Turing Machine

Turing Machine is advancement over FA & PDA made By Alan turing (1912-1954). Turing machine consist of

→ finite control
→ tape
→ A read/write head that can be used for reading & writing on the tape.

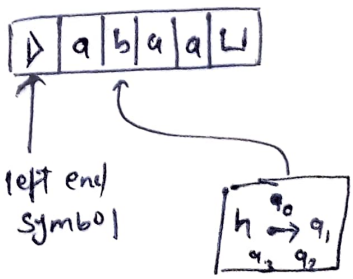
The control unit perform 2 function that depends upon its current state & tape symbol currently scanned by read/write head.

1st function :- Put the control unit on the new state.

2nd function :- either,

(a) Write the symbol on the tape square currently being scanned replacing the one that already have

(b) Move the read/write head on the tape square to left or right.



→ Tape had a left end but it extends indefinetly to the right, This could be Possible by introducing left end symbol on to the tape, such that when R/W head reads a left end symbol, it imidiatly moves to the right.

we use distinct symbol (←, →) to denote the movement of the head to the left or Right.

This two symbol are Not member of alphabet set.

A turing Machine (TM) is a quantuple having.

$(K, \Sigma, \delta, s, H)$

K = Non Empty finite set of state

Σ = is a alphabet set, containing a blank symbol ($\sqcup$) and left end symbol ($\triangleright$) but Not containing $\leftarrow \& \rightarrow$ respectively.

$s \in K$ (Initial state)

$H \subseteq K$ Is a set of halting states.

δ is a transision function

$$(K-H) \times \Sigma \longrightarrow K \times (\Sigma \cup (\leftarrow, \rightarrow))$$

Examples

Consider the TM given below

$M = (K, \Sigma, \delta, s, \{h\})$

$K = \{q_0, q_1, h\}$

$\Sigma = \{a, \sqcup, \triangleright\}$

$s = q_0$

& δ is given as

q	Σ	$\delta(q, \Sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$
q_1	$\triangleright$	$(q_1, \rightarrow)$

Check wheather the string "aaa" is accepted or Not

Sol Initial configuration : $(q_0, \triangleright aaa\sqcup) \vdash$

~~$(q_0, \triangleright aaa\sqcup) \vdash (\triangleright \sqcup aa\sqcup)$~~

~~$(q_1, \triangleright \sqcup aa$~~

$(q_0, \triangleright aaa\sqcup) \vdash (q_1, \triangleright \sqcup aa\sqcup)$

$(q_1, \triangleright \sqcup aa\sqcup) \vdash (q_0, \triangleright \sqcup a a\sqcup)$

$(q_0, \triangleright \sqcup a a\sqcup) \vdash (q_1, \triangleright \sqcup \sqcup a\sqcup)$

$(q_1, \triangleright \sqcup \sqcup a\sqcup) \vdash (q_0, \triangleright \sqcup \sqcup a\sqcup)$

$(q_0, \triangleright \sqcup \sqcup a\sqcup) \vdash (q_1, \triangleright \sqcup \sqcup \sqcup\sqcup)$

$(q_1, \triangleright \sqcup \sqcup \sqcup\sqcup) \vdash (q_0, \triangleright \sqcup \sqcup \sqcup\sqcup)$

$(q_0, \triangleright \sqcup \sqcup \sqcup\sqcup) \vdash (h, \triangleright \sqcup \sqcup \sqcup\sqcup)$

we are at halting state so string is accepted.

2## Consider the TM given

$M = (K, \Sigma, \delta, s, \{h\})$

$K = \{q_0, q_1, h\}$

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

$s = q_0$

Check whether the string "abba" accepted or Not

q	σ	$\delta(q, \sigma)$
q_0	a	(q_1, b)
q_0	b	(q_1, a)
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	$(q_0, \rightarrow)$
q_1	b	$(q_0, \rightarrow)$
q_1	$\sqcup$	$(q_0, \rightarrow)$

Initial configuration of Machine $(q_0, \triangleright \underline{a}bba\sqcup)$

$(q_0, \triangleright \underline{a}bba\sqcup) \vdash (q_1, \triangleright \underline{b}bba\sqcup)$

$(q_1, \triangleright \underline{b}bba\sqcup) \vdash (q_0, \triangleright b\underline{b}ba\sqcup)$

$(q_0, \triangleright b\underline{b}ba\sqcup) \vdash (q_1, \triangleright ba\underline{b}a\sqcup)$

$(q_1, \triangleright ba\underline{b}a\sqcup) \vdash (q_0, \triangleright baba\sqcup)$

$(q_0, \triangleright baba\sqcup) \vdash (q_1, \triangleright baaa\sqcup)$

$(q_1, \triangleright baaa\sqcup) \vdash (q_0, \triangleright baaa\sqcup)$

$(q_0, \triangleright baaa\sqcup) \vdash (q_1, \triangleright baab\sqcup)$

$(q_1, \triangleright baab\sqcup) \vdash (q_0, \triangleright baab\sqcup)$

$(q_0, \triangleright baab\sqcup) \vdash (h, \triangleright baab\sqcup)$

String Accepted.

3. # Consider the turing machine given

$M = (K, \Sigma, \delta, s, H)$

$K = (q_0, h)$

$\Sigma = (a, \triangleright, \sqcup)$

$s = q_0$

$H = \{h\}$

q	a	$\delta(q, a)$
q_0	a	$(q_0, \leftarrow)$
q_0	$\sqcup$	$(h, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$

This is the turing Machine never halting ($h = \phi$)

To Specify the turing Machine, we need to specify :-

1. The State
2. The content of the tape
3. Position of the R/W head

we choose to break the finite string into two pieces

→ one part to the left of the scanned Square (cell) including the single symbol in the scanned Square.

→ Second Part Possibly empty to the right of the scanned Square.

