be solved with k tapes (for a fixed k) can be solved with a single tape turing machine.

Proof: If a language A is decidable with a k tape tape turing machine, then A is decidable (recognisable) with a single tape turing machine.

k - tape turing machine



Single tape turing Machine



To identify the current symbol, we use a "dotted" symbol, $\{\dot{a}, \dot{1}, \dot{x}, \dot{y}, \dot{\sqcup}\}$

⇒ Scan the tape once, to identify current symbols - $a_1, a_2, \ldots, a_k$. (till k^{th} dotted symbol)

$$\delta(q_1, a_1, a_2 \ldots, a_k) = (r, b_1, b_2 \ldots, b_k, D_1, D_2 \ldots, D_k)$$

⇒ Making changes in all k tapes is done in another scan.

↳ increase in tape size or some tape i involves shifti the entire symbols towards right by one positio

Non - Deterministic Turing Machine

Multiple choices of moves available.

$$\delta(q_1, 0) = \{(q_1, 0, R), (q_2, X, R), (q_1, X, L)\}$$

We say a non-deterministic Turing Machine accepts a string w if any one of the execution paths leads to q_{accept}.

N rejects w if all paths lead to q_{reject} at some point.

The 2 way tape can be wrapped around to make a 2-track tape, that can carry 2 symbols on one cell.

earlier set of tape symbols, Γ

Now $\Gamma' = \Gamma \times \Gamma$

$\langle q, 0 \rangle \Rightarrow$ right half $\quad\quad$ $\langle q, 1 \rangle \Rightarrow$ left half

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

$$\delta(q_1, a_1) = (r, b_1, R)$$

$$\delta'\left(\langle q, 0 \rangle, \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}\right) = \left(\langle r, 0 \rangle, \begin{pmatrix} b_1 \\ a_2 \end{pmatrix}, R\right)$$

↳ On original machine, we are on the left half, direction is left. $\delta(q_1, a_2) = (r, b_2, L)$

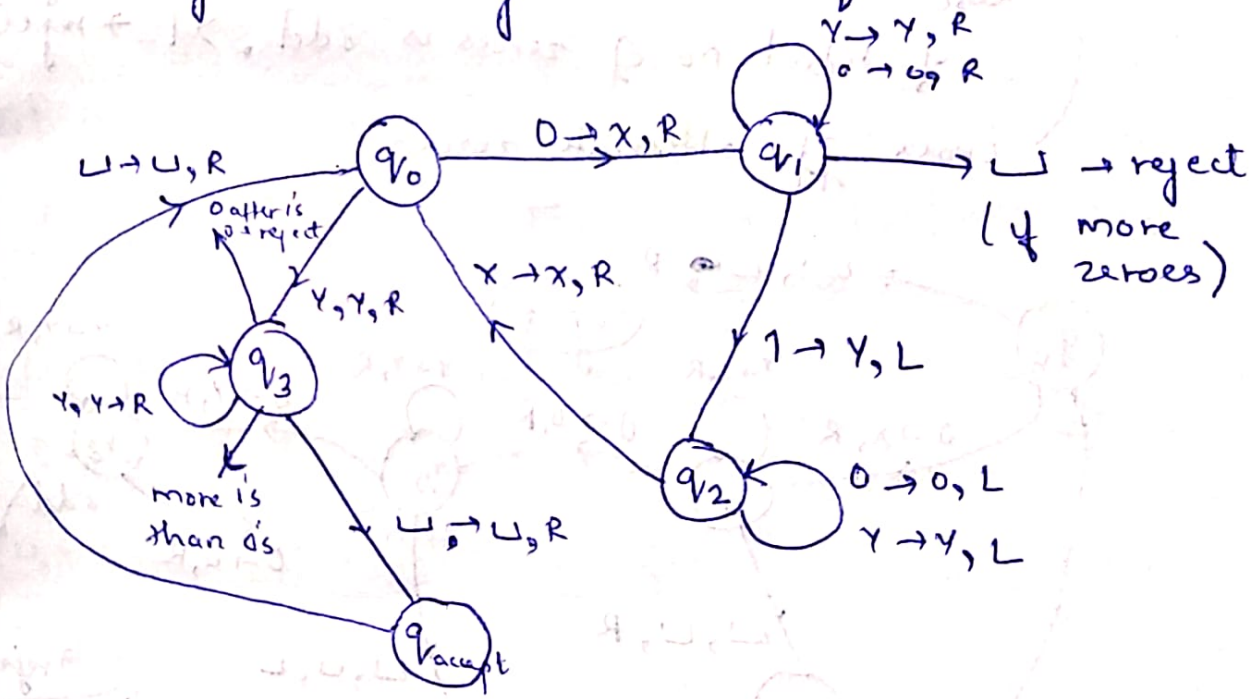
↳ Original move was in right half, not on the leftmost cell, direction left.

