UNIT-5

Turing Machine

* Limitations of previous machines :-

FA: Regular lang : Limited memory & no comparisons

PDA : CFL : FA + 1 stack

⇒ Turing machine (TM) : FA + n stacks

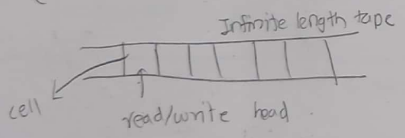
DTM ≡ NTM (Power)

accepts recursive enumerable lang (REL)

⇒ Turing machine has a tape of infinite length

Tape is divided into no of cells.

(FA/PDA can move only in 1 direction)



In TM, we can move in left & right i.e both direction

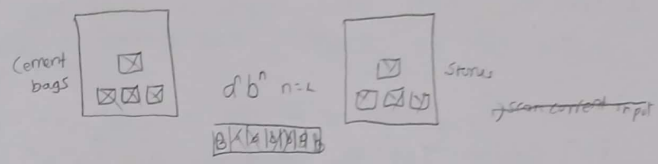
*Functions of read/write head :-

① Read the current input symbol.

② Write / update the current input symbol.

③ Move left / right.

* How TM works?



* Formal defⁿ of TMs :-

A TM consists of 7 tuples

$(Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q : finite set of states

Σ : ~~input~~ finite set of input symbols

$\Gamma = B \cup \Sigma \cup$ {step alphabets} : finite set of tape alphabets

q_0 : initial state

B : blank symbol, $B \in \Gamma$

F : set of final set, $F \subseteq Q$

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times (L, R)$

Ex: $\delta(q_0, a) \rightarrow (q_1, X, R)$ → transition func.

$q_0 \xrightarrow{(a, X, R)} q_1$ → state diagram

▷ 16 Mar ⇒ Thursday

Non-deterministic eg]

Ex] $\delta(q_0, a) \rightarrow (q_1, X, R)$

$\rightarrow (q_2, Y, L)$

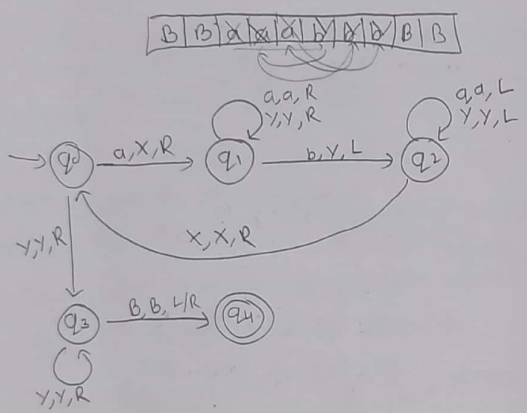
NDTM

+ Turing machine is more accurate model of general computer.

+ TM can do everything that a real computer does.

+

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



→ Instantaneous description of TM

$w = aab$

$q_0aab \vdash Xq_1ab$
$\vdash Xaq_1b$
$\vdash Xq_2aY$
$\vdash q_2XaY$
$\vdash Xq_0aY$
$\vdash XXq_1Y$
$\vdash XXYq_1B$

q_1, B has no transition

If TM halts in final state, then TM accepts the lang.

If TM halts in non-final state, TM rejects the lang

Ex) $w = ab$

$q_0ab \vdash Xq_1b$
$\vdash q_2XY$
$\vdash Xq_0Y$
$\vdash XYq_3B$
$\vdash XYBq_4$

∴ Halts in final state

∴ w is accepted by TM.

Ex) $w = abb$

$q_0abb \vdash Xq_1bb$
$\vdash q_2XYb$
$\vdash Xq_0Yb$
$\vdash XYq_3b$

∴ Halts in non-final state.

∴ w is not accepted by TM.

