Mathematical Theory of Computation

Muralikrishnan S

- Obj: To mathematically model what an alg. is.

→ Turing Machine Model - Alan Turing
→ λ calculus - Alanzo Church
→ Combinatorics
→ Term rewriting System - Amil Post
→ Recursive fn Model - Kürt Gödel
→ Grammatical Structures - Noam Chomsky

1940 (around)

Church Turing Thesis

Turing m/c is the standard model of reference for Algorithmic Computability
(Just like carnot engine is an ideal engine for comparison)
It formalises the notion of algorithmic computation.

→ What is mathematical formln of a problem ??
- Limit problem to Y/N problem OR Decision Problems

Problem: Given a no: N, check whether N is prime or not ??

⇒ Def: An Alphabet (Σ) is a finite set of symbols. eg: ASCII, Binary {0,1}, Decimal {0,9}

⇒ A word/string over Σ is a finite sequence of symbols from Σ.
eg 0119 → dec string, abcca → ASCII
Given 'n' → given a finite string over some fixed alphabet Σ.

⇒ Def: A language over alphabet Σ is any collection (subset) of strings over Σ.

⇒ Σ^*: Set of all strings over Σ.
eg $\Sigma = \{0,1\}$ $\Sigma^* = \{\epsilon, 00, 01, 10, 11, 000, 001 \cdots\}$
i.e A language L over Σ is any ~~subset~~ element of 2^{Σ^*} or subset of Σ^*

ϵ - null string - identity when concatinated with any strig gives the same.

⇒ Set of language is uncountably infinite.

⇒ Any decision problem can be modelled as a language over some finite alphabets
$\Sigma = \{0,1\}$ Parity = $\{0, 0^{\Sigma_i}, i \geq 0, 1 \cdot 1^{2i}, i \geq 0, \cdots\}$
= All string with even 1s.

(Q) Decision Problems ⇔ Language recognition problems.
Modelled as: Given language L over Σ and a string $x \in \Sigma^*$
To determine if $x \in L$??

a) ~~Set of all binary strings~~ Given $x \in \{0,1\}^*$. Is $x \in$ PARITY
(Decision problem - strigs that map to 0 or 1)
Yes/No

A computational problem is a function which maps from string to string.

$f: \{0,1\}^* \mapsto \{0,1\}^*$

$x \mapsto f(x)$

A computational solution to f is an algorithm A.

such that an i/p $x \in \{0,1\}^*$ A satisfies :-

a) A terminates in finite # of steps

b) The o/p $A(x) = f(x)$

Finite State Machine : finite no: of states, potentially ∞ memory.

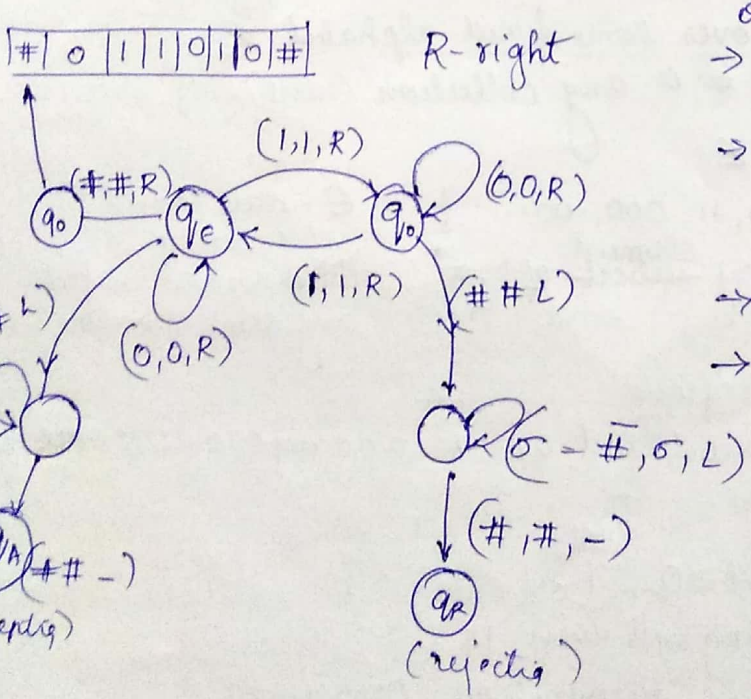
Min prgm segments - read/store
- if -
- goto