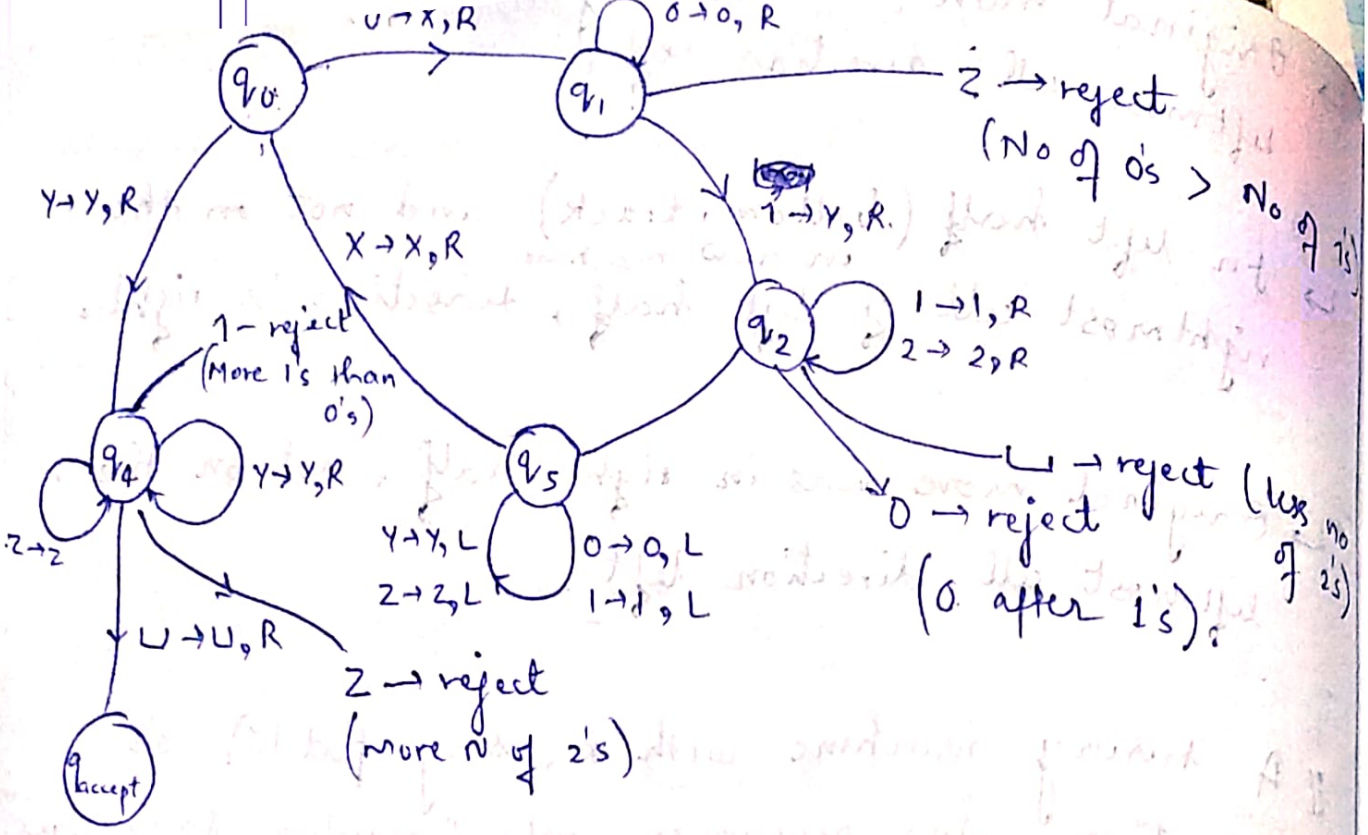
↳ In left half (bottom track) and not on the rightmost cell of left half, direction is right. (in new machine)

↳ Original move was in right half, not on the leftmost cell, direction left.

// A turing machine with R, stay put (S) as directions. This recognizes only "regular languages" Give a proof.

// Design a turing machine for $0^N 1^N 2^N$?