→ Transition table

	a	b	X	Y	B
q_0	q_1X,R	—	—	q_3,Y,R	—
q_1	q_1a,R	q_2Y,L	—	q_1,Y,R	—
q_2	q_2a,L	—	q_0X,R	q_2,Y,L	—
q_3	—	—	—	q_3,Y,R	q_4,B,R
q_4	—	—	—	—	—

→ Transition function

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

$\delta(q_0,Y) \rightarrow (q_3,Y,R)$

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

$\delta(q_1,Y) \rightarrow (q_1,Y,R)$

$\delta(q_1,b) \rightarrow (q_2,Y,L)$

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

$\delta(q_2,Y) \rightarrow (q_2,Y,L)$

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

$\delta(q_2,Y) \rightarrow (q_3,Y,R)$

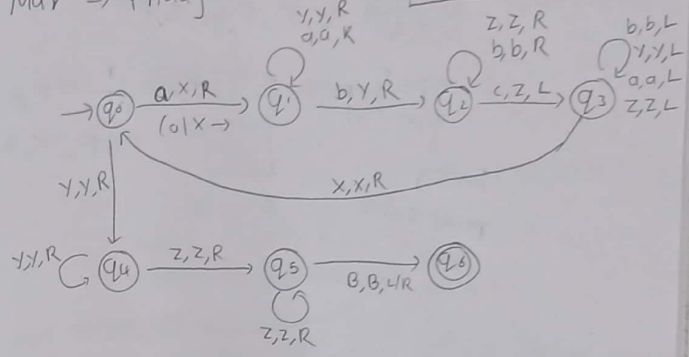
$\delta(q_3,B) \rightarrow (q_4,B,R/L)$

Ex] Design TM for $L=\{a^n b^n c^n \mid n>1\}$

> 17 Mar ⇒ Friday

B	a	a	a	b	b	b	c	c	c	B

⇒



→ $w=ab$

$q_0ab \vdash aq_1b$

$\vdash abq_2B$

∴ Halts in q_2.

∴ NOT ACCEPTED

* Church Turing thesis:-

① Anything that can be done on any existing computer can also be done by TM.

② No one has yet been able to suggest a problem that is solved by an algorithm, for which TM cannot be designed.

③ Alternative model has been proposed for mechanical computation but none of them is more powerful than TM.

TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$L(M) = \{ w \in \Sigma^* \mid q_0 w \vdash^* \alpha p \beta$, where $p \in F, \alpha, \beta \in \Gamma^* \}$

↓

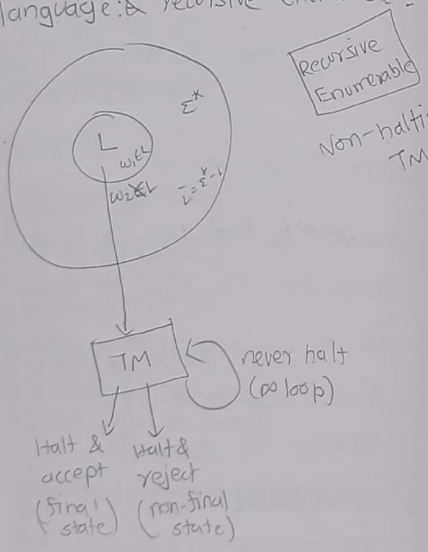
Turing recognizable lang.

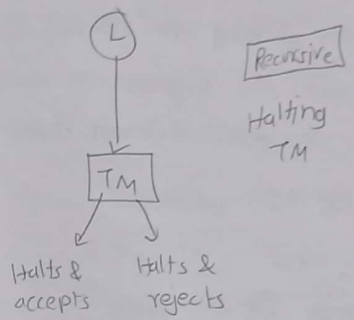
→ Recursive lang

↓

Recursive enumerable lang.

⇒ Recursive language & recursive enumerable.





→ If L is recursive, then, $\bar{L}$ is also recursive lang.

→ If L & $\bar{L}$ are recursive enumerable, then L is recursive lang.

→ If L is RE-R, then complement of L, $\bar{L} \notin RL$.

