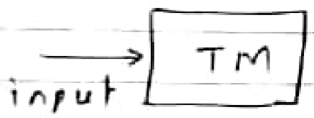
(3) Recursive languages and Recursively Enumerable language

→

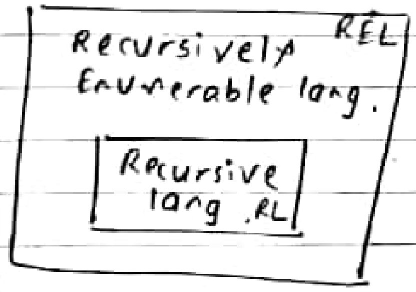


(1) Halts and accepts the input
Lang = Tm accepts = R.Enu.lang.

(2) Halts and rejects the input

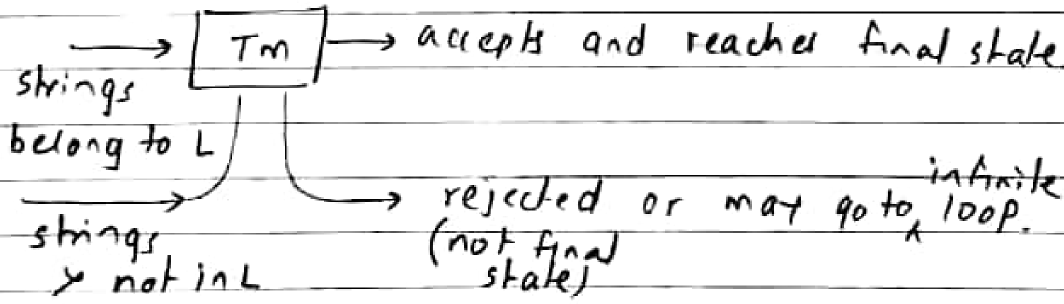
(3) Never halts (falls into loop)

(4) crash

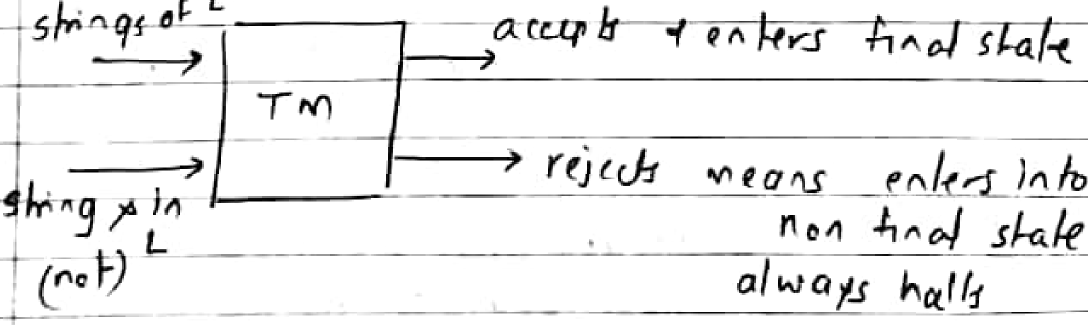


RL ⊆ REL
subset of.

(4) Recursively Enumerable Language –



(5) Recursive Language.



so Recursive lang. is also called as Turing decidable lang.

e.g. $\{a^n b^n c^n \mid n \geq 1\}$ is recursive closure properties of Recursive lang.

① L_1 and L_2 = Recursive lang.

$L_1 \cup L_2$ is also recursive.

ie if TM halts at L_1 and

TM halts at L_2

then TM halts at $L_1 \cup L_2$

② $L_1 \cdot L_2$ is also recursive.

$L_1 = \{a^n b^n c^n \mid n >= 0\}$

$L_2 = \{d^m e^m f^m \mid m >= 0\}$

$L_1 \cdot L_2 = \{a^n b^n c^n d^m e^m f^m \mid n \geq 0$ and $m \geq 0\}$

is also recursive

③ if L_1 = recursive L_1^* is also recursive

$L_1 = \{a^n b^n c^n \mid n \geq 0\}$

$L_1^* = \{a^n b^n c^n \mid n \geq 0\}^*$ is also recursive.