~~0 0 0 0 1 1 2 2 2~~ ⇒ ~~X X X X Y Y Y~~

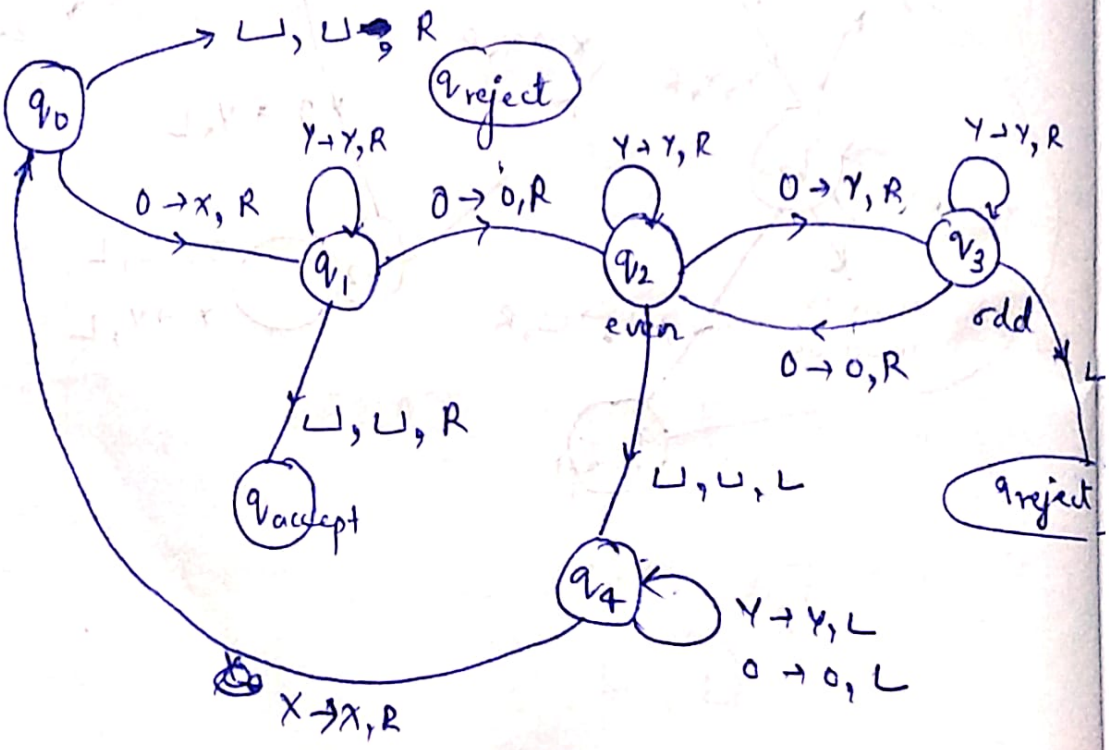
2) 0^{2^N}

If total no of 0's is 1 → accept

else

If total no of zeroes is odd, >1 → reject

cross off alternate zeroes



Non-Deterministic Turing Machine

If N is a non-deterministic turing machine that recognizes the same language A, then there is an equivalent deterministic turing machine M that recognizes the same language A.

$$\delta(q, a) = \{(r_1, b_1, D_1)$$
$$(r_2, b_2, D_2)$$
$$\vdots$$
$$(r_k, b_k, D_k)\}$$

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

set of all subsets of $Q \times \Gamma \times \{L, R\}$.

$W = 0\,1\,0$

$=$ | 0 | 1 | 0 | ⊔ | ⊔ |

Let

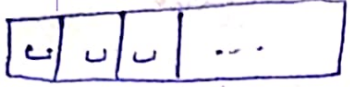
$$\delta(q_0, 0) = \{(q_0, X, R), (q_1, 0, R), (q_4, 0, R)\}.$$

Let 'b' be the maximum no of choices for any move in δ. Max size of $\delta(q, a)$ or $\max|\delta(q, a)|$

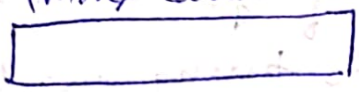
$\rightarrow L, R$ Maximum value for b is $|Q| \times |\Gamma| \times 2$

(State) (symbol) (direcho

M_1, deterministic 3 tape turing Machine

Tape 1:  // input tape
(never changed)

(moves carried out)

Tape 2:  // (execution / simulation type)

Tape 3:  // current branch of execution tape or Address tape.