▷ Mar 23 ⇒ Thursday

Recursive language → Total TM / Halting TM

- accept: Halts at accepting state
- reject: Halts at non-accepting state

Recursive enumerable → Non Halting

- accept: (Halts at accepting state)
- reject: (Halts at non-final state)
- loop: infinite loop

When we call a lang non-REL?

↓

For this, we cannot design TM

* Variants of TM

→ Standard TM

① Tape of infinite length

② r/w head can move left/right

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

→ Turing machine with stay option -

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

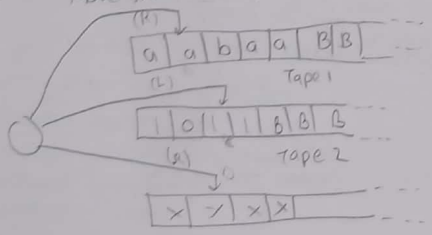
Power of this TM is same as standard TM.

→ TM with semi-infinite length -

Same power as standard TM

→ Multitape TM :-

More than one tapes



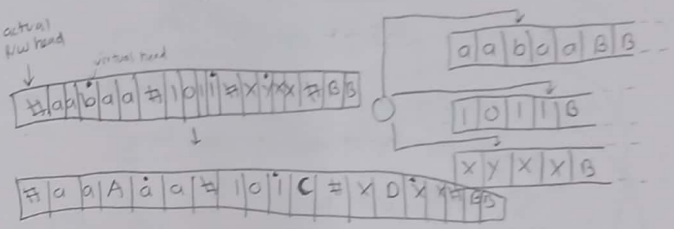
$\delta(q_0, a1Y) \longrightarrow (q_1, A\underline{CD}, RLR)$

Γ^* Γ^*

$\delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times (L,R)^k$

Power is same as standard TM, but it is more efficient than standard TM.

• Theorem - Every multitape TM has an equivalent single tape TM



* Non-deterministic TM (NTM) :-

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

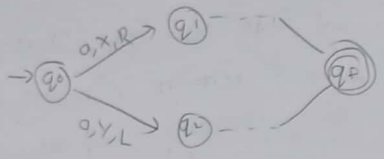
Non-deterministic $\delta: Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times (L,R)}$

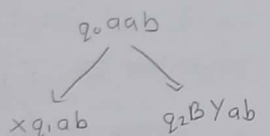
> Mar 24 ⇒ Friday

(Ex) $L = \{a^n b^n \mid n \geq 1\}$ — STM → $O(n^2)$ moves; multitape TM → $O(n)$ moves

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

eg. $(q_0, a) \longrightarrow \{(q_1, X, L), (q_2, Y, R) \cdots\}$





If a single branch of NTM reaches accepting state, the NTM accepts the lang.

* Theorem :-

Every NTM has an equivalent det. TM.

Proof - Given an NTM 'N', show how to construct equivalent DTM 'D'.

if N accepts on any branch, then D will accept.

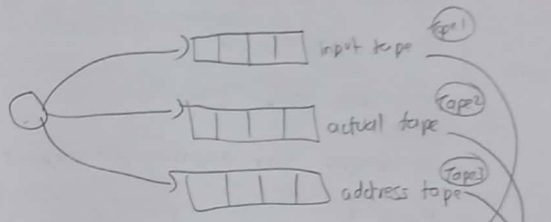
if N halts on any branch without accepting, then D will halt & rejects.

Approach: Simulate all the branches of computation tree of N.

Search for any way N can accept.

On DTM, we use BFS to explore all the branches of the computation tree.

DTM 'D' will finally get to accepting state



Input tape will contain the string.

Input tape will never be modified

This is the actual tape of DTM.

It is used to control the BFS, tells which choice we have to make during the computation

→ Algorithm

Initially, tape 1 will contain the input

tape 2 & tape 3 are empty.