* F.S.M has read only memory

Turing machine : Any machine : finite state, ∞ memory

* Requirements for a General Purpose Computing System :-
- Finite states, Test & jump transition s/m (memory)
- An unrestricted amnt of read-write memory.

Finite State Machine for PARITY



It's just a convention to come back to starting pt.

At each step the m/c
→ reads from input ponted (Tape Head)
→ write back to the tape (Not possible for FSM)
→ Move the head left/right (optional)
→ Transfers control to some other state. (finite state)

Turing

Church Turing Thesis: Turing m/c model is sufficiently rich to capture "ANY" computing system

Computability Theory: What is programmable & what is not ?? i.e what language can be recognised or what decision problem can be solved with TM's & what's not ??

Turing Machine $M = (Q, \Sigma, \delta, q_0, q_A, q_R)$ where

i) Q is a finite set of state

ii) Σ is an alphabet (finite) $(\# \in \Sigma)$ - spl symbol in alphabet

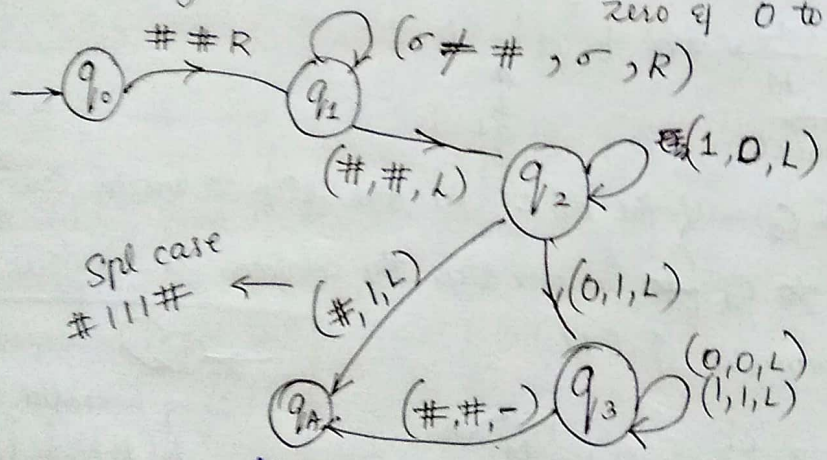
iii) $q_0, q_A, q_R \in Q$ - start, accepting & rejecting state

iv) δ - transition function δ is a map that takes 2 i/p state & symbol (present) & maps to next m/c state, what are you gonna put at the tape end and whether we should move to left, right or stay (TM we can write)

$$\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{L, R\}$$

Q) Turing machine for computing $f(x) = x + 1$

ans) Given eg: #1011011# - In the last all ones till 1st zero is changed to zero & 0 to 1.



$(\sigma \neq \#, \sigma, R)$

$\delta(q_0, \#) = (q_1, \#, R)$

$\begin{cases} \delta(q_1, 1) = (q_1, 1, R) \\ \delta(q, 0) = (q, 0, R) \end{cases}$

Spl case

#111#

000

Configuration ★ (is unique ∵ δ is a fn)

Describe a) State

b) Tape content (b/n the 1st # on L & 1st # on R)

c) Tape head position

} for stopping a process & fn starting over again or need all this.