A configuration of a TM $M = (K, \Sigma, \delta, s, H)$ is a

$$K \times \triangleright \Sigma^* \times (\Sigma^* (\Sigma - \{\sqcup\}) \cup \{\epsilon\})$$

- K: State
- $\triangleright \Sigma^*$: Part of the left of the scanned square including the currently scanned symbol
- $(\Sigma^* (\Sigma - \{\sqcup\}) \cup \{\epsilon\})$: Part possibly empty to the right of the scanned square

$(q_0, \triangleright a b \underline{a} b \sqcup) \xrightarrow[\text{Represented as}]{\text{can be}} (q_0, \triangleright a b a, b \sqcup)$

$(h, \triangleright \sqcup \sqcup \underline{\sqcup} \sqcup a) \longrightarrow (h, \triangleright \sqcup \sqcup \sqcup, \sqcup a)$

$(q, \triangleright \sqcup a \underline{\sqcup \sqcup}) \longrightarrow (q, \triangleright \sqcup a \sqcup \sqcup, \epsilon)$

Notation for TM :-

- # Symbol writing TM $\Rightarrow M_a$ or a
- # Head Moving TM $\Rightarrow$ $\rightarrow$ or R
  $\leftarrow$ or L

NOTE :- Combining Machine to be done later :-

Start State $>$

If $\Sigma = \{a, b, \triangleright, \sqcup\}$
then

$>R \xrightarrow{a, b, \triangleright, \sqcup} R$ or $>R \xrightarrow{a,b,\triangleright,\sqcup} R$ or $>R \rightarrow R$

or $>RR$ or $>R^2$

move its head Right one square, then If that square containing 'a', 'b', 'Δ' or '␣' It move its head one square further to the Right

>R⟲␣ or >R⟲a≠␣ or >R␣

Scane its tape to the Right untill it finds a blank (␣) Character

>R⟲␣ or >R⟲a=␣ or >R␣

~~Scane~~ Find the first Non Black charactered Square to the right of the currently scanned Square.

>L⟲␣ or >L⟲a≠␣ or >R␣

Sane its tape to the left untill it finds a blank character.

>L⟲␣ or >L⟲a=␣ or >L␣

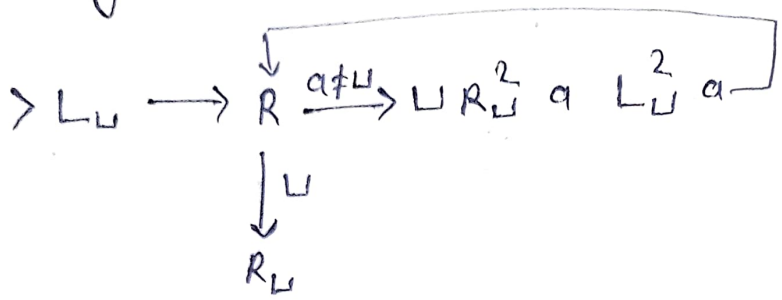
Find the first Non Blank Square to the Left of the currently Scanned square.

>R⟲␣ —a≠␣→ La

Scans to the Right untill it finds a Non Blank Square, then copy the Symbol in that Square, onto the square immidiatly to the Left of where it was found.

Examples:-

Copying Machine:-



$\sqcup w \sqcup \vdash^* \sqcup w \sqcup w \sqcup$

↑ any string

Initial configuration $\sqcup w \sqcup$

$\sqcup w \sqcup \vdash \sqcup w \sqcup \quad (\because L_\sqcup)$

$\sqcup w \sqcup \vdash \sqcup w \sqcup \quad (\because R)$

$\sqcup w \sqcup \vdash \sqcup \sqcup \sqcup \quad (\because \text{copy } w \text{ & } \because a \neq \sqcup)$

$\sqcup w \sqcup \vdash \sqcup \sqcup \sqcup \sqcup \quad (R^2_\sqcup)$

~~$\sqcup \sqcup w \sqcup$~~

$\sqcup \sqcup \sqcup \sqcup \vdash \sqcup \sqcup \sqcup w \quad (a)$ (writing symbol)

$\sqcup \sqcup \sqcup w \vdash \sqcup \sqcup \sqcup w \quad (L^2_\sqcup)$

$\sqcup \sqcup \sqcup w \vdash \sqcup w \sqcup w \quad (a)$

$\sqcup w \sqcup w \vdash \sqcup w \sqcup w \quad (R)$

~~$\sqcup w \sqcup w \vdash$~~

$\sqcup w \sqcup w \vdash \sqcup w \sqcup w \sqcup \quad (R_\sqcup)$ & $(\because \sqcup)$

Do the same thing with [w='ab']

Initial configuration ␣ ab ␣

␣ ab ␣ ⊢ ␣ a b ␣ ($\because L_\sqcup$)

␣ ab ␣ ⊢ ␣ a b ␣ ($\because R$)

␣ a b ␣ ⊢ ␣ ␣ b ␣ (copy a)

␣ ␣ b ␣ ⊢ ␣ ␣ b ␣ ␣ ($\because R_\sqcup^2$)

␣ ␣ b ␣ ␣ ⊢ ␣ ␣ b ␣ a (Paste a)

␣ ␣ b ␣ a ⊢ ␣ ␣ b ␣ a ($\because L_\sqcup^2$)

␣ ␣ b ␣ a ⊢ ␣ a b ␣ a (Paste a)

~~⊢~~

␣ a b ␣ a ⊢ ␣ ab ␣ a ($\because R$)

␣ ab ␣ a ⊢ ␣ a ␣ ␣ a (copy b)

␣ a ␣ ␣ a ⊢ ␣ a ␣ ␣ a ␣ ($R_\sqcup^2$)

␣ a ␣ ␣ a ␣ ⊢ ␣ a ␣ ␣ ab (Pasteb)