Step

- Copy tape 1 to tape 2.
- Use tape 2 as actual tape

- when choices (non-deterministic branch points) occur, consult tape 3 as tape 3 contains path. Each number tells which choice to make.
- Run the simulation all the way down the branch, till computation dies.
- Try for next branch
- Increment the address on tape 3.
- Repeat

▷ Mar 27 ⇒ Monday

* Enumerator → prints all the strings generated by a lang. (output tape)
  → TM with an attached printer
  → Has two tapes
    ① Working tape
    ② Output tape

a a a b bb

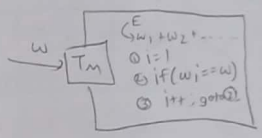
|abab|#|aaa|bbb|#

L(TM): Lang. accepted by TM
L(E): Lang. generated by enumerator

Lang. accepted by TM → Turing recognizable lang (REL)

* Theorem: A lang is Turing recognizable iff some enumerator enumerates it.

Proof: If enumerator E generates the lang A, then you can design TM M, that accepts A.



* Decidable and undecidable problems.-

There must be an algorithm → ∴ Halting TM exists.

No TM exists → Non-halting TM exists } semi-decidable

~~▷ Mar 28 ⇒ Tuesday~~

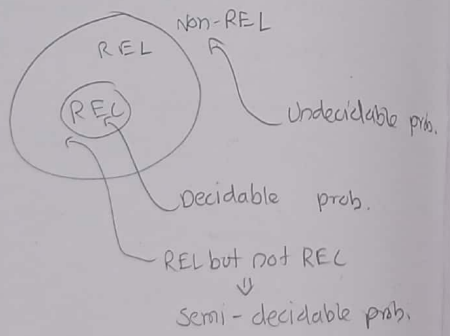
UNIT- 6

Any problem can be converted into a lang.

* Recursive lang (REC): A lang L is called recursive if ∃ a TM that accepts all the strings in L & rejects all the strings not in L

* Recursive enumerable lang. (REL): A lang L is called REL if ∃ a TM which will accept all the strings in L, but the TM may or may not halt for input strings which are not in L. → Non-halting TM

* Any decidable problem can be converted to REC.

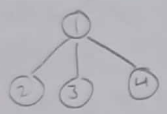


Generally, we call a problem undecidable if no halting TM exists

	REG	DCFL	CFL	CSL	REC	REL
Membership prob. $w \in \Sigma^*$	✓	✓	✓	✓	✓	✗
Infinite prob L = infinite	✓	✓	✓	✗	✗	✗
Emptiness prob $L = \emptyset$	✓	✓	✓	✗	✗	✗
Equality prob. $L_1 = L_2$	✓	✓	✗	✗	✗	✗
Completeness prob $L = \Sigma^*$	✓	✓	✗	✗	✗	✗
$L_1 \cap L_2 = \emptyset$	✓	✗	✗	✗	✗	✗
IF $L_1 \subseteq L_2$ (Subset problem)	✓	✗	✗	✗	✗	✗
L is ambiguous	✓	✓	✗	✗	✗	✗

* TM can solve any problem.
Any problem can be converted to a lang
Any instance of the problem can be treated as string.
If string is in the lang., TM will accept (halt at final state)
If string is not in lang, TM will reject (halt at non-final state)

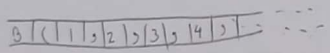
* Is this undirected graph connected?



$G(V,E)$

$\langle G \rangle = \{(1,2,3,4), ((1,2),(1,3),(1,4))\}$

$\Sigma = \{(,), ,, 1,2,3,4\}$



Algorithm -

Select a node & mark it.

For each node N, if N is unmarked & there is an edge from N to already marked node then mark N until no more nodes can be marked.

