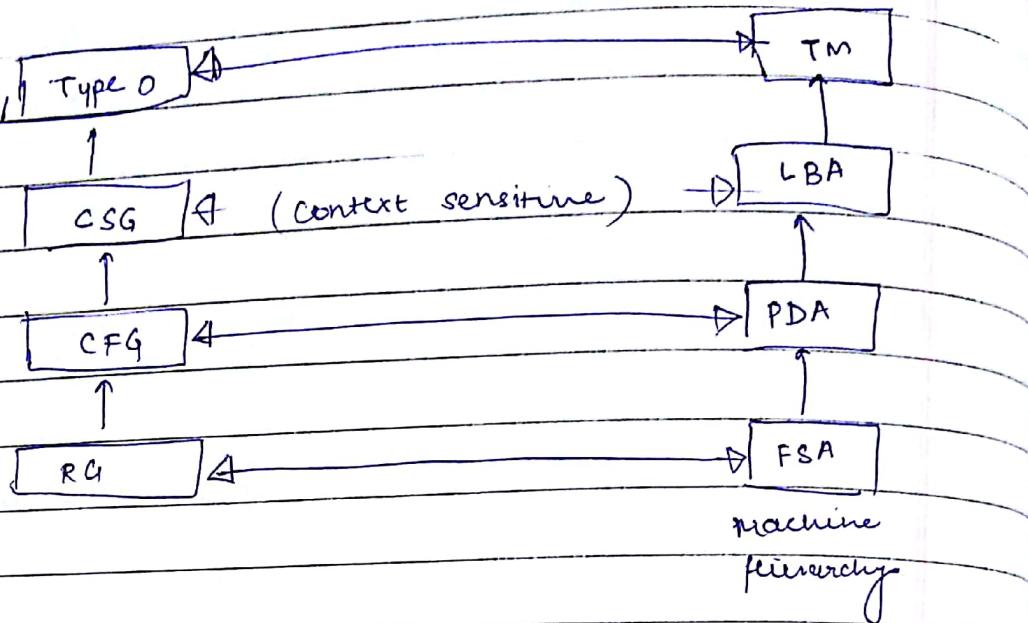
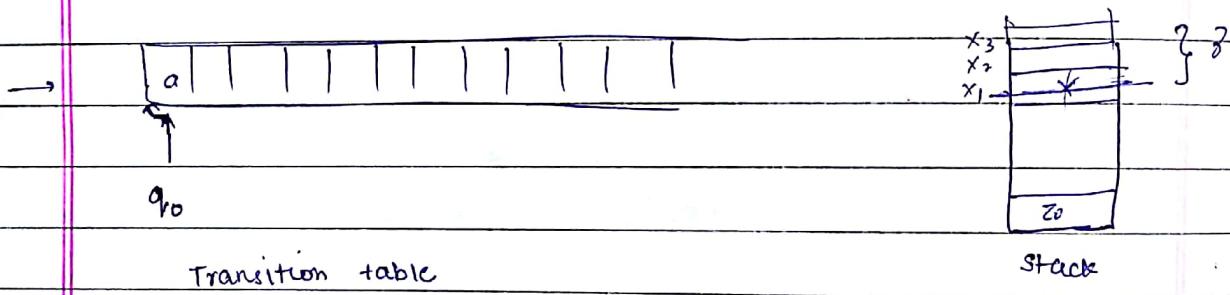


PUSHDOWN AUTOMATA

Grammar hierarchy:



Chomskian Hierarchy
of languages



not only sees 'a' but also top of stack (symbol at top of stack)
(i/p symbol)

Transition :

$$FSA: \quad s(q_i, a) = q_j$$

$$PDA: \quad s(q_i, a, x) = (q_j, y)$$

x can be replaced by

sequence of symbols called $\gamma = x_1 x_2 x_3 \dots x_k$

\downarrow
can be empty (ϵ)

$\left\{ \begin{array}{l} x \text{ is deleted } \\ x \text{ is popped up} \end{array} \right\}$

* Automata: finite representation of an infinite no. of things.