~~␣ a ␣ ␣ a b ⊢~~

␣ a ␣ ␣ ab ⊢ ␣ a ␣ ␣ a b ($L_\sqcup^2$)

␣ a ␣ ␣ ab ⊢ ␣ ab ␣ ab (Paste b)

␣ ab ␣ ab ⊢ ␣ ab ␣ ab (R)

␣ ab ␣ ab ⊢ ␣ ab ␣ ab ␣ ($R_\sqcup$)

Right shifting TM

$\sqcup w \sqcup \overset{*}{\vdash} \sqcup\sqcup w \sqcup$

where w is any string that do Not contain any blank ($\sqcup$) character

$\sqcup w \sqcup \vdash \sqcup \underline{w} \sqcup$ (L)

$\vdash \sqcup \underline{\sqcup} \sqcup$ (copy w)

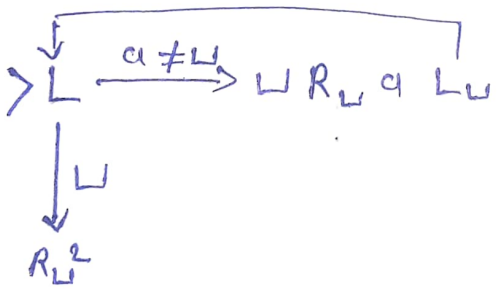
$\vdash \sqcup\sqcup \underline{\sqcup}$ ($R_\sqcup$ or R)

$\vdash \sqcup\sqcup \underline{w}$ (Paste w)

$\vdash \sqcup \underline{\sqcup} w$ ($b L_\sqcup$)

$\vdash \underline{\sqcup} \sqcup w$ (L)

$\vdash \sqcup \sqcup w \sqcup$ ($R_\sqcup^2$)



Checking for $w = 'ab'$

$\sqcup ab \sqcup \vdash \sqcup a \underline{b} \sqcup$ (L)

$\vdash \sqcup a \underline{\sqcup} \sqcup$ (copy b)

$\vdash \sqcup a \sqcup \underline{\sqcup}$ (R)

$\vdash \sqcup a \sqcup \underline{b}$ (Paste b)

$\vdash \sqcup a \underline{\sqcup} b$ ($L_\sqcup$)

$\vdash \sqcup \underline{a} \sqcup b$ (L)

$\vdash \sqcup \underline{\sqcup} \sqcup b$ (copy a)

$\vdash \sqcup \sqcup \underline{\sqcup} b$ (R)

$\vdash \sqcup \sqcup \underline{a} b$ (Paste)

$\vdash \sqcup \underline{\sqcup} a b$ ($L_\sqcup$)

$\vdash \underline{\sqcup} \sqcup a b$ (L)

$\vdash \sqcup \sqcup a b \sqcup$ ($R_\sqcup$)

check for abc

$\sqcup abc\sqcup \vdash \sqcup\sqcup abc\sqcup$ (L)

$\vdash \sqcup\sqcup ab\sqcup\sqcup$ (copy c)

$\vdash \sqcup ab\sqcup\sqcup$ ($R_\sqcup$)

$\vdash \sqcup ab\sqcup c$ (Paste c)

$\vdash \sqcup ab\sqcup c$ ($L_\sqcup$)

$\vdash \sqcup ab\sqcup c$ (L)

$\vdash \sqcup a\sqcup\sqcup c$ (copy b)

$\vdash \sqcup a\sqcup\sqcup c$ ($R_\sqcup$)

$\vdash \sqcup a\sqcup b c$ (Paste b)

$\vdash \sqcup a\sqcup bc$ ($L_\sqcup$)

$\vdash \sqcup a\sqcup bc$ (L)

$\vdash \sqcup\sqcup\sqcup bc$ (copy a)

$\vdash \sqcup\sqcup\sqcup bc$ ($R_\sqcup$)

$\vdash \sqcup\sqcup abc$ (Paste a)

$\vdash \sqcup\sqcup abc$ ($L_\sqcup$)

$\vdash \sqcup\sqcup abc$ (L)

$\vdash \sqcup\sqcup abc\sqcup$ ($R^2_\sqcup$)

left shifting TM

$\sqcup ab \sqcup \vdash^{*} ab \sqcup$

$\sqcup ab \sqcup \vdash \underline{\sqcup} ab \sqcup$ ($L_{\sqcup}$)

$\vdash \sqcup \underline{a} b \sqcup$ (R)

$\vdash \sqcup \sqcup \underline{b} \sqcup$ (copy a)

$\vdash \underline{\sqcup} \sqcup b \sqcup$ ($L_{\sqcup}$)

$\vdash \boxed{\ } \underline{a} \sqcup b \sqcup$ (Paste a)

$\vdash \underline{a} \sqcup b \sqcup$ ($R_{\sqcup}$)

$\vdash a \sqcup \underline{b} \sqcup$ (R)

$\vdash a \sqcup \underline{\sqcup} \sqcup$ (copy b)

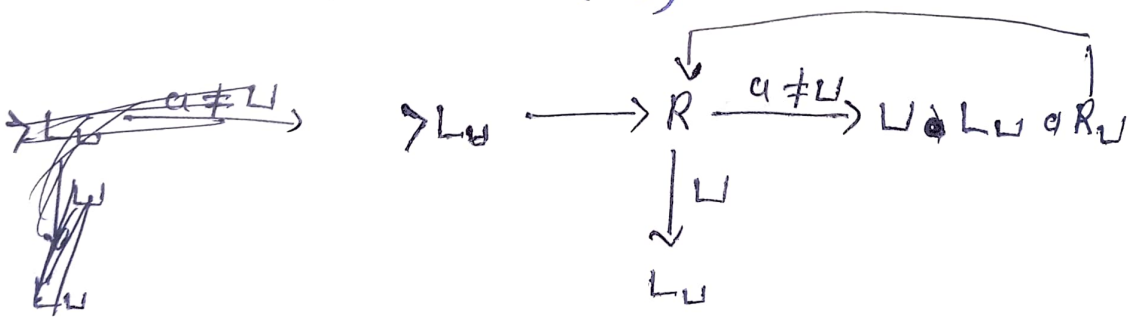
$\vdash a \underline{\sqcup} \sqcup \sqcup$ ($L_{\sqcup}$)