> Mar 29 ⇒ Wed

* Binary encoding

DFA $(Q, \Sigma, q_0, F, \delta)$

$Q: \{q_0, q_1, \ldots q_{n-1}\}$ Encode q_i as 0^{i+1} string

$q_0 \to 0$

$q_1 \to 00$

$\Sigma: \{a_1, a_2, \ldots a_n\}$ Encode a_i with 0^i string

$q_2 \to 00$

$\delta(q_j, a_i) \to q_k$ q_j a_i q_k

$0^{j+1}10^i10^{k+1}$

Eg. 01001000

q_0 a_2 q_2

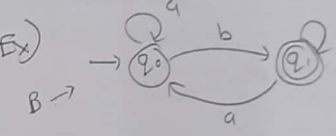
$\delta(q_0, a_2) \to q_2$

F can be encoded as a list of final states

q_2 & q_3 are final states

00010000

Ex)



$a \to 0$

$b \to 00$

$\delta(q_0, a) \to q_0 \Rightarrow 01010$

$\delta(q_0, b) \to q_1 \Rightarrow 0100100$

$\delta(q_1, a) \to q_0 \Rightarrow 001010$

$\delta(q_1, b) \to q_1 \Rightarrow 00100100$

Encode DFA as binary string

for differentiating final state

$\langle B \rangle \to$ 01010 1 0100100 1 001010 1 00100100 11 00

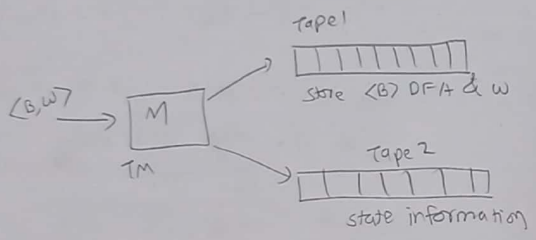
transition ① ② ③ ④ Final state

B is the DFA, $\langle B \rangle$ → binary encoding of B

Ex) $w = aba$

$\langle w \rangle = 010010$

* Theorem:- $A_{DFA} = \{\langle B, w \rangle\}$ is a decidable lang



* Binary encoding of CFLs:-

$Q \{q_1, q_2, q_3, \ldots q_n\}$

Encode 0^j

$q_1 = 0$

$q_2 = 00$

Assume tape symbol (r)

$X_1, X_2, X_3, \ldots, X_n$

Encode X_j with 0^j

We shall refer direction L as D_1 & direction R as D_2

$D_1 \rightarrow L \rightarrow 0$

$D_2 \rightarrow R \rightarrow 00$

$\delta(q_i, X_j) \rightarrow (q_j, X_L, D_M)$

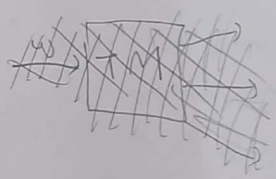
q_i, X_j, q_j, X_L, D_M

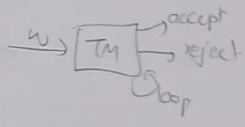
$0^i 1 0^j \; 1 0^k 1 0^l 1 0^m$

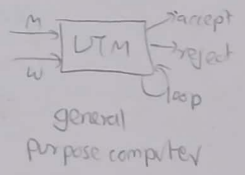
1 transition of TM

Eg. 00100100010000100

$q_2 \; X_2 \; q_3 \; X_4 \; R$







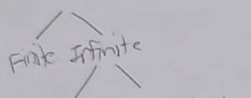
general purpose computer

Mar 30 ⇒ Thursday

* Mathematical concepts:-

Bijection func.: A func is called bijection if it is one-one & onto.

Sets



Countable :- A set S is countable if its elements can be put in one on one correspondence with set of natural nos

Uncountable :- (opposite)

Ex] $E = \{0, 2, 4, 6, 8, \ldots\}$ Countable or uncountable?

Infinite set

$\Rightarrow N = \{1, 2, 3, 4, \ldots\}$

$\therefore$ E is countable

Ex] $R = \{1, 1.01, 1.001, \ldots, 2\}$ → uncountable

Ex] $I = \{0, 1, -1, 2, -2, \ldots\}$ → countable

* In TOC, $\Sigma = \{a, b\}$

Σ^* = universal lang

Σ^* countable or uncountable?