④ if L_1 and L_2 are recursive

$L_1 \cap L_2$ is also recursive.

$L_1 = \{a^n b^n c^n d^m \mid m \geq 0 \;\; n \geq 0\}$

$L_2 = \{a^n b^n c^n d^n \mid n \geq 0\}$

$L_3 = L_1 \cap L_2 = \{a^n b^n c^n d^n \mid n \geq 0\}$ is also recursive.

⑤ If L is recursive

$\Sigma^* - L$ is also recursive.

- closure properties of Recursively Enumerable languages

REnum Lang $L_1 \vee L_2$ are REnum. lang.

$L_1 \cdot L_2$ are — " —

$L_1{}^*$ is — " —

$\Sigma^* - L_1$ is Not need Recursively Enum. lang.

Recursive lang (RL)	Recursively Enum lang. (REnuL)
① lang is RL if and only if there ~~is~~ exits a membership algo. for it.	① Lang is REnuL if there exists TM that accepts it
② RL are also called as TM decidable lang.	② REnuL are also called as TM acceptable lang
③ REnuL may not be RL but RL is REnuL.	③ RL is subset of REnuL.
④ A problem whose lang. is recursive is said to be decidable	④ A problem whose lang. is R.Enu. is said to be semi-decidable or undecidable
⑤ Recursive lang of TM = recursive Post m/c = recursive 2PDAs.	⑤ REnuL on TM is = 2PDA REnuL = is also Post m/c REnuL.

① Context sensitive Grammar-

The context sensitive Grammar is defined as

$$G = (V, T, P, S)$$

V = set of non terminals
T = set of terminals
S = starting symbol
P = set of productions.

CSG – Language is context sensitive lang.
– Type (1) grammar.
– M/C is Linear bounded automata
Non deterministic TM.
– production rules are of type

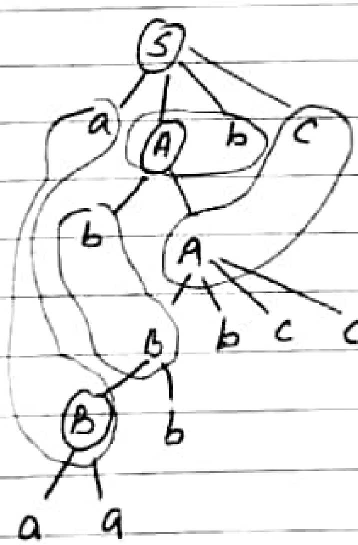
$$\alpha \rightarrow \beta$$

$\alpha = (V \cup T)^*$ $\beta = (V \cup T)^*$ $\alpha \neq \epsilon$ $|\beta| \geq |\alpha|$

length of string at LHS $\leq$ length of string at RHS.

⊛ e.g. lang. $L = \{a^n b^n c^n \mid n > 0\}$ is context sensitive.

$S \rightarrow abc \mid aAbc$
$Ab \rightarrow bA$
$Ac \rightarrow Bbcc$
$bB \rightarrow Bb$
$aB \rightarrow aa \mid aaA$



Context sensitive Grammar is Recursive

⊛ $L = \{a^n b^m c^n d^m \mid n, m \geq 1\}$ is CSL

① $L = \{ a^n b^n c^n \mid n \geq 1 \}$

Rule :- $S \to aSAc \mid abc$

$cA \to Ac$

$bA \to bb$

② $L = \{ a^n b^m c^n d^m \mid m, n \geq 1 \}$

$S \to aAcD \mid aBcD$

$A \to aAc \mid aBc$

$Bc \to cB$

$Bb \to bB$

$BD \to Ed$ $\quad$ $BD \to FDd$

$cE \to Ec$ $\quad$ $cF \to Fc$

$bE \to Eb$ $\quad$ $bF \to Fb$

$aE \to ab$ $\quad$ $aF \to abB$.