$a^n b^n$: not accepted by ϵ FA;

↳ don't have memory, don't know count of 'a' occurred

↳ can't move to b till a is completed.



Possible in case of Pushdown Automata.

→ can store a in stack & whenever get a 'b', pop out 1 'a' out of the stack.

* everything FA is a PDA (without any use of stack)

$\{a^n b^n ; n \geq 1\}$: accepted by PDA

$\{a^n b^n c^n ; n \geq 1\}$: not " "

Q. Every finite \Rightarrow regular subset of Non-regular set is regular.

Yes.

Q. Let L be a RL. What can you say about L^R ?

Yes

Q. Union of RL is RL

Yes

Q. Infinite Union of Regular set is Regular

No.

Teacher's Signature

12/3/18

PDAIntroⁿ to Automata Theory - Hopcroft, Ullman, MotwaniDFA = NDFA = ϵ -NDFA

(can convert from either one to anyone)

Deterministic PDA \rightarrow DPDA $<$ NPDA (Power)

stack starting stack alphabet

PDA : $(Q, \Sigma, \Gamma, S, q_0, Z_0, F)$ $Q \rightarrow$ finite set of all states $\Sigma \rightarrow$ finite set of i/p symbols / alphabets $\Gamma \rightarrow$ finite set of stack alphabets $q_0 \rightarrow$ Initial state $Z_0 \rightarrow$ start stack symbol $F \rightarrow$ set of all final statesset of all subsets of $Q \times \Gamma^*$ NPDA : $S \rightarrow S : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^* [2^{Q \times \Gamma^*}]$ (go to multiple states $\Rightarrow 2^m$)
(subset)string of symbols over Γ

2 ways of accepting automata?

→ Acceptance by final states

→ Acceptance by empty stack : whenever stack is empty
& you've read all the whole i/p string.ID of a PDA :↓
instantaneous descrpⁿ $(q, w, x) / (q, w, \emptyset)$

current state i/p string

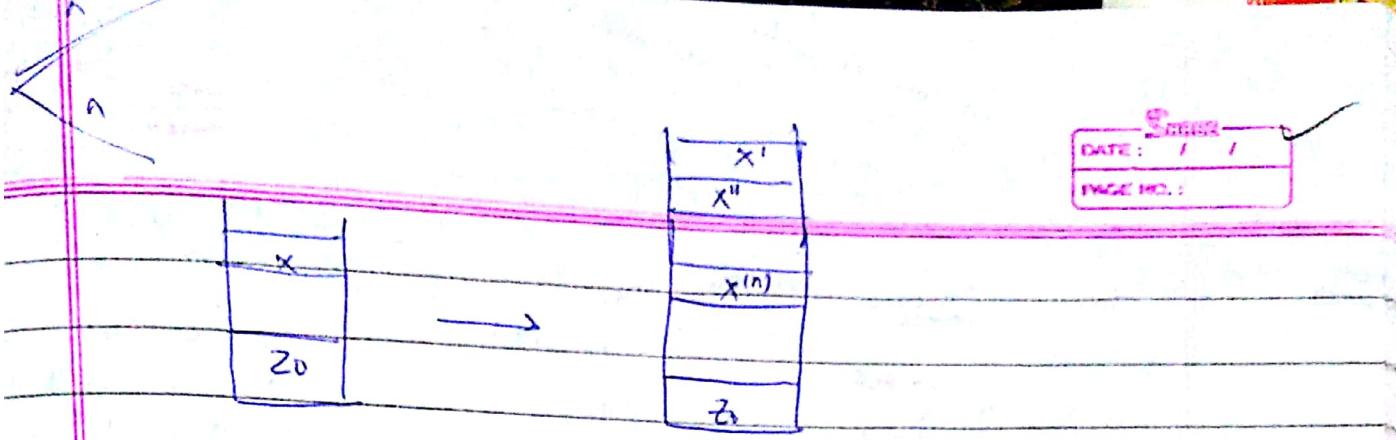
top of stack

 $S : (q, aW, x) \vdash (q', W', x')$

because it's non-deterministic

 $S(q, a, x) \text{ includes } (q', x)$ string of symbols $q \text{ goes to } q'$ $aW \text{ goes to } W' \Rightarrow$ no need to show x $x = x' x'' x''' \dots x^{(n)}$ going to be top of stack