// Tape 1 contains input (never altered) followed by blank symbols

// Tape 2 works as execution tape ; (one branch of execution at a time)

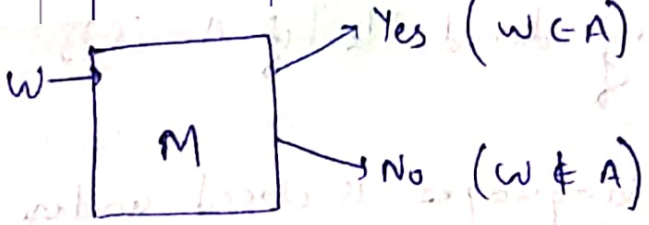
// Tape 3 current branch of execution $(i_1, i_2, \ldots, i_k)$

Based on contents of tape 3, the moves are executed one after the other or tape 2 (if it goes to q_{accept}) M also goes to q_{accept}, q_{reject}, halt.

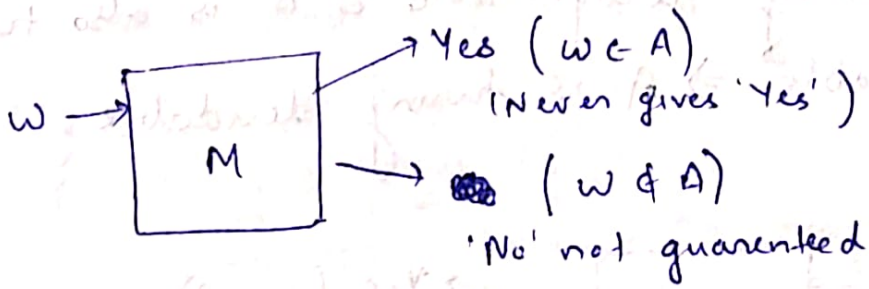
Update the "current branch" on type 3, continue.

Undecidability :-

Any finite language is turing decidable, and hence (recursive)
turing recognizable also
(recursive enumerable)



A is decidable



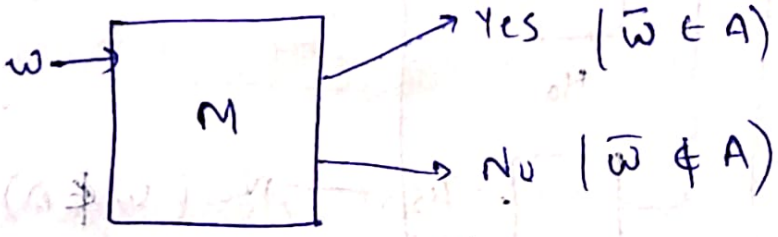
A is recognisable

If there is a turing machine, that recognizes A & halts on all inputs, ~~A is recognizable~~ then A is turing decidable.

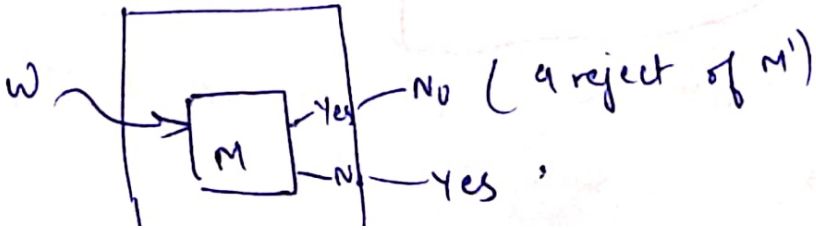
If A is turing decidable, what can we say about $\bar{A}$?

↳ $\bar{A}$ is the complement of A. If $w \in A \Rightarrow w \notin \bar{A}$

$w \notin A \Rightarrow w \in \bar{A}$ over same alphabet.