② Linear Bounded Automata.

is ↳ (NDTM)

LBA is NDTM with two conditions.

① Σ includes two special symbols ¢ and \$ as left end marker and right end marker.

② It has no moves to left of ¢ and no moves to right of \$ and it can not overwrite another symbol over ¢ & \$.

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

Q → Finite set of states of TM

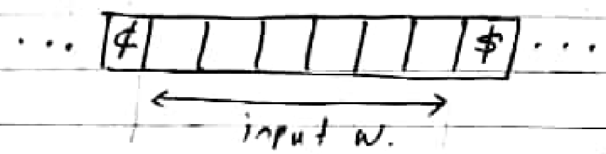
Σ → Finite set of input symbols

Γ → tape symbols

q_0 → Initial state.

δ - mapping funⁿ

$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$



linearly bounded tape.

A linear funⁿ is used to restrict the length of tape.

¢ will not allow R/w head to go to left further

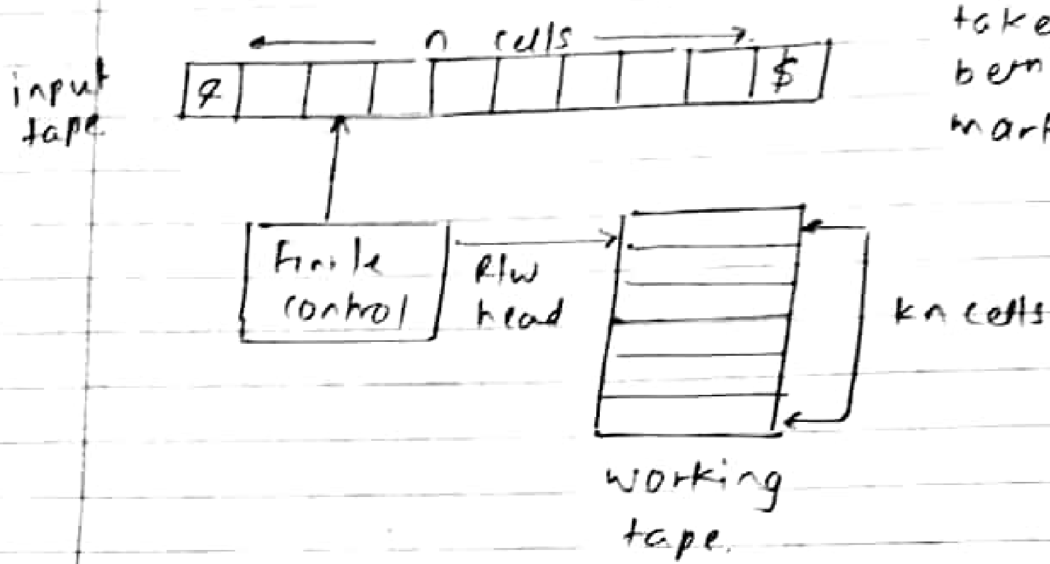
\$ will not allow R/w head to go to right

R/w head can not replace any symbol on tape with ¢ or \$. and ¢ or \$ can not be overwritten with anything else

LBA with empty lang is undecidable.

→ input string w is accepted by tape

$|w| = n - 2$



computation takes place betn end markers.

K = constant specified in desc. of LBA
= does not depend on input
= property of the m/c.

w is accepted by LBA where it ~~it~~ is also accepted by TM using no more than kn cells of input tape

In LBA

on input tape, the head never prints and never moves to left.

on working tape the head can modify contents in any way without any restriction.

④ Decidability and undecidability

Language L is said to be decidable if there exists

$\xrightarrow[L]{}$ [TM] → accepts yes or no

(- - -) string of L

(- - -) not in L ⟶ rejects means stops in non-final state.

it Never loops infinitely.

So we can categorise the problems of TM into two

① Decidable Problems

→ problems in which TM can halt in accepting or rejecting state are called decidable problems.

→ decidable L

$w \in L$, M enters in q_{accept}

$w \notin L$, M enters in q_{reject}

- ~~Thus~~ such Problems have two answers (yes/no).

such L is recursive.