A configuration of M is a string in

$$\# (\Sigma - \#)^* ((\Sigma - \{\#\}) \times Q), (\Sigma - \{\#\})^* \#$$

eg initial #01101# → c_0 start

q_0 MV 1

#0 1101#

or configrn: $\Sigma^* \times (\Sigma \times Q) * \Sigma^*$ with the condition that the left most & right-most symbol are # blank

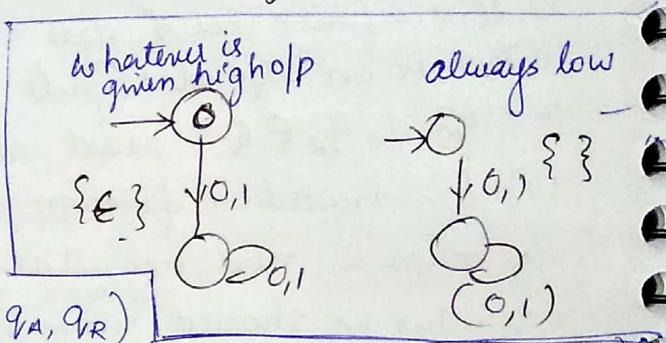
⇒ Given a TH M & i/p x (with our pgm convention) The start config of M on i/p $x \in (\Sigma - \#)^*$

Σ - nul string i/p is possible

$C_M^0(x) = C_0 = \underset{q_0}{\#}\, x \#$

$C_M^f(x)$ Suppose the m/c is computing fn f. Then after computation the final config.

$C_M^f(x) = C_f = \underset{q_A}{\#}\, f(x) \#$



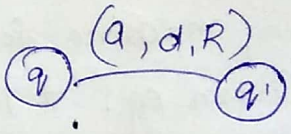
Turing Machine

A configurn of a Turing m/c $M = (Q, \Sigma, \delta, q_0, q_A, q_R)$

a string of the form $\# \alpha \underset{q}{\sigma} \beta \#$ (There are ∞ many blanks to right & left)

$\alpha, \beta \in \Sigma^*$ $\sigma \in \Sigma$ $q \in Q$

Starting config on i/p x : $\underset{q_0}{\#}\, x \#$

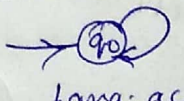


Let C_1, C_2 be 2 config

$\# \alpha\, b\, a\, c\, B\, \# \xrightarrow[M]{1} \# \alpha\, b\, d\, c\, B\, \#$

q ⇒ one step q'

& We say $C_1 \xrightarrow[M]{1} C_2$ if the m/c in one step transitn can change from configrn C_1 to C_2 {∵ In one step the config is unique ∵ δ is a fn}. Deterministic F.S.M



hang · accepted
$L(M) = \{\}$

$C_0 \xrightarrow[M]{1} C_1 \xrightarrow[M]{1} C_2 \xrightarrow[M]{1} C_3 \cdots \xrightarrow[M]{1} C_n$

then we say $C_0 \xrightarrow[M]{i} C$ $\forall i \geq 0$

Assumptions: From state q_A & q_R the only transitn possible are

$\delta(q_A, \sigma) = (q_A, \sigma, -)$

$\delta(q_R, \sigma) = (q_R, \sigma, -)$

let $f: \Sigma^* \mapsto \Sigma^*$ $f(x) = x^2$ assuming $\Sigma \div \{0,1\}$

A turing Machine M computes a function f if

$\forall x \in \Sigma^* \; \exists i \in N \quad C_0 = (\underset{q_0}{\#}\, x \#) \xrightarrow[M]{i} (\underset{q_A}{\#}\, f(x) \#)$

Def: A fn $f: \Sigma^* \to \Sigma^*$ is turing computable if there exist a turing m/c M such that M computes f.