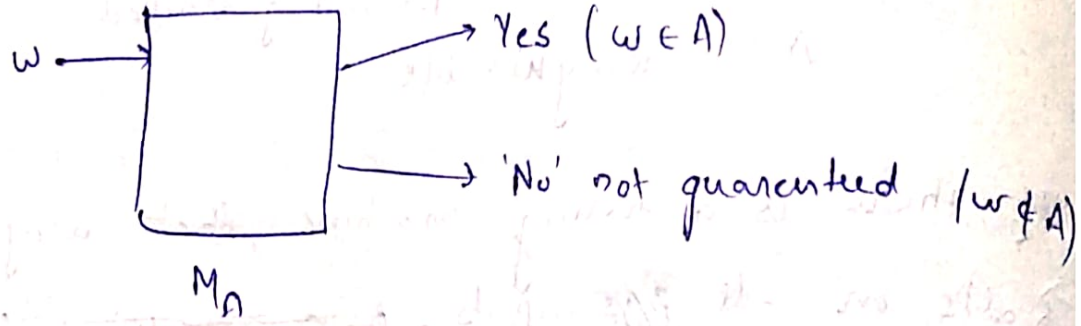


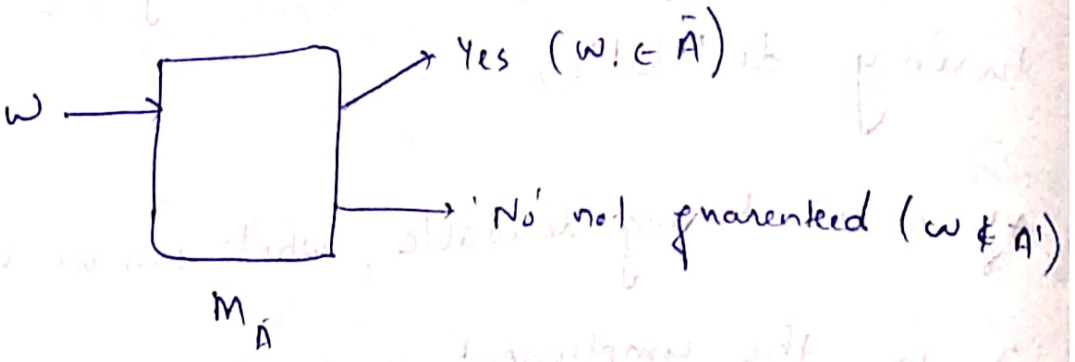
⇒

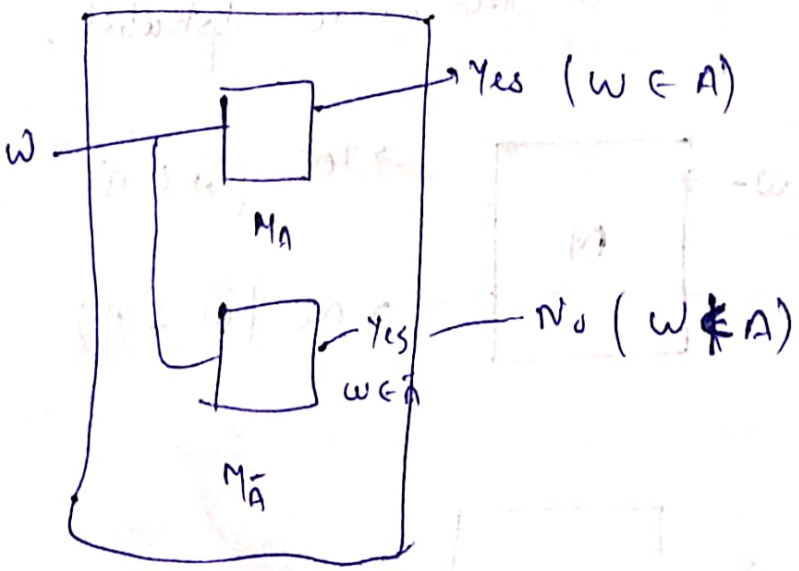


The set of recursive languages is closed under complement-tion.

A is turing recognizable & $\bar{A}$ is also turing recognizable $\Rightarrow$ A is turing decidable.