Teacher's Signature



DATE: / /
PAGE NO.:

$\alpha \in \Gamma^*$

if $\alpha = \lambda \Rightarrow$ just popping out x

$\alpha = x_i \Rightarrow$ replacing x by x_i

① Acceptance by final state :

(q_0, w, z_0)

starting ID

$w = \underline{a^n b^n}$

$L = \{ w \mid w \in \Sigma^* \text{ and } (q_0, w, z_0) \vdash^* (q_f, \epsilon, x) \}$

where $q_f \in F$
 $x \in \Gamma$

can be
any symbol
in stack

② Acceptance by Empty store :

$L = \{ w \mid w \in \Sigma^* \text{ and } (q_0, w, z_0) \vdash^* (q, \epsilon, \epsilon) \}$

$0 \vdash^* 0$

$\rightarrow L = \{ ww^R \mid w \in \{0,1\}^* \}$

$S(q_0, 0, z_0)$ includes $(q_0, 0z_0) \Rightarrow$ pushing 0 in z_0

need to store

0 because
have to check
pattern

$S(q_0, 1, z_0)$ includes $(q_0, 1z_0)$

Teacher's Signature

at any point of time, it may say now, check for is final. Now, check for WR \Rightarrow go to new state q₁ & RD change in stack final state

DATE: / /
PAGE NO.:

To accept all string $\leftarrow \{ S(q_0, \epsilon, z_0) \text{ includes } (q_0, z_0) \} \rightarrow \text{Non-deterministic move}$

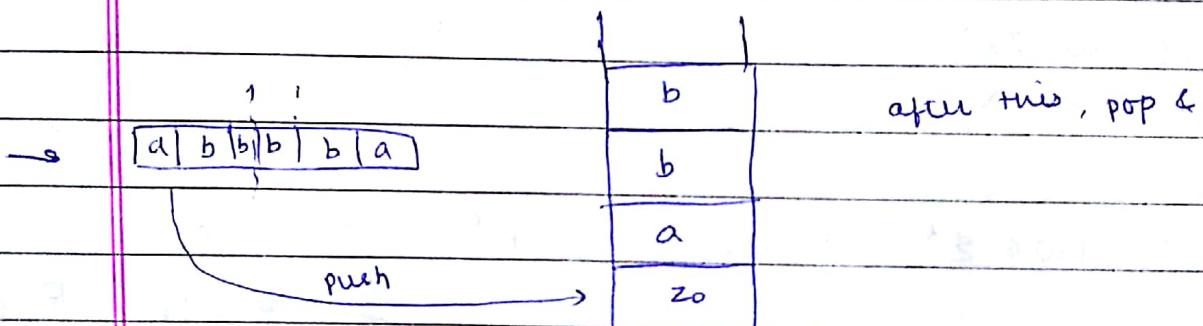
$S(q_0, \epsilon, z_0)$	$\xrightarrow{\text{tos}}$	(q_0, z_0)
$S(q_0, 0, 0)$	"	$(q_0, 00)$
$S(q_0, 0, 1)$	$\xrightarrow{\text{tos}}$	$(q_0, 01)$
$S(q_0, 1, 0)$	$\xrightarrow{\text{tos}}$	$(q_0, 10)$
$S(q_0, 1, 1)$	$\xrightarrow{\text{tos}}$	$(q_0, 11)$
$S(q_0, \epsilon, 011)$	"	$(q_1, 011)$
$S(q_1, 0, 0)$	"	(q_1, ϵ)
$S(q_1, 1, 1)$	"	(q_1, ϵ)
$S(q_1, \epsilon, z_0)$	"	(q_2, z_0)

at any time u may say w is finishing so reach to q₁

reverse: (WR) $\xrightarrow{\text{tos}}$ ① 0 1

pushing ϵ in stack If have 0, tos will also be 0

final state accept it



q₀ : for reading w

when checking for WR : go to q₁

From q₁ : check if string & tos are same. If so, pop the tos.