2^{Σ^*} = {set of ~~pos~~ all possible subsets of Σ^*}

each subset represents a lang

$\therefore$ set of all possible languages over Σ^*

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

$2^{\Sigma^*} = \{L_1, L_2, L_3, L_4 \ldots\}$

Set of all TMs is a countable set

S → countable set

$S_1 \subseteq S$ every subset of countable set is countable

Every TM can be encoded using binary strings

$\Sigma = \{0, 1\}$

$\Sigma^* = \{$ set of all possible strings on 0 & 1 $\}$

M → set of all TMs

$\therefore M \subseteq \Sigma^*$

$\therefore$ Set of all TMs is countable.

$\Rightarrow 2^{\Sigma^*}$ → uncountable set

Proof :- $\Sigma^* = \epsilon, a, b, aa, ab, ba \ldots$

(contradiction)

Assume 2^{Σ^*} is countable

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

$L_1 = 1\ 0\ 0\ 0\ 0\ 0 \ldots$

$L_2 = 0\ 1\ 1\ 0\ 0\ 0 \ldots$

$L_3 = 1\ 1\ 0\ 1\ 0\ 0 \ldots$

$L_4 = 0\ 0\ 1\ 0\ 1\ 1 \ldots$

$L_5 = 0\ 0\ 0\ 1\ 0\ 1 \ldots$

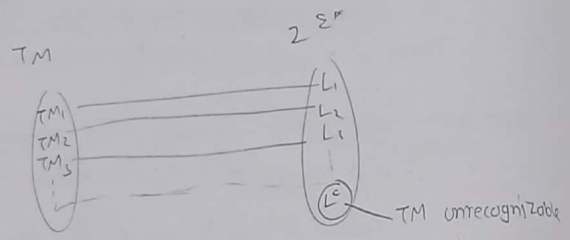
$L_6 = 1\ 0\ 0\ 0\ 0\ 0 \ldots$

$L_{Diagonal} = 1\ 1\ 0\ 0\ 0$

$\overline{L_{Diagonal}} = 0\ 0\ 1\ 1\ 1$ → not part of our consideration

∴ contradiction

2^{Σ^*} is an uncountable set.

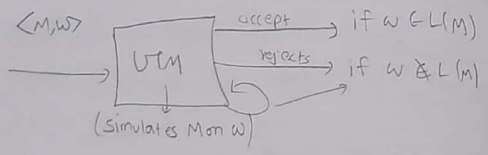


Mar 31 ⇒ Friday

* Acceptance problem in TM :-

$A_{TM} = \{ \langle M, H \rangle \mid M \text{ is a TM \& } M \text{ accepts } w \}$

It is an undecidable problem.



Contradiction

Proof - We assume ATM is decidable

Suppose 'H' (a UTM) decides A_{TM}.

$$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ rejects } w \end{cases}$$

We now construct a new TM 'D'

D = " On input <M> " where M is a TM

Run H on input (M, <M>)

if H accepts, reject
if H rejects, accept

$$D(\langle M \rangle) = \begin{cases} \text{accept if } M \text{ doesn't} \\ \text{accept } \langle M \rangle \\ \text{reject if } M \text{ accepts } \langle M \rangle \end{cases}$$

Now D runs on itself

$$D(\langle D \rangle) = \begin{cases} \text{accepts if } D \text{ rejects } \langle D \rangle \\ \text{rejects if } D \text{ accepts } \langle D \rangle \end{cases}$$

H is telling D rejects, but D is telling itself accept.

H is telling D accepts, but D is telling itself reject.