$\vdash a \underline{b} \sqcup \sqcup$ (Paste b)

$\vdash ab \underline{\sqcup} \sqcup$ ($R_{\sqcup}$)

$\vdash ab \sqcup \underline{\sqcup}$ (R)

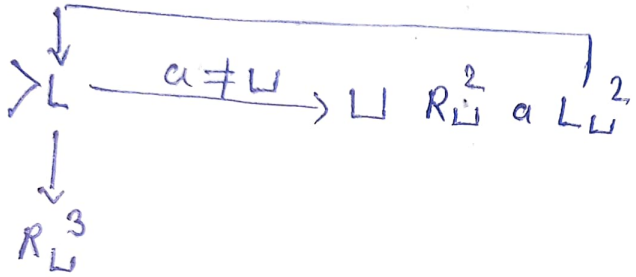
$\vdash ab \underline{\sqcup}$ ($L_{\sqcup}$)



Construct TM which shift 2 time Right

NOTE :- No Need to construct another machine we can modify the (Left ~~Shift~~ Right machine) By Increasing the power of "$R_\sqcup$" & "$L_\sqcup$"

we are going to increasing the Power 2 time.



Target $\boxed{\sqcup w \sqcup \vdash \sqcup\sqcup\sqcup w \sqcup}$

Solve

$\sqcup w \sqcup \vdash \sqcup \underline{w} \sqcup$

$\vdash \sqcup \sqcup \sqcup$

$\vdash \sqcup \sqcup \sqcup \sqcup$

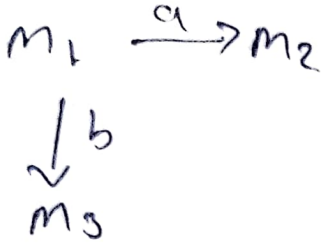
$\vdash \sqcup \sqcup \sqcup w$

$\vdash \sqcup \sqcup \sqcup w$

$\vdash \sqcup \sqcup \sqcup w$

$\vdash \sqcup \sqcup \sqcup w \sqcup$

Combining Machine :-



(i) Start in the initial state of M_1

(ii) Operate as M_1 until it halts

(iii) If the current scanned symbol is

a → (i) Initiate M_2 (ii) operate as M_2

b → (i) Initiate M_3 (ii) operates as M_3

$M_1 = (K_1, \Sigma, \delta_1, s_1, H_1)$
$M_2 = (K_2, \Sigma, \delta_2, s_2, H_2)$
$M_3 = (K_3, \Sigma, \delta_3, s_3, H_3)$
all are disjointented

$\Downarrow$

$M = (K, \Sigma, \delta, s, H)$ combining machine.

where $K = K_1 \cup K_2 \cup K_3$

$s = s_1$

$H = H_2 \cup H_3$

for each $a \in \Sigma$ & $q \in K - H$, $\delta(q, a)$ is defined as follow :-

(a) if $q \in K_1 - H_1$, then $\delta(q, a) = \delta_1(q, a)$

(b) if $q \in K_2 - H_2$, then $\delta(q, a) = \delta_2(q, a)$

(c) if $q \in K_3 - H_3$, then $\delta(q, a) = \delta_3(q, a)$

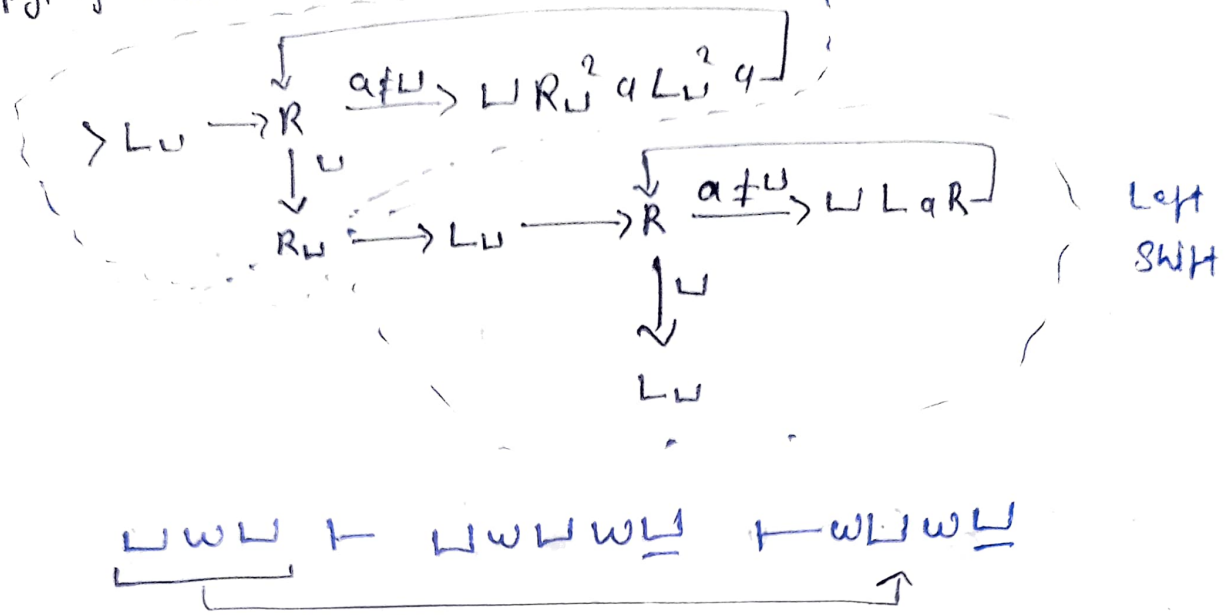
(d) If $q \in H_1$

then

$\delta(q, a) = s_2$ and $\delta(q, a) \in H$ $a \neq a$

$\delta(q, b) = s_3$ otherwise $a \neq b$

Copying Machine



$\sqcup w \sqcup \vdash \sqcup w \sqcup w \sqcup \vdash w \sqcup w \sqcup$

Computing with TM :-

The input string with no blank symbol in it, has a left most symbol $\triangleright$, with a blank to the left & blank to the Right.