→ Note: If a string is accepted by PDA, there will be at least 1 path to reach final state.

→ At any moment, it has equal probability to either remain on same state or move to next state

$$Q = \{q_0, q_1, q_2\}$$

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

$$F = \{q_2\}$$

$$\Gamma = \{Z_0, 0, 1\}$$

Eg.

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

→ here, we can find when a will comes and b will come (with each other)

Order dhyaaan

when reads a → push

rakhna \Rightarrow b aane pe state change

when reads b → pop 1 a with 1 b

$a, Z_0/a_0$



a
c
Z ₀

$$s(q_0, a, Z_0) \Rightarrow s(q_0, a_0)$$

$$s(q_0, a, a) \Rightarrow s(q_0, aa)$$

$$s(q_0, b, a) \Rightarrow s(q_1, \epsilon)$$

$$s(q_1, b, a) \Rightarrow s(q_1, \epsilon)$$

$$s(q_1, \epsilon, Z_0) \rightarrow s(q_1, \epsilon) \quad s(q_1, \epsilon, Z_0) \rightarrow (q_f, \epsilon, Z_0)$$

Acceptance by empty store

$$1.) L = \{a^i b^j c^k \mid i, j, k \geq 1, i+j=k\}$$

$$2.) L = \{a^i b^j c^k \mid i, j, k \geq 1, i+k=j\}$$

$$* 3.) L = \{a^m b^n \mid m, n \geq 1, m \leq n \leq 2m\}$$

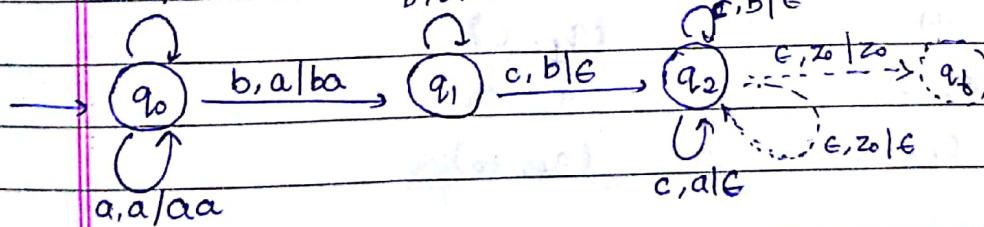
$$4.) L = \{a^n b^n \mid n \geq 1\}$$

$$1.) L = \{a^i b^j c^k \mid i, j, k \geq 1, i+j=k\}$$

$a, Z_0/a_0$

$b, bb/bb$

$c, b/c$



$$s(q_0, a, Z_0) \rightarrow s(q_0, a_0)$$

$$s(q_2, c, b) \rightarrow s(q_2, c)$$

$$s(q_0, a, a) \rightarrow (q_0, aa)$$

$$s(q_2, c, a) \rightarrow (q_2, \epsilon)$$

$$s(q_0, b, a) \rightarrow (q_1, ba)$$

$$s(q_1, b, b) \rightarrow (q_1, bb)$$

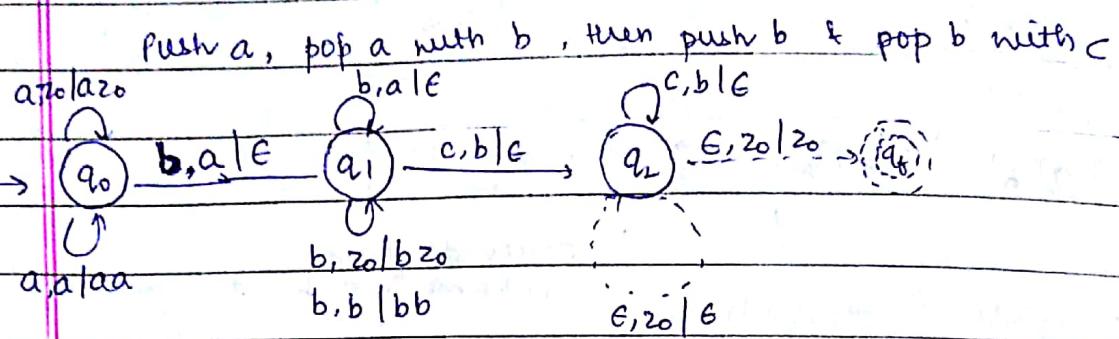
$$s(q_1, c, b) \rightarrow (q_2, \epsilon)$$

$$s(q_2, \epsilon, Z_0) \rightarrow (q_2, \epsilon, \epsilon)$$

$$s(q_2, \epsilon, Z_0) \rightarrow (q_f, \epsilon, \epsilon)$$

Teacher's Signature

$$2) L = \{a^i b^j c^k \mid i, j, k \geq 1, i+k=j\}$$



$s(q_0, a, z_0)$ includes $(q_0, a z_0)$

$s(q_0, a, a)$ " $(q_0, a a)$

$s(q_0, b, a)$ " (q_1, a)