∴ Contradiction

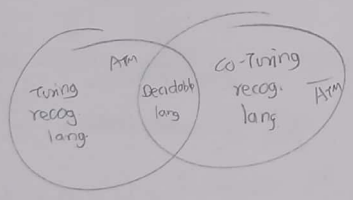
∴ A_{TM} is undecidable

* Turing Recognizable lang
  - Rec. lang (REC)
  - Rec. enum. lang. (REL)

- REC is closed under intersection & complementation & union.
- REL is closed under union, intersection but not on complementation.
- Complement of Turing Recognizable lang is known as co-turing recognizable lang.

* A_{TM} is an undecidable problem.
  $\overline{A_{TM}}$ is not Turing recognizable.

Proof :- Assume $\overline{A_{TM}}$ is Turing recognizable
A_{TM} is Turing recognizable lang.
From these two, A_{TM} is decidable lang.
But, we know that, A_{TM} is undecidable.
∴ Contradiction
∴ $\overline{A_{TM}}$ is not Turing recognizable.



* Reducibility :-

A reduction is a way of converting one problem to another problem, So the solution of the second problem is used to solve the 1st problem.

Problem A is reduced to problem B. $\Rightarrow$ $(A \leq B)$ ~~$(A \leq B)$~~

Ex] Find area of rectangle ≤ Find length & width of the rectangle

If B is decidable, then A is decidable
If A is undecidable, then B is undecidable

~~if problem~~ if B is undecidable, then A is undecidable.
~~if A is undecidable~~, then B is decidable

~~Apr~~ Apr 3 ⇒ Monday

* Problem A is reduced to problem B. ~~$(A \leq B)$~~
  $A \leqslant B$

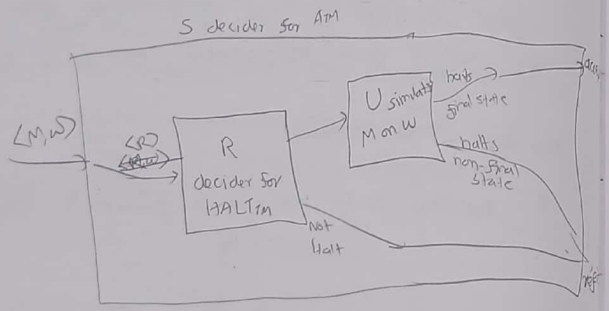
① if problem B is decidable, then A is decidable
② if problem A is undecidable, then B is also undecidable.

Decider : A TM is called decider if it halts at final state on input string w_1 & halts at non final state on w_2 (no loop)

$HALT_{TM} = \{ \langle M,w \rangle \mid M \text{ is a TM, halts on } w \}$ is undecidable problem

Proof ; A_{TM} reduces to $HALT_{TM}$

Assume $HALT_{TM}$ is decidable. Let there be a decider R for $HALT_{TM}$. We use R to construct a decider S for A_{TM}.



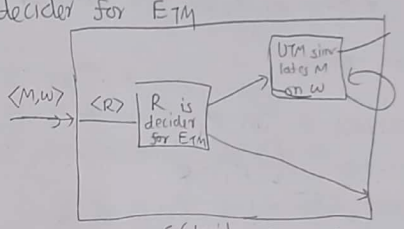
Already we know that A_{TM} is undecidable, so we cannot construct decider S for A_{TM}.

∴ Contradiction

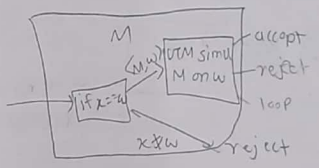
Q $E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM \& } L(M) = \phi \}$ is undecidable

⇒ A_{TM} reduces to E_{TM}

Suppose decider R exists that chooses whether lang of a TM M is empty or not. R is decider for E_{TM}

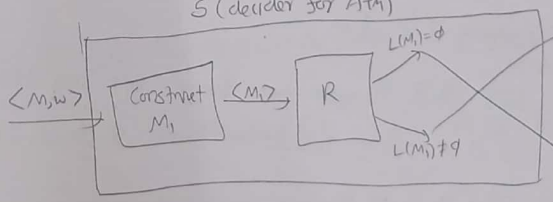


S (decider for A_{TM})



$$L(M_1) = \begin{cases} w & \text{if M accepts w} \\ \phi & \text{otherwise} \end{cases}$$

S (decider for A_{TM})



We know A_{TM} undecidable.