~~$(s, \triangleright, \sqcup)$~~ $(s, \triangleright \sqcup w)$

Let $M = (K, \Sigma, \delta, s, H)$ be a TM such that

$H = \{y, n\}$

yes (w is accepted) NO (w is rejected)

(1) if $(s, \triangleright \sqcup w)$ yields an accepting configuration then M accepts an input w

(2) if $(s, \triangleright \sqcup w)$ yields an ~~a~~ Rejecting configuration then M Rejects an input w

This is called Decidable Machine or Turing Acceptable Machine.

* M decides a language $L \subseteq \Sigma_0^*$ where

$$\Sigma_0 \in \Sigma - \{\sqcup, \triangleright\}$$

if for ∀ string $w \in \Sigma_0^*$ the following is true:

(i) if $w \in L$ then M accepts w

(ii) if $w \in L$ then M Rejects w

(Language) L: **Recursive language**

कोई भी Machine उस Language के सभी w को accept करती है या Reject करती है।

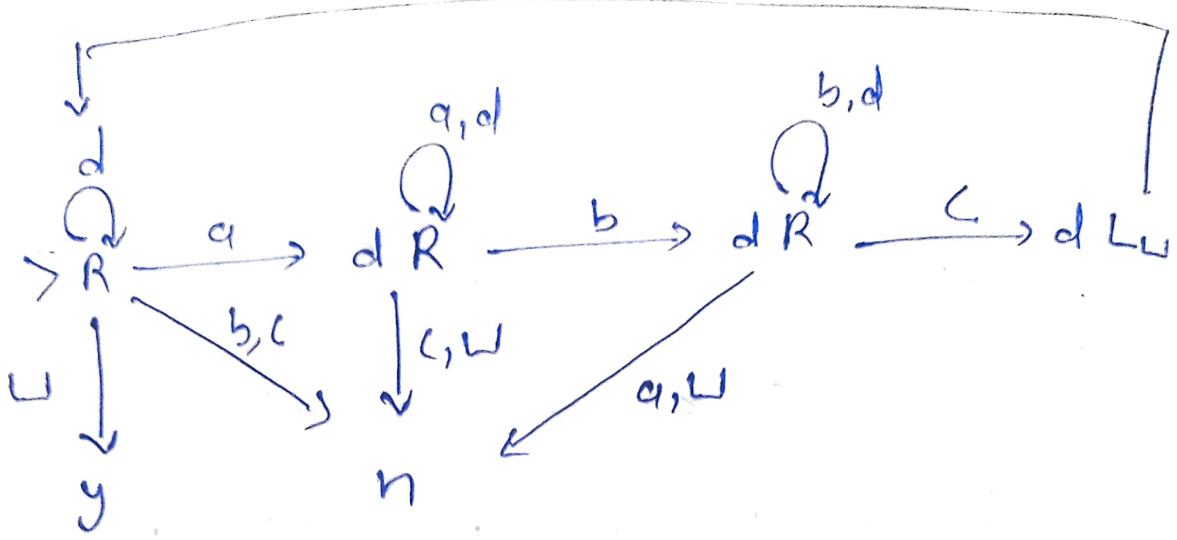
$L = \{ a^n b^n c^n : n \geq 0 \}$ Non CFL Language.

↓

we have to construct turing Decidable machine for this language Because it is Not accepted By PDA.

a a b b c c ⟶ "N"

a a b b c c ⟶ 'y'



Ex:-

$\triangleright$ ␣ a a b b c c

$\vdash$ $\triangleright$ ␣ a a a b b c c

$\vdash$ $\triangleright$ ␣ d a b b c c

$\vdash$ $\triangleright$ ␣ d a a b b c c

$\vdash$ $\triangleright$ ␣ d a b b b c c

$\vdash$ $\triangleright$ ␣ d a d b c c

$\vdash$ $\triangleright$ ␣ d a d b c c

$\vdash$ $\triangleright$ ␣ d a d b c c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d a d b d c

$\vdash$ $\triangleright$ ␣ d d d b d c

$\vdash$ $\triangleright$ ␣ d d d b d c

$\vdash$ $\triangleright$ ␣ d d d d d c

$\vdash$ $\triangleright$ ␣ d d d d d c

$\vdash$ $\triangleright$ ␣ d d d d d c

$\vdash$ $\triangleright$ ␣ d d d d d d

Automata theory

Design a TM for

$$f : \Sigma_0^* \longrightarrow \Sigma_0^*$$

where $f(w) = ww^R$

that is

$$q_0, \triangleright \sqcup abb \sqcup \vdash h, \triangleright \sqcup abba \sqcup$$

Let f be any function from Σ_0^* to Σ_0^*

$\Sigma_0 = \Sigma - \{\sqcup, \triangleright\}$

$\omega \in L$ } halt
$\omega \notin L$ }

we say that, M computes function f, if

$\forall\ \omega \in \Sigma_0^* \quad M(\omega) = f(\omega)$

that is, M halts on input ω

A function f is called recursive, if there is a TM M that compute f

recursive function.