$s(q_1, b, a)$ " (q_1, a)

$s(q_1, b, z_0)$ " $(q_1, b z_0)$

$s(q_1, b, b)$ " $(q_1, b b)$

$s(q_1, c, b)$ " (q_2, b)

$s(q_2, c, b)$ " (q_2, b)

~~so~~

$s(q_2, \epsilon, z_0) \rightarrow (q_f, \epsilon, z_0)$ $s(q_2, \epsilon, z_0) \rightarrow (q_s, \epsilon)$

3.) Push a, read b \Rightarrow pop a or read 2 b & pop 1 a.

$s(q_0, a, z_0)$ includes $(q_0, a z_0)$

$s(q_0, a, a)$ " $(q_0, a a)$

1st choice

$s(q_0, b, a)$ " $\{(q_1, \epsilon), (q_2, a)\}$

~~so~~

pop a, " won't pop a

$s(q_1, b, a)$ " $\{(q_1, \epsilon), (q_2, a)\}$

$s(q_2, b, a)$ " (q_1, ϵ)

$s(q_1, \epsilon, z_0)$ " (q_f, z_0)

$s(q_2, \epsilon, z_0)$ " (q_f, z_0)

$q_1 \rightarrow$ for every b : popping ~~a~~ a

$q_2 \rightarrow$ skipping a for 1 b

1) 2)

→ showing Acceptance by Final states \equiv Acceptance by Empty store

a) $1 \Rightarrow 2$

Let : PDA. A = $(Q, \Sigma, \Gamma, S, q_0, z_0, F)$
with acceptance by final state

(Want to construct acceptance by empty store for A)

Let $L = L(A)$

We need to construct A' (new PDA) s.t.

$L = L(A')$ [acceptance by empty store]

$A' = (Q', \Sigma, \Gamma, S', q_0, z_0, F)$

+ Once we go to z_0 & S' , we can empty the stack for "

$$F = \{q_1, q_2, \dots, q_n\}$$

so that

S' should include all transitions of S in addⁿ to that we've the following

$\forall q_i \in F \quad \nexists S'(q_i, \epsilon, x) \rightarrow$ includes (q_{empty}, x) : go to state q_{empty} & empty all the stack

$S'(q_{empty}, \epsilon, x)$ includes (q_{empty}, ϵ) → stack empty
Acceptance by empty store

By construction, we see that

$$Q' = Q \cup \{q_{empty}\}$$

$$L(A') = L(A)$$

$$+ \{q_{empty} \rightarrow q_{empty}\}$$

$$+ \{q \in Q \mid \exists x \in \Sigma^*, q \xrightarrow{\delta, x} q_{empty}\}$$

b) $2 \Rightarrow 1$

We have : $B = (Q', \Sigma, \Gamma, S, q_0, z_0, \emptyset)$ \rightarrow final state take parameter

B is accepting by empty store.