∴ Contradiction.

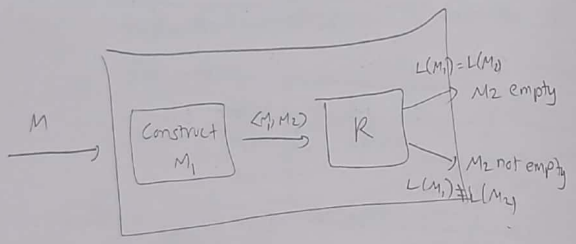
Q $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs \& } L(M_1) = L(M_2) \}$

is undecidable.

⇒ E_{TM} reduces to EQ_{TM}

Say EQ_{TM} decidable. So decider R for EQ_{TM} exists.

Construct a TM M_1 that will reject all the inputs $L(M_1) = \emptyset$



But E_{TM} undecidable

∴ Contradiction.

* Mapping reduction

Let A & B be two langs. We say that there is a mapping reduction from A to B denoted by $A \leq_m B$.

If there is a computable func. $f: \Sigma^* \to \Sigma^*$ such that $\forall w \in A \iff f(w) \in B$

If $A \leq_m B$ implies $\bar{A} \leq_m B$

Ex] Let A be $0^* = \{\epsilon, 0, 00, \ldots\}$

B be $\{0, 1\}$

Let f be a func defined as

if $w \in 0^*$, $|w|$ is odd, then $f(w) = 1$

else if $w \in 0^*$, $|w|$ is even, then $f(w) = 0$

else if $w \notin 0^*$, then $f(w) = 11$

* Theorems

1] If $A \leq_m B$ & $B \leq_m C$, then $A \leq_m C$.

2] If $A \leq_m B$ & B is decidable, then A is decidable

3] If $A \leq_m B$ & A is undecidable, then B is undecidable.

4] If $A \leq_m B$ & B is REL (Turing Recognizable), then A is REL

5] If $A \leq_m B$ & A is not REL (Turing Unrecognizable), then B is not REL

6] If $A \leq_m B$, & A is not co-turing recognizable, then B is not ——— 〃 ———

* Rice theorem

Let P be a non-trivial property of language of TM (RE L), then

$L_P = \{ \langle M \rangle \mid P(L(M)) = 1 \}$ is undecidable

where L_P is the lang of encoding of TMs that satisfy P.

$P(L) = 1$ if L satisfies property P
$P(L) = 0$ if L doesn't ——— 〃 ———

Trivial property } If all (REL) langs satisfy a property (or)

Ex] P: TMs that recognize the lang.

Non-trivial } opposite

Ex] P: L is a finite lang
P: L has odd string
P: $L = \Sigma^*$

* Theorem: EQ_{TM} & $\overline{EQ_{TM}}$ are not Turing recognizable.

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

$\overline{A_{TM}}$ → not 〃 〃

$\overline{A_{TM}} \leq_m EQ_{TM}$: To prove EQ_{TM} is not Turing recognizable

$\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$: To prove $\overline{EQ_{TM}}$ ——〃——

TM 'F' computes mapping reduction

F: on input $\langle M, w \rangle$ where $M \rightarrow TM, w \rightarrow$ string

M_1: on any input reject

M_2: on any input, run M on w if it accepts w, then accept everything.

output $\langle M_1, M_2 \rangle$

(not complete)

> Apr 4 => Tuesday

* Complexity theory :- (read slides)

> Apr 11 => Tuesday

* Complexity class:
  - P class (polynomial)
  - NP-class (non-deterministic polynomial)

  ↓

  Class of problems (langs) that can be solved in polynomial time using NTM.