Recursive Enumerable language :-

Let $M = (K, \Sigma, \delta, s, H)$ be a TM

let $\Sigma_0 \subseteq \Sigma - \{\sqcup, \triangleright\}$ be an alphabet

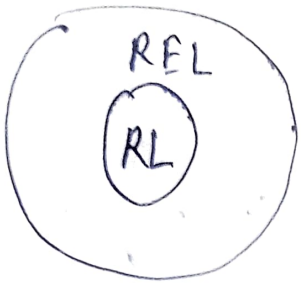
& $L \subseteq \Sigma_0^*$ be a language

we say that, M semidecides L if for any string $\omega \in \Sigma_0^*$ the following is true

(1) $\omega \in L$ if M halt on ω

(2) $\omega \in L$ M never enter a halting state. i.e machine continous its computation continously

L is Recursive Enumerable languages



REL: Recursive Enumerable language
RL : Recursive language (Decidable)
{Y, N}

REL = Semi decidable Machine
{Y}
w∈L
M halt
w∉L

Question :- Prove that if L is recursive language then its compliment $\bar{L}$ is also recursive.

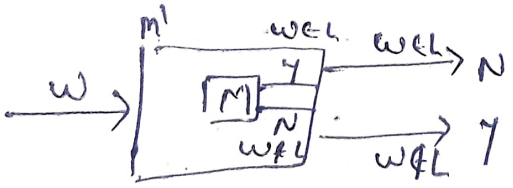
That is Recursive languages are closed under complimentation.

Ans :-

Let M = (K, Σ, δ, s, {Y, N})
is a TM that decides L then there must exist a TM M' which will be

M' = (K, Σ, δ', s, {N, Y})

that Must also decides L



M & M' are identical, except that it reverses the role of Y & N

$$\delta'(q,a) = \begin{cases} n & \text{if } \delta(q,a) = y \\ y & \text{if } \delta(q,a) = N \\ \delta(q,a) & \text{otherwise} \end{cases}$$

$M'(w) = y$ iff $M(w) = N$ and therefore M' decides $\overline{L}$

So we can say Recursive Language are closed under complimentation.

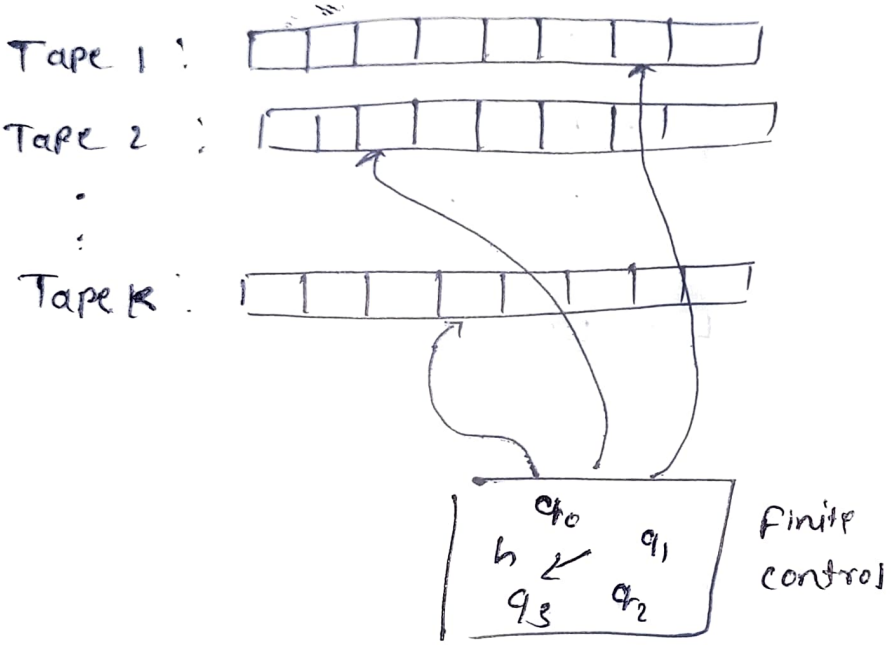
← multitape turing machine

Multitape TM

Let $M = (K, \Sigma, \delta, s, H)$ be a

k-tape TM. A configuration of M is a Member of

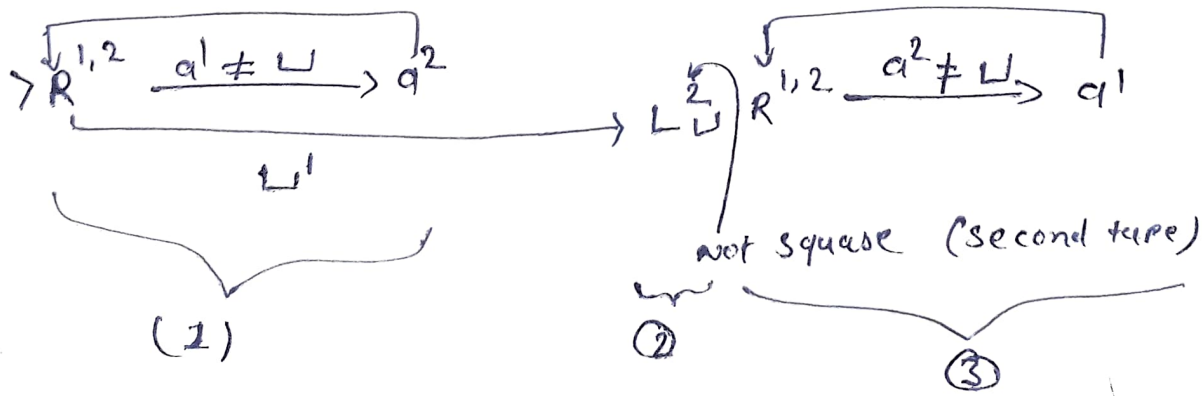
$$K \times \triangleright \Sigma^* \times (\Sigma^*(\Sigma - \{\sqcup\} \cup \{\epsilon\}))^k$$