→ There exist functions (from $\Sigma^* \to \Sigma^*$) that are not turing computable

[• strings - countably ∞
• fn from $\Sigma^* \to \Sigma^*$ - uncountable
• Set of m/c is countable] → ∴

* Write $C_0 \xrightarrow[M]{*} C_n \iff \exists i \geq 0 \quad C_0 \xrightarrow[M]{i} C_n$ ($C_0 \xrightarrow[M]{*} C_n \Leftrightarrow$ reachable)

Note :- There is a bijection b/n $\{L \mid L \subseteq \Sigma^*\}$ and $\{f : \Sigma^* \to \{0,1\}\}$ Boolean fns.

$L \xrightarrow{\text{Bijection}} f_L(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases}$ Set of all boolean fns: uncountably ∞

$L_f = \{x \mid f(x) = 1\} \longleftarrow f$

Consequence: Since $\{L \mid L \subseteq \Sigma^*\} = 2^{\Sigma^*}$ is uncountable there exist a lang. L such that f_L is not turing computable.

A Turing M/c M decides a language L if $\forall x \in \Sigma^*$

if $x \in L \Rightarrow (\#_{q_0} x \#) \xrightarrow[M]{*} (\#_{q_A} \#)$ (erasing tape once we reach accepting state is just a technicality)

if $x \notin L \Rightarrow (\#_{q_0} x \#) \xrightarrow[M]{*} (\#_{q_R} \#)$

i.e $x \in L$ then M accepts else M rejects. (• soundness • completeness • guaranteed termination)

Def: A L is turing decidable if $\exists$ M/c M such that M decides L. (decide & accept is diff)

A m/c M accepts $L \subseteq \Sigma^*$

if $\forall x \in \Sigma^*$ if $x \in L \; (\#_{q_0} x \#) \xrightarrow[M]{*} (\#_{q_A} \#)$

if if $x \notin L \; (\#_{q_0} x \#) \xrightarrow{*/M} (\alpha \#_{q_A} \beta)$ for any $\alpha, \beta \in \Sigma^*$

M/c may go to infinite loop or reject

L is Decidable (Turing) $\Rightarrow$ L is Accentable (Turing)

* HALTING PROBLEM: Given (M, x) M is a TM $x \in \Sigma^*$ will M halt on i/p x.

Recap

Notation : $C_1 \vdash_M^* C_2$ is used for "C_2 is reachable from C_1 in 0 or more steps"

Def : A language L is accepted by a m/c M, if

1) $\forall x \in \Sigma^*$ if $x \in L$ $\quad C_0^M(x) \vdash_M^* C_A^M$

2) $\forall x \in \Sigma^*$ if $x \notin L$ $\quad C_0^M(x) \not\vdash_M^* C_A^M$

M decides L if

1) $\forall x \in L$, $C_0^M(x) \vdash_M^* C_A^M$ $\quad$ 2) $\forall x \notin L$ $\quad C_0^M(x) \vdash_M^* C_R^M$

A $f : \Sigma^* \to \Sigma^*$ is a turing computable function if there exist a TM, M such that ~~starting~~ $\forall x \in \Sigma^*$

$$C_0^M(x) \vdash_M^* \#\, q_A\, \#\, f(x)\, \#$$

L is a turing decidable language if there exist a m/c M that decides L (recursive language)

L is turing acceptable if there exist a turing m/c M that accepts L (recursively enumerable language)

Properties of decideable languages

1) If L_1, L_2 are T.D then $L_1 \cup L_2$, $L_1 \cap L_2$, $\bar{L_1}$, $\bar{L_2}$ ... are T.D

$M_{L_1 \cup L_2}(x) = \{$ /* let M_1 decide L_1 & M_2 decide L_2.