Let $L = L(B)$

We need to construct B' (new PDA) s.t.

$$L = L(B')$$

Teacher's Signature

$$T' = T \cup \{z_0'\}$$

$$Q' = Q \cup \{q_0', q_{\text{final}}\}$$

$$B' = (\alpha', \Sigma, T', S', q_0', z_0', F) \quad (1a)$$

B' accepts by final state.

- * when stack becomes empty, make it much to F.s.

when start processing for B' , put z_0' in stack
 z_0' is initial symbol

$$S'(q_0', \epsilon, z_0') \text{ includes } S(q_0, z_0 z_0')$$

from q_0 , we can apply all transitions of S .

$$S(q, \epsilon, z_0') \text{ includes } (q_{\text{final}}, z_0') \quad F = q_{\text{final}}$$

$q \in Q'$ ↓
if B ka stack
is empty $\rightarrow q_0$ to final
stack

* z_0' included to check when B ka stack becomes empty.
moreover

* S' includes all transitions of S .

→ By construction,

$$L(B) = L(B')$$

$$\text{eg. } L = \{ w \in \{a, b\}^* \mid n_a(w) = n_b(w) \}$$

$$L = \{ w \in \{a, b\}^* \mid n_a(w) = n_b(w) \} \rightarrow \text{Dyck set}$$

for any prefix w' of w
 $n_a(w') \geq n_b(w')$ } (Dyck language)

→ $\begin{array}{ccccccc} a & b & a & b & a & b \\ (&) & (&) & (&) &) \end{array} : \text{well formed parenthesis}$

$\begin{array}{ccccccc} (& (&) &) &) & & \\ a & a & b & b & b & & \end{array} : \text{not well formed}$

$\begin{array}{ccccccc}) &) &) & (& C & C & \\ b & b & b & a & a & a & \end{array} : \text{not well formed}$

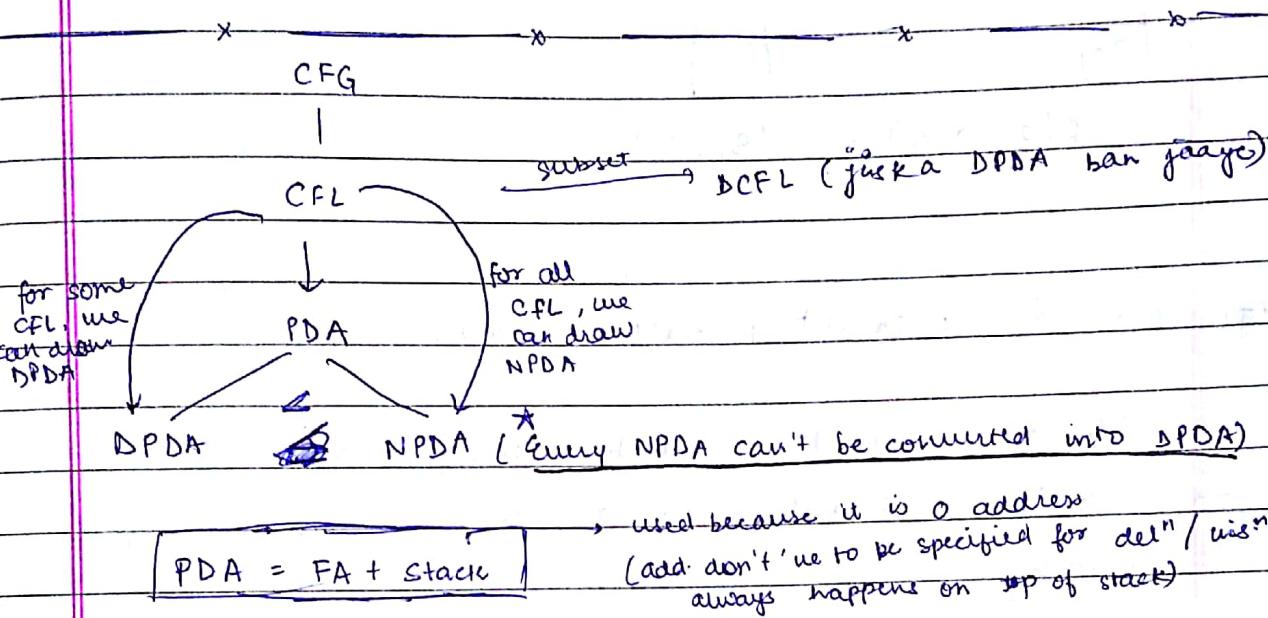
no. of open brackets $>$ no. of closed brackets
in prefix of string