- DFA, CFG, CSL is decidable

② Undecidable problems.

→ Problems that are only solved by TM are called as undecidable problems, TMs may not halt at all. if they do not accept input.

→ Problems that do not have algo. are undecidable we use diff techniques to solve.

- A problem is said to be undecidable if it is unsolvable.

- To prove given problem is undecidable we can show that there is no TM that can decide lang.

- RE_{NUL} is undecidable.

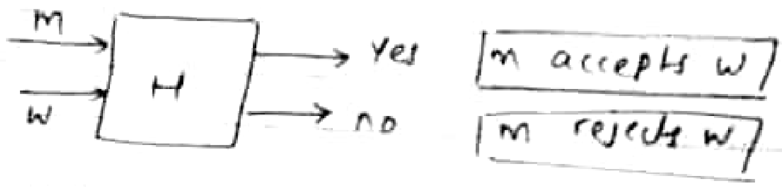
Undecidable problem -

① e.g. membership problem.

Input - Turing machine M
- string w.

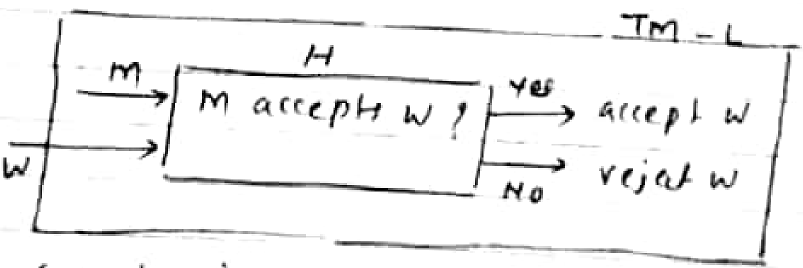
Question - Does M accepts w?
w ∈ L(M) ?

Prove
Assume
- membership problem is decidable
- so there exists TM H that solves the membership problem
-



- let L is recursively enumerable lang. and let M accepts L.

We will prove that L is also recursive. so TM that accepts L and halts on any input



- so L is recursive.

- since L is chosen arbitrarily, every recursively enumerable language is also recursive.

- But there are recursively enumerable languages which are not recursive.
- Contradiction !!! so membership problem is undecidable

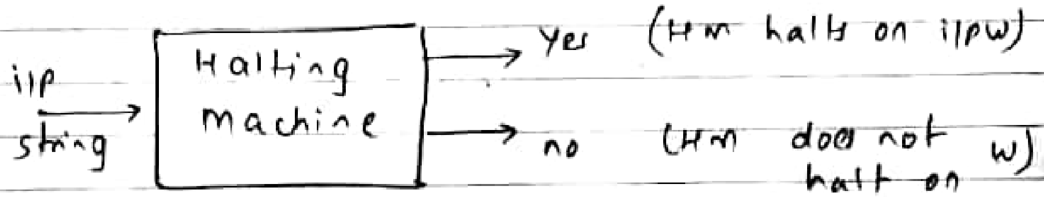
undecidable problem,

② e.g Halting problem of TM.

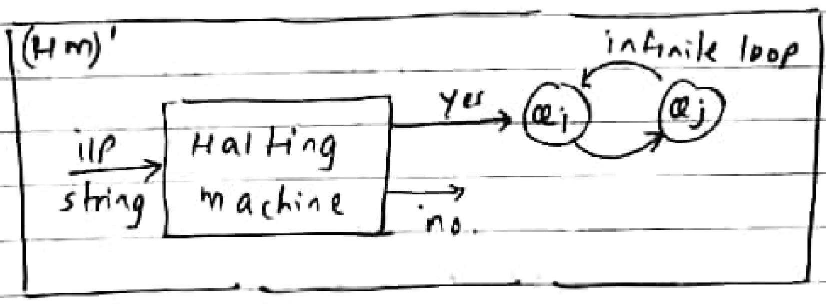
input - Turing machine M
- string w.