If $M_1(x)$ = Accept, return 1

If $M_2(x)$ = Accept, return 1

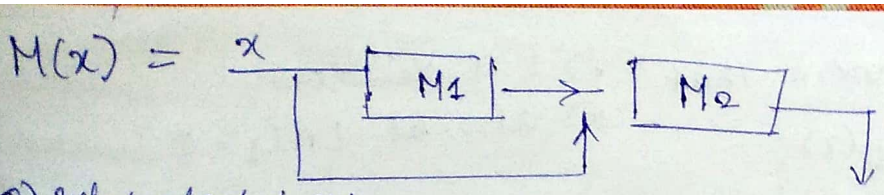
return 0;

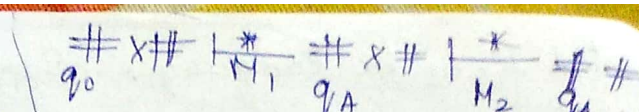
$\}$

RE TA

If L_1, L_2 are recursively enumerable or turing acceptable. Is $L_1 \cap L_2$ RE?

let M_1 accept L_1 & M_2 accept L_2





M(x) { if ($M_1(x)$ & $M_2(x)$) Accept else reject }

Q) What about $L_1 \cup L_2$

(If m/c goes to ∞ loop)

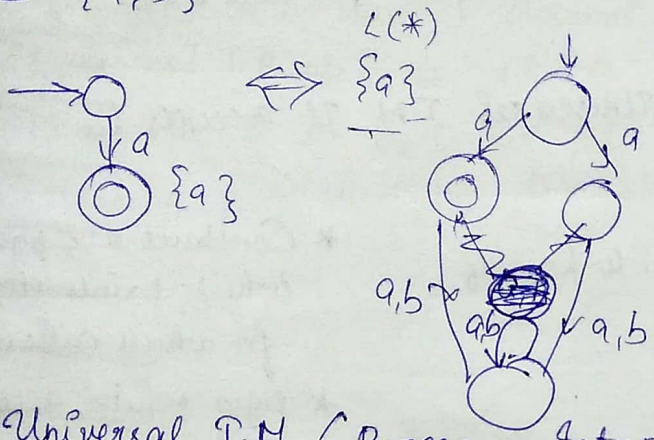
Keep simulating M_1 & M_2 on alternate steps & accept if one of them is accepting.

- Concurrency

~~If one terminates continue with~~ & rejects

NFA

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



Complement wont be obtained if we interchange accepting & non accepting state.

Universal T.M (Program Interpreter)

$L_u = \{ (M, x) \mid M \text{ accepts } x \}$. = HALTING PROBLEM

UNIVERSAL LANGUAGE

Is it possible to design a TM m/c U such that U accepts i/p (M, x) ⇔ M accepts x.

↓

YES m/c U is called Universal T.M

U

#M#x (See Marvic Minsky) - if interested)

q_0 ↑

If M accepts x, U accepts M(x)

Given M define L(M) = lang. accepted by M $L(u) = L_u$

Qn: Is L_u decidable??

Prove that L_u is not decidable

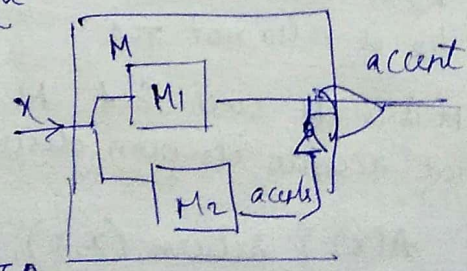
⇒ Suppose L, $\bar{L}$ are T.A.

(let M_1, M_2 accepts L & $\bar{L}$

↳ Then both L & $\bar{L}$ are T.D

Converse also true.

L is T.D iff both L & $\bar{L}$ are T.A.