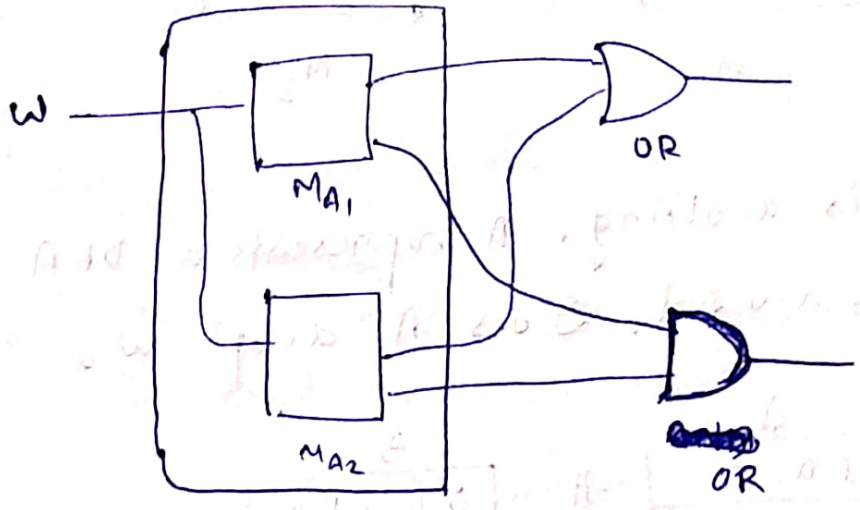




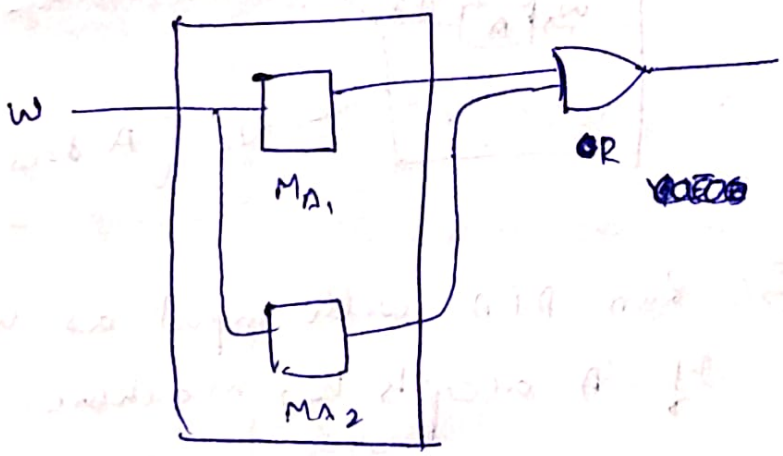


Union :- A_1 is decidable, A_2 is decidable what about $A_1 \cup A_2$?

$w \in A_1 \cup A_2 \Rightarrow w \in A_1$ or $w \in A_2$.



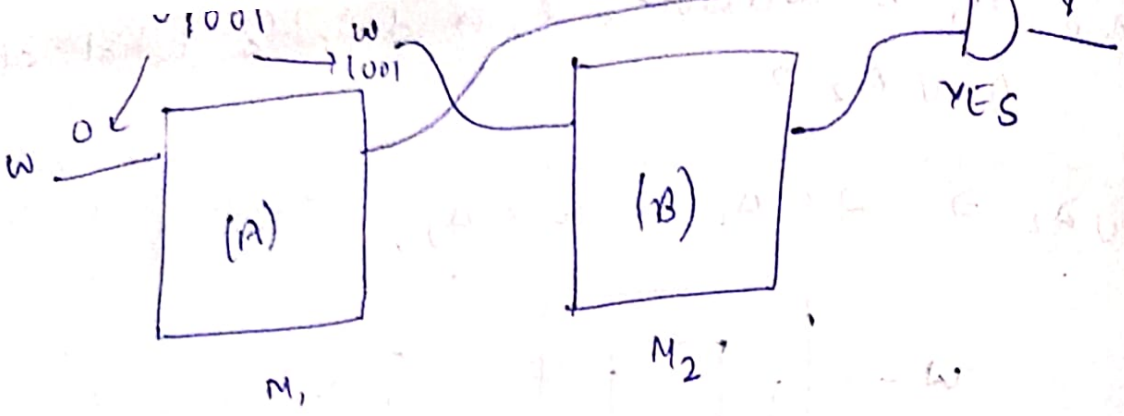
A_1 is turing recognisable
A_2 is turing recognizable
} $A_1 \cup A_2$ = recognizable.



Turing recognizable languages are also closed under intersection.

Concatenation :- $A_1 \cdot A_2$ $w_1 \cdot w_2 \in A_1 \cdot A_2$

$w_1 \in A_1$ & $w_2 \in A_2$.

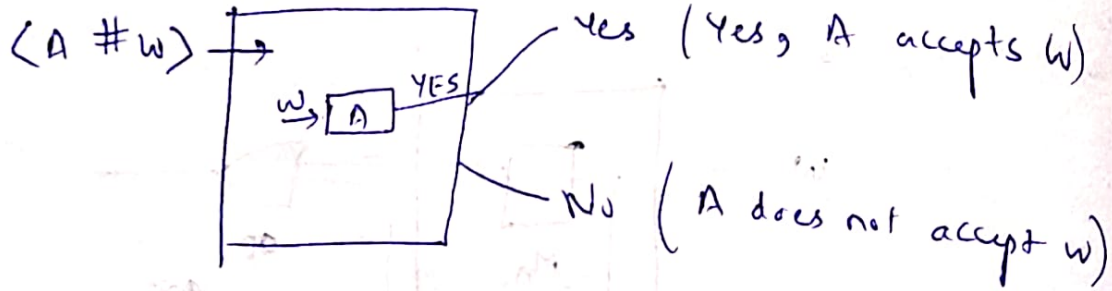


$\langle A, w \rangle$ is a string, A represents a DFA & w is a string over 0 & 1. Does A accept w.