Quest - Does M halt on input w?

Proof - Assume TM solves this problem.
called as
(Halting machine)



Now we design Inverting halting machine
(HM)'



Further a machine $(HM)_2$ which input itself is constructed as follows.

- if $(HM)_2$ halts on i/p, loop forever.

- else halt

- contradiction - so halting problem is undecidable.

(6) The Post correspondence problem (PCP)

PCP - is undecidable problem
- useful tool to prove certain problems in formal lang to be undecidable.
- It deals with the manipulation of strings

$P = \{[t_1/s_1], [t_2/s_2], [t_3/s_3] \cdots [t_n/s_n]\}$

t_n & s_n are non null strings over Σ

- PCP is concerned with determining whether a collection of dominos has a match.

PCP = { <D> | D = set of dominos with a match }

- belongs to a class of yes/no problem
- task is to find

$t_1 t_2 t_3 \cdots t_n = s_1 s_2 s_3 \cdots s_n$

reading off sym from top   string of sym on bottom.

- match is a seq. $i_1 i_2 \cdots i_n$ of P compon

where $t_{i_1} t_{i_2} \cdots t_{i_n} = b_{i_1} b_{i_2} \cdots b_{i_n}$

some set of dominos do not have a match as the top always have more symbols than bottom.

Scanned by CamScanner

PCP is to find an algorithm that tells us

For a given Post correspondence system P, whether or not there exists a match of P ?

It's easy to describe this problem as a collection of n distinct groups of dominos where each domino from i^{th} group containing two strings, one on each side, having string t_i on the top half
string s_i on the bottom half

- Duplicate domino's can be used.
- It is not necessary to use all distinct domino types.

sol^n $\boxed{\frac{t_1}{s_1}}$ $\boxed{\frac{t_2}{s_2}}$ .... $\boxed{\frac{t_n}{s_n}}$

PCP = useful tool in logic
& theory of formal lang.

for proving the undecidability of many other problems by means of reducibility.

If there is a sol^n to PCP, there ~~exit~~ exists infinitely many solutions.

e.g. let a Post correspondence system (PCS) P is represented by

[10/101], [01, 100], [0, 10], [100, 0], [1, 010]

find whether there exists a match of P

soln

$\boxed{10}$	$\boxed{1}$	$\boxed{01}$	$\boxed{0}$	$\boxed{100}$	$\boxed{100}$	$\boxed{0}$	$\boxed{100}$
t_1	t_5	t_2	t_3	t_4	t_4	t_3	t_4

$\boxed{101}$	$\boxed{010}$	$\boxed{100}$	$\boxed{10}$	$\boxed{0}$	$\boxed{0}$	$\boxed{10}$	$\boxed{0}$
s_1	s_5	s_2	s_3	s_4	s_4	s_3	s_4

there exists seq. of dominos

{[10/101], [1, 010], [01, 100], [0, 10], [100, 0]
[100, 0], [0, 10], [100, 0]}

such that reading off the top string is same as reading off the bottom string.

1010101001001000100

(2) $\Sigma = \{0, 1\}$ x & y - list of three strings

	List x	List y
i	w_i	x_i
1	b	bbb
2	babbb	ba
3	ba	a

→ PCP has a soln $p = 4$ $i_1 = 2$ $i_2 = 1$ $i_3 = 1$ $i_4 = 3$

$\underset{w_2}{babbb}\ \underset{w_1}{b}\ \underset{w_1}{b}\ \underset{w_3}{ba} = \underset{x_2}{ba}\ \underset{x_1}{bbb}\ \underset{x_1}{bbb}\ \underset{x_3}{a}$

→ prove that PCP has no solution X and Y be list of over $\Sigma = \{0,1\}$ three strings.

i	list X w_i	List Y x_i
1	10	101
2	011	11
3	101	011

→

no match

w_2 011 w_3 101

x_2 11 x_3 011

try

w_1 $\overset{w_1}{10}$ — $\overset{w_3}{101}$ — 011 No match.

x_1 101 — ~~101~~ 011 11