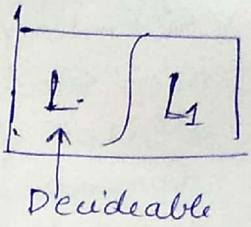


Time shared/ not parallel

Let L, L_1 be languages. Suppose that i) L is decidable
ii) also let $L \cap L_1 = \phi$

- If L_1 is decidable then $A_{L_1 \cup L}(x)$

{ If $(x \in L_1)$ or $(x \in L)$
return 1;
else return 0; }



- If $L_1 \cup L$ is decidable, then is L_1 decidable?
when L is decidable, $L_1 \cap L = \phi$ then $L_1 \cup L$ is decidable $\Leftrightarrow$ L_1 is decidable
- If L is decidable, $L_1 \cap L = \phi$ $L \cup L_1$ is decidable $\Leftrightarrow$ L_1 is decidable

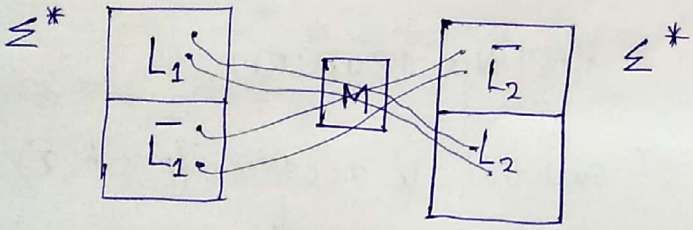
$L_u = \{ (M, x) \mid M \text{ accepts } x \}$
- Is turing acceptable because the Universal T.M U accepts L_u.

Redⁿ b/n 2 languages L_1 & L_2

Def: A turing m/c M is a redⁿ from L_1 to L_2 if
i) $\forall x \in L_1 \quad M(x) \in L_2$
ii) $\forall x \notin L_1 \quad M(x) \notin L_2$

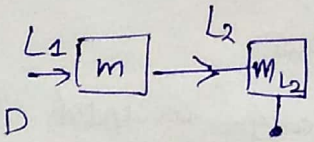
* Construct a C pgm which prints itself fn which calls itself
* pgm which accepts nothing but itself

Theorem: L_u is not turing decidable



Def: $L_1 \leq_m L_2$ [L_1 is (many-one) reducible to L_2] if there is redⁿ from L_1 to L_2.

Obs: If $L_1 \leq_m L_2$



a) If L_2 is T.D then L_1 is T.D
b) If L_2 is T.A then L_1 is T.A

Reducability carries :-
a) Difficulty from left to right
b) Solvability from right to left

If L_1 is undecidable L_2 is undecidable
If L_1 is not T.A L_2 is also not T.A

Define: $L_0 = \{ M \mid M \text{ is a TM such that } M \notin L(M) \}$
L_0 set of machines that accepts its own description.

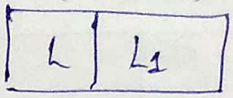
Claim: $L_0 \leq_m L_u$ $\quad A(x)$ { return (x, x); }
If $x \in L_0 \Leftrightarrow x = M$ for some T.M M such that $M \in L(M) \Leftrightarrow (M, M) \in L_u$
$\Rightarrow$ L_0 is not TD then L_u is not turing decidable.

L_0 is not T.D $\Leftrightarrow$ $\bar{L_0}$ is not T.D

$\overline{L_0} = \{ x \mid x$ doesnot represent a TM $\} \cup \{ M \mid M$ is a TM such that $M \notin L(M) \}$

$=$ $L \cup L_1$ $(L \cap L_1 = \phi)$

(L : easily decidable)



To show $\overline{L_0}$ is undecidable shout L_1 is undecidable.

Claim: L_1 is not T.A

Proof: Suppose L_1 is T.A. Let T be a TM that accepts L_1

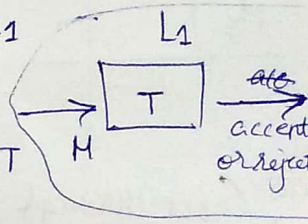
T accepts input M iff M doesnot accept M.

When i/p to T is T we get T accepts i/p T $\Leftrightarrow$ T doesnot accept T is a contradiction. Hence T doesnot exist.