Teacher's Signature

* PDA ban skta h \Rightarrow CFL h
Otherwise, nhi h

DATE:	/ /
PAGE NO.:	

→ whenever see a \rightarrow push in stack
b \rightarrow pop out of stack



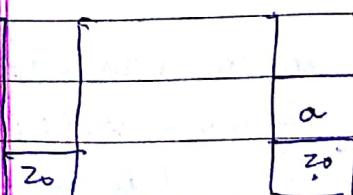
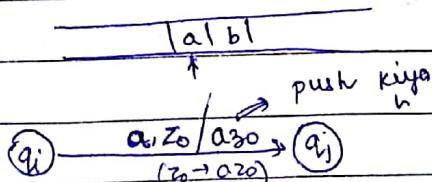
Transition func'

1) DPDA : $s : Q \times \Sigma \times \Gamma \xrightarrow{\text{toS}} Q \times \Gamma^*$

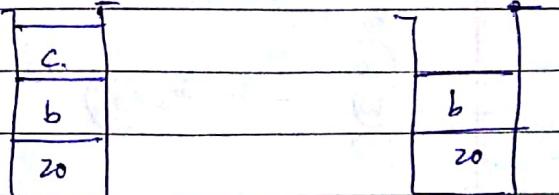
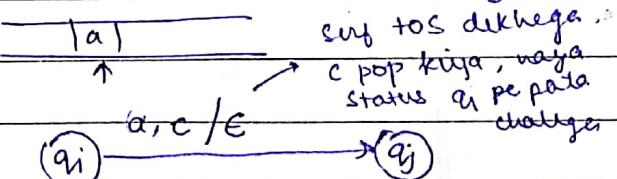
2) NPDA $s : Q \times \Sigma \times \Gamma \xrightarrow{\text{2}} (\Omega \times \Gamma^*)$

Basic Ops on Stack :-

① PUSH



② POP

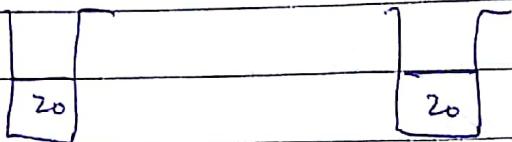
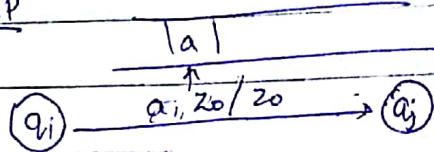


$$s(q_i, a, z_0) \rightarrow (q_j, a z_0)$$

$$s(q_i, a, c) = (q_j, \epsilon)$$

Teacher's Signature

③ SKIP



$$\delta(q_i, a, z_0) = (q_j, z_0)$$

Eg. $L = \{ w \in \{a, b\}^+ \mid n_a(w) = n_b(w) \}$

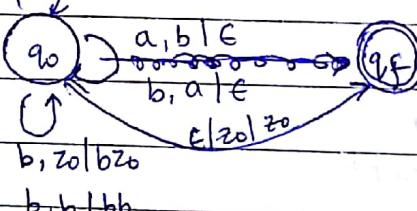
→ i/p 'a' : a will pop out b, else, push a

→ i/p 'b' : b will pop out a, else, push b.