Q :- Design a 2-tape TM for the copying Machine. to accomplish the task of tranforming

$\triangleright \sqcup w \sqcup$ into $\triangleright \sqcup w \sqcup w \sqcup$

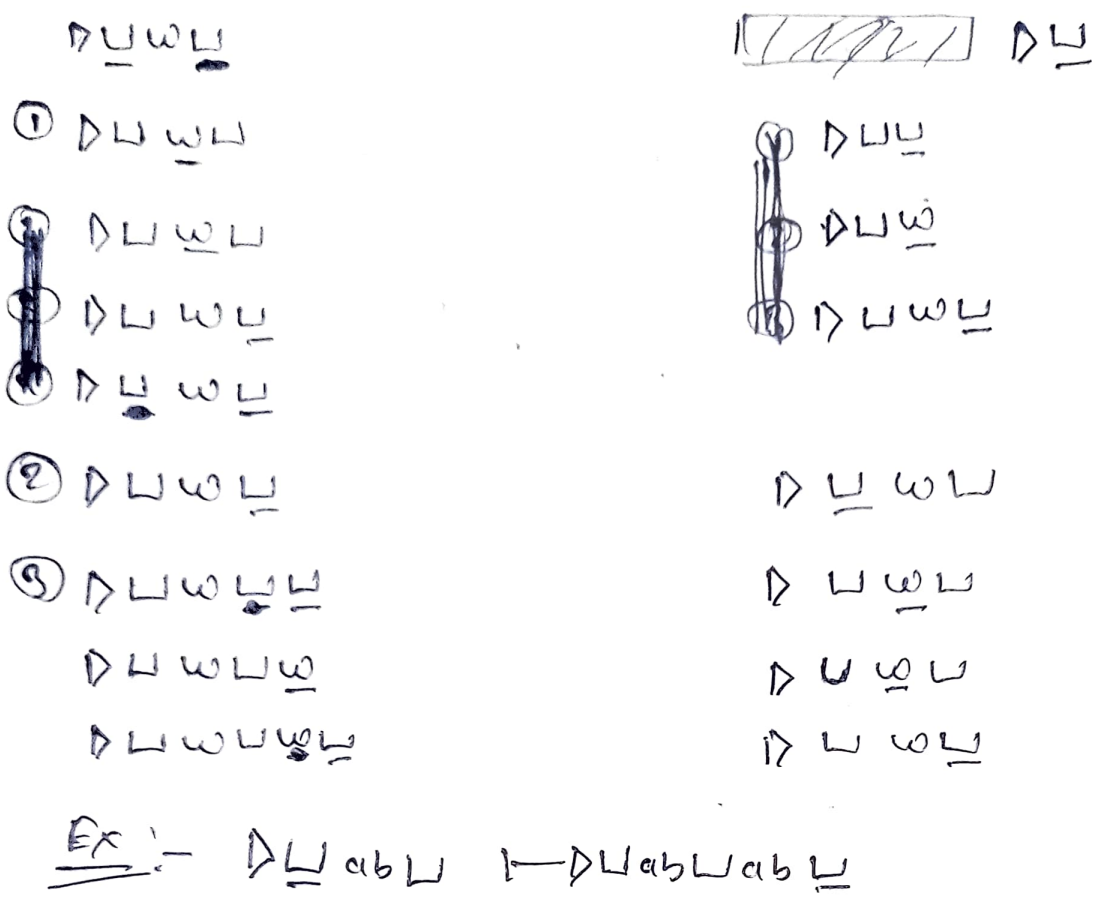
where $w \in \{a, b\}^*$



1st tape

2nd tape

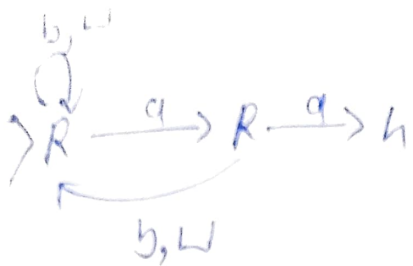
(initial config)



Ex :- ▷␣ab␣ ⊢ ▷␣ab␣ab␣

Date :- 25/12/24

Ex.6 # Design a TM that scans to the right until it finds two consecutive a's and then halts.



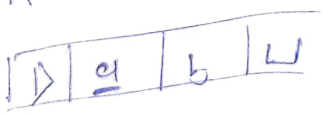
$\Sigma = \{a, b, \triangleright, \sqcup\}$

baab

Then halt, ans

Ex.9 # Do the machines LR and RL always accomplish the same thing.?

for LR



(∵ L)

▷ a b ⊔

▷ a b ⊔

▷ a b ⊔ (∵ R)

for RL

▷ a b ⊔

▷ a b ⊔ (∵ R)

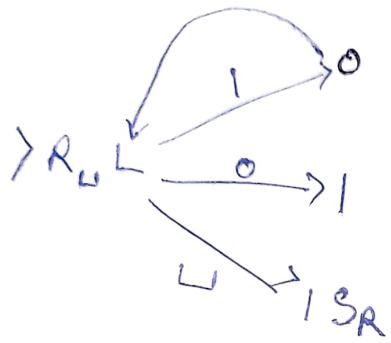
▷ a b ⊔ (∵ L)

Not Proceed to the same thing

$Succ(n) : n+1$ (Successor of n could be $n+1$)

Ex:

$Succ(\triangleright \underline{1}011\sqcup) \vdash \triangleright 1100\sqcup$



S_R : Right shifting TM

$\triangleright 1011\sqcup \vdash \triangleright \underline{1}011\sqcup$

$\vdash \triangleright 10\underline{1}1\sqcup$

$\vdash \triangleright 10\underline{1}0\sqcup$ (copy 1)

$\vdash \triangleright 10\underline{1}0\sqcup$

$\vdash \triangleright 10\underline{0}0\sqcup$ (copy 1)