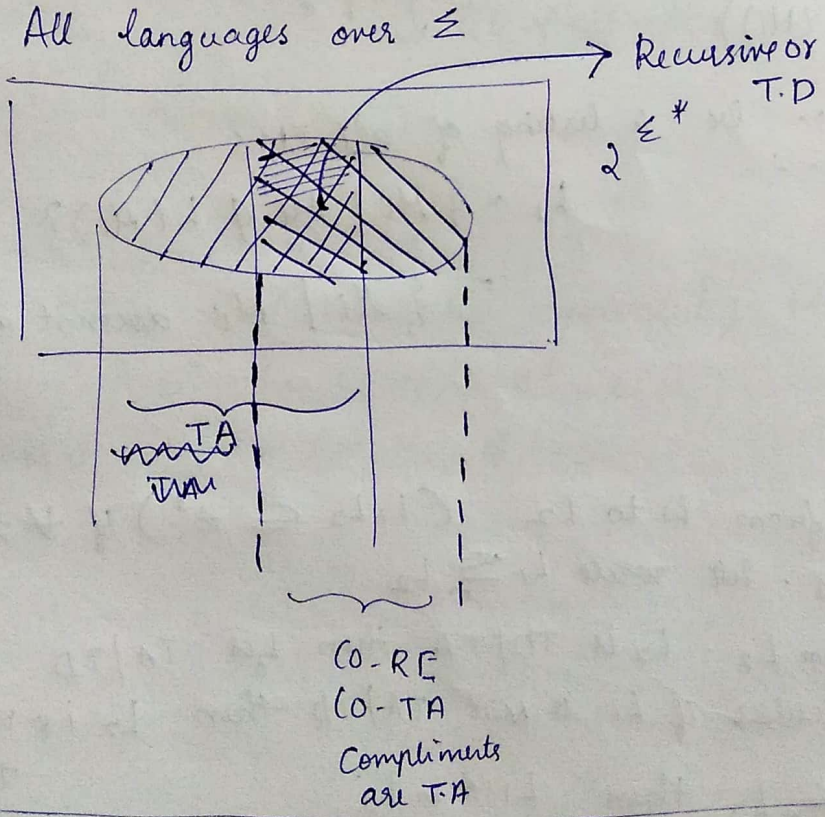
$\Rightarrow L_1$ is not T.A $\Rightarrow \overline{L_0}$ is not TA

$\Rightarrow L_0$ is not TD $\Rightarrow L_u$ is not T.D $\because L_0 \leq_m L_u$.

$\therefore$ Halting problem is not decidable.



- L_u is ~~turn~~ TA $\Rightarrow L_0$ is T.A.
- $\overline{L_u}$ is not TA



Another way of showing that unsolveable problem exists.

Obs: The set of all TMs is countable. Let $M_0, M_1, M_2, M_3 \ldots$ be an enumern of all TMs.

$\because \Sigma^*$ is countably ∞ let $x_0, x_1, x_2, x_4, x_5 \ldots$ be an enumern of all strings in Σ^*

	x_0	x_1	x_2	x_3	x_4 ...
M_0					
M_1					
M_2					
M_3					
⋮					

$L = \{\}$
for $i = 0$ to ∞
{ if x_i is not accepted by M_i add x_i to L
}
$L = \{ x_i \mid x_i \notin L(M_i) \}$

(?) Cross through diagonal why??

Claim: L is not accepted by any M_i

Suppose M_i accepts L, then $x_i \in L(M_i) \Leftrightarrow x_i \in L$
$\Leftrightarrow x_i \notin L(M_i)$ - Contradiction.
(By def of L)

$\Downarrow$
$L \equiv L(M_i) = L$
$(\because L = L(M_i))$

Recap: Let $M_0, M_1 \cdots$ be a listing of all TMs

M_0 M_1 M_2 ...