$L_{DFA} = \{ \langle A, w \rangle : A \text{ accepts } w \}$.

⇓

Turing Decidable.



Algorithm :- Run DFA with input as w.

If A accepts w, machine should say Yes.

$L_{TM} = \langle M, w \rangle : M$ accepts w) this Language is turing decidable.

$L_{TM} = \{\langle M, w\rangle = M \text{ accepts } w\}$

A similar algorithm does not work for L_{TM} as M is not guaranteed to halt.

Theorem :- L_{TM} is undecidable

First step :- We "construct" (or define) a language in such a way that there is no turing machine that recognizes that language. (Using 'Diagonalization' technique).

Idea :- Each number represents a single turing machine.

(11) Number each input string (over {0,1}).

$w_1 = 0$
$w_2 = 1$
] length = 0

$w_3 = 0\,0$
$w_4 = 0\,1$
$w_5 = 1\,0$
$w_6 = 1\,1$
] length = 1

Numbering a Turing Machine :-

↳ enough to identify all moves ("encode" all moves in δ).

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

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

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

⇒ First order the states : q_1 (start state) q_2 (accept sta

$q_3, q_4 \ldots\ldots q_n$

Rename the states.

$q_0 \to q_1$ $\quad q_{accept} \to q_2$ $\quad q_1 \to q_3$ $\quad q_3 \to q_4$

(New moves after Renami

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

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

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

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

Number the tap alphabets also : $X_1, X_2 \ldots\ldots, X_k$

$\{0, 1, \sqcup\}$ $\quad \{L, R\}$

$X_1 \; X_2 \; X_3$ $\quad D_1 \; D_2$

$\delta(q_1, X_3) = (q_3, X_3, D_2)$ move

$\delta(q_3, X_2) = (q_4, X_2, L)$ move 2

$\delta(q_4, X_1) = (q_2, X_1, R)$ move 3

$\delta(q_3, X_2) = (q_3, X_2, R)$ move 4

If we have the 5 numbers i, j, k, l, m : then it corresponds to move

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

$$\Downarrow$$

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m.$$

$\langle$move 1$\rangle$ 11 $\langle$move 2$\rangle$ 11 $\langle$move 3$\rangle$ 11 ... ... $\langle$move m$\rangle$

$$\Downarrow$$

gives a number for the turing machine.

From a given number, we can get all moves of the turing machine. There is exactly one TM, or no TM for a given number.

Defining a language such that there is no TM that recognizes that language.

// Mi accepts w_j if we have entry in the M-table as 1.

L_d (diagonal) = { w_i s.t entry $\langle i,i \rangle$ is 0 in this matrix }

such that L_d is not recognized by a M_i.

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

show that L_d is not recognized by any M_j.

↳ Assume M_j recognizes the language L_d.

What about the entry $\langle i,j \rangle$?

From the definition of L_d, if $w_j \in L_d$, M_j does not accept w_j. But by our assumption, M_j recognizes $L_d \Rightarrow M_j$ accepts $w_j \Rightarrow$ contradiction.