$\vdash \triangleright 1\underline{0}00\sqcup$

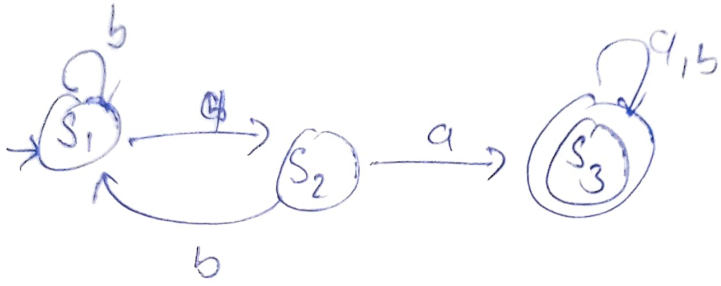
$\vdash \triangleright \underline{1}100\sqcup$

Ex:- ▷ | | | | | ⊔ ⊢ ▷

▷ ⊔ | | | | ⊔ ⊢ ▷ | | | ⊔ ⊔ ⊔

Marking unambigous CFG.

For 'aa' as substring.



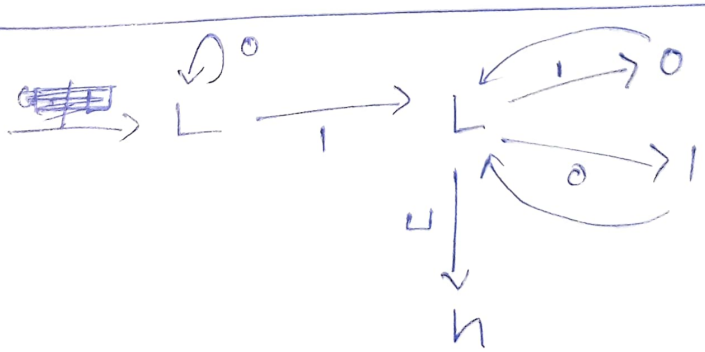
FA to CFG.

$S_1 \rightarrow bS_1 \mid aS_2$

$S_2 \rightarrow aS_3 \mid bS_1$

$S_3 \rightarrow aS_3 \mid bS_3 \mid \epsilon$

#



▷ 0 | | 0 ⊔ ⊢ ▷ 0 | | 0 ⊔

2's compliment

⊢ ▷ 0 | | 0 ⊔

⊢ ▷ 0 | | 0 ⊔

⊢ ▷ 0 | 1 0 ⊔

⊢ ▷ 0 0 | 0 ⊔

⊢ ▷ 0 0 | 0 ⊔

⊢ ▷ 1 0 | 0 ⊔

Turing Decidable :-

A language L is Decidable if there exist a TM such that for ∀ strings w

if w ∈ L, M enteres 'y'
if w ∉ L, M enters 'n'

Alonzo Church
↳ Alan turing

function which are algorithmically computable that has a TM.

26/11/24

Constructing a Universal turing machine :-

The limitation of turing machine is that they are Hardwired that is they specialized to solve one particular Problem.

Since the real computer are reprogrameble, so we need a generic TM. which should be programmed. in a manner a general purpose computer is Programmed.

Universal TM simulates & any TM.

→ Inputs of UTM are :-

(1) Description of the transition of M

(2) Input string of M

→ Convention :-

(1) A string representing a TM state is of form $(\{q\}, \{0,1\}^*)$ i.e. a letter q following by a binary string.

(2) A tape symbol is always represented as a string $(\{a\}\ \{0,1\}^*)$

Let $M = (K, \Sigma, \delta, s, H)$ be a TM & Let i & j are 2 smallest integer such that $2^i \geq |K|$

$2^j \geq |\Sigma| + 2$

$\{\leftarrow, \rightarrow\}$

$\sqcup = q0^j$

$\rhd = a\,0^{j-1}\,1$

$\leftarrow = a\,0^{j-2}\,10$

$\rightarrow = a\,0^{j-2}\,11$

j bits

Ch.5 { Lewis Papadimitriou }

Start state $q\,0^j$

The sequence of strings of the form

(q, a, P, b) with q & P representing state & a & b representing symbols.

Example for UTM:- 5.2

Que Consider a TM $M = (K, \Sigma, \delta, s, \{h\})$

where $K = \{s, q, h\}$

$\Sigma = \{\sqcup, \rhd, a\}$

δ is given below

State	symbol	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\rhd$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\rhd$	$(q, \rightarrow)$

Consider a TM give the Discription / Encoding of this machine in form of UTM.

Solve

$K = \{S, q, h\}$

$\Sigma = \{\sqcup, \triangleright, a\}$

$2^i \geq |K|$ Hence $i = 2$

$2^j \geq |\Sigma| + 2$ Hence $j = 3$

State / Symbol	Representation
S	$q\ 00$
q	$q\ 01$
h	$q\ 11$ ← Higher Priority Halting State.
$\sqcup$	$a\ 000$
$\triangleright$	$a\ 001$
$\leftarrow$	$a\ 010$
$\rightarrow$	$a\ 011$
a	$a\ 100$

① $\delta(S, a, q, \sqcup) \rightarrow q00\ a100\ q01\ a000$

② $(S, \sqcup, h, \sqcup) \rightarrow q00\ a000\ q11, a000$

③

④

⑤

⑥

Therefore the discription of TM is given below ① ② ③ ④ ⑤ ⑥ . . . .

q_{00} a_{100} q_{01} a_{000} q_{00} a_{000} q_{11} a_{000}

Halting Problem :- Suppose that you have written a Program (in any language)

It takes as input any Program 'P', written in the Same language & an input 'n' of that Program. By Some clever analysis your Program always determines correctly whether the Program is halting on input 'n' or whether it would run forever. You have Named this Program halts (P, n).

(Undecidability)