M_0 □
M_1 □
M_2 □
⋮

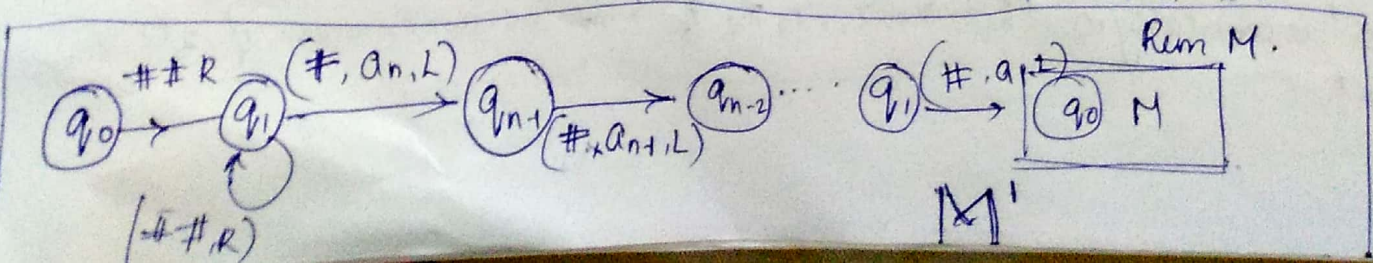
$L_1 = \{ M_i \mid M_i \notin L(M_i) \}$
$= \{ M_i \mid M_i$ doesnot accept $M_i \}$

A TM M is a redn from L_1 to L_2 $(L_1, L_2 \subseteq \Sigma^*)$ if $\forall x \in \Sigma^*$
$M(x) \in L_2 \Leftrightarrow x \in L_1$. We write $L_1 \leq_m L_2$

Properties: If $L_1 \leq_m L_2$. L_2 is TD/TA then L_1 is TA/TD
In particular if L_1 is Not TA/TD then L_2 is not T.A/T.D

If $L_1 \leq_m L_2$ $L_2 \leq_m L_3$ then $L_1 \leq_m L_3$

$A(M, x)$ { // i/p is an instance of L_u $x = a_1 a_2 \cdots a_n$ (x is accepted by M)
return M' // M' looks as below }
M/c that erases its i/p writes x to the tape &



M accepts $x \Leftrightarrow$ M' accepts all i/ps

i.e $(M,x) \in L_u \Leftrightarrow L(M') = \Sigma^*$

$(M,x) \in L_u$ then $L(M') = \Sigma^* \Rightarrow M' \in L_0$

$(M,x) \notin L_u$ then $L(M') = \phi \Rightarrow M' \notin L_0$

Whatever be the i/p. say w which is not accepted by M. Then M' erases w & writes x (which is accepted by M) on tape
∴ M' accepts every string

$L_0 = \{ M \mid L(M) = \Sigma^* \}$ = set of all pgms that accept all i/ps

A is redn from L_u to L_0 i.e $L_u \leq_m L_0$

$L_1 = \{ M \mid L(M) \neq \phi \}$ $\therefore L_u \leq_m L_1$

$L_a = \{ M \mid L(M)$ contains $\epsilon \}$ $L_u \leq_m L_a$

$\{ M \mid L(M)$ is turing acceptable $\}$

$\{(M_1, M_2) \mid M_1, M_2$ are T.M such that $L(M_1) = L(M_2)\}$ - Equivalence pgm

Given 2 pgms are they computing the same thing.

Claim: $L_0 \leq_m L_{EQ}$ L_0: set of all mfcs that accepts everything.

Redn A(M) { $M = M_1$ return (M, M_1); }

$M_2 = q_0 = q_A$

$L(M) = \Sigma^*$

If $L(M) = \Sigma^*$, then $L(M_1) = \Sigma^* = L(M_2)$

i.e If $M \in L_0$ then (M_1, M_2 are equivalent) $M_1, M_2 \in L_{EQ}$

If $L(M) \neq \Sigma^*$, then $L(M_1) \neq \Sigma^* = L(M_2)$

i.e If $M \notin L_0$, then $(M_1, M_2) \notin L_{EQ}$