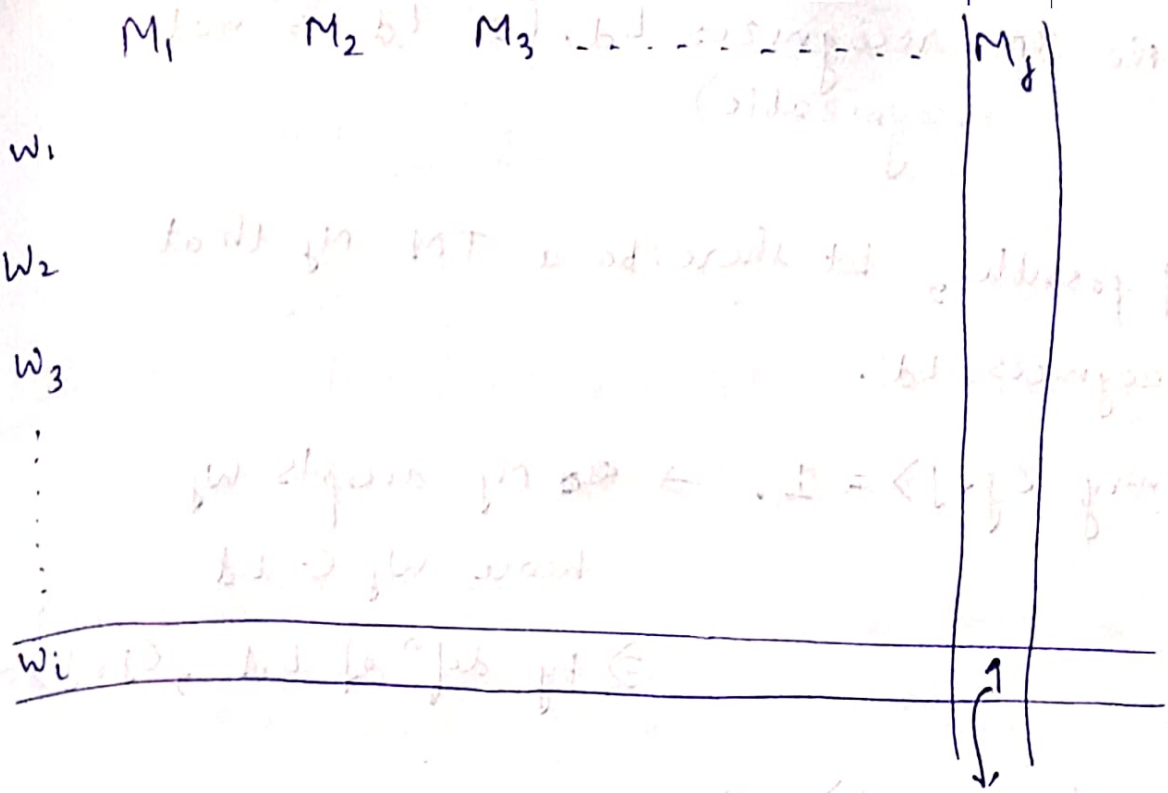
Case 2 :- if $w_j \neq L_d$. M_j recognizes L_d only.

it accepts strings of L_d, it does not accepts w_j (because $w_j \neq L_d$).

$\Rightarrow w_j \in L_d$ (by def of L_d) Contradiction

// Case 1 : $\langle j,j \rangle = 0$.

// case 2 : $\langle j,j \rangle = 1$.



M_j is a valid TM, and it accepts w_i.

Number all turing Machines

(Same TM may be assosiated with different numbers but is OK).

For a given number, there is exactly one or zero TM.

Number strings (one number $\longleftrightarrow$ one string)

Think of the matrix with strings (in order) as rows TMs (in order) as columns.

$$\text{entry}(i,j) = \begin{cases} 1 & M_j \text{ accepts } w_i \\ 0 & \text{otherwise} \end{cases}$$

Define $L_d = \{ w_i : \text{entry} \langle i,i \rangle = 0 \text{ or } w_i = M_i \text{ does not accept } w_i \}$

Claim : No TM recognizes L_d. (or L_d is not recognizable)

Proof :- If possible, let there be a TM M_j that recognizes L_d.

Entry $\langle j, j \rangle = 1$. $\Rightarrow$ M_j accepts w_j

hence $w_j \in L_d$

$\Rightarrow$ by def^n of L_d, $\langle j, j \rangle$.

If entry $\langle j, j \rangle = 0$

$\Rightarrow$ $w_j \notin L_d$.

$\Rightarrow$ M_j does not accept w_j

but by our assumption M_j recognizes L_d

$\Rightarrow$ M_j accepts w_j

$\rightarrow$ $\langle j, j \rangle = 1$ contradiction.

Given a string w, k bits & value is n. What is i such that $w = w_i$?

check $\left(2 + 2^2 + 2^3 + \ldots + 2^{k-1}\right) + (n+1)$

$2\left(2^{k-1} - 1\right) + n + 1$

$2^k + n - 1$ ✱

$$k = \lfloor \log_2 (i+1) \rfloor$$

$$n = i - 2^k + 1.$$

Theorem:- The language $TM = \{ \langle M, w \rangle = M$ accepts $w \rangle$ is undecidable.

Proof: If L_{TM} is decidable (if we have a TM, M_{TM} that decides L_{TM}, then L_d is also decidable.