$a, z_0/z_0$

aababb

a, a/a



(q_0, a, z_0) includes $(q_0, a z_0)$

(q_0, a, a) " (q_0, aa)

(q_0, b, z_0) " (q_0, bz_0)

(q_0, b, b) " (q_0, bb)

(q_0, a, b) " (q_0, ϵ)

(q_0, b, a) " (q_0, ϵ)

Eg. $\xrightarrow{\epsilon} (q_0, \epsilon, z_0) \xrightarrow{\epsilon} (q_0, \epsilon) \xrightarrow{\epsilon} (q_0, \epsilon, z_0) \xrightarrow{\epsilon} (q_f, z_0)$

Eg. : Paranthese :

$(, z_0/z_0$

$, (/)$

$\xrightarrow{\epsilon} (q_0, (/), z_0/z_0) \xrightarrow{\epsilon} (q_0, (/))$

$\xrightarrow{\epsilon} (q_0, (/))$

$(/) / \epsilon$

$(q_0, (, z_0)$ includes $(q_0, (z_0))$

$(q_0, .(, .))$ includes $(q_0, ((.)))$

$(q_0, (, .))$ " $(q_0, (.))$

$(q_0, (, \epsilon))$ " (q_f, z_0)

Teacher's Signature

in $\{wwr\}$, $\{amb^n : m \leq n \leq 2m\}$: non-deterministic PDA.

Can't be accepted by DPDA

Source
DATE: / /
PAGE NO.:

Deterministic PDA (DPDA)

$$PDA = (\mathbb{Q}, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

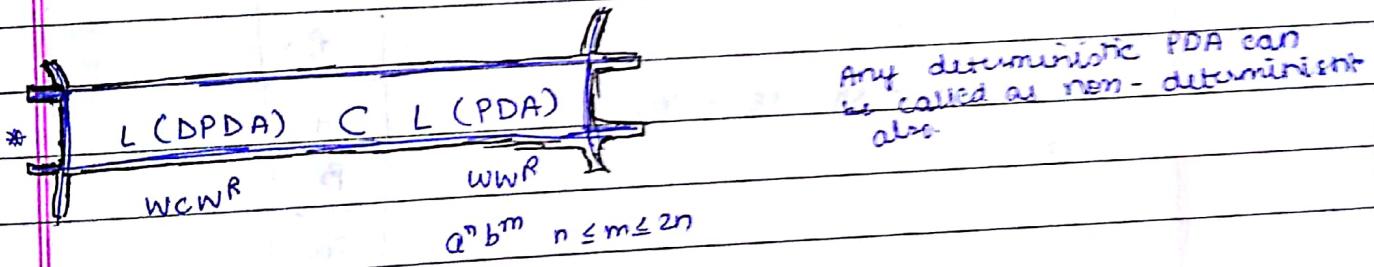
in which

- (1) $\forall q \in \mathbb{Q}, a \in \Sigma \cup \{\epsilon\} \times \Gamma$ ^{at most}
 $\delta(q, a, x)$ contains at unique pair (q_r, x)

- (2) If $\delta(q, \epsilon, x)$ is non-empty, then \Rightarrow can't read ϵ & a
 $\delta(q, a, x)$ should be empty & vice versa

$\rightarrow \{wcr\}$: whenever c is here, we know w is odd

\Downarrow accepted
it'll be DPDA



Equivalence of PDA and CFG

1) $CFG \Rightarrow PDA$

2) $PDA \Rightarrow CFG$

3) $CFG \Rightarrow PDA$

$$P: A \rightarrow \alpha, \alpha \in V^*$$
$$\alpha \in (V \cup T)^*$$

Given a CFG, $G = (V, T, P, S)$

Construct an eq. PDA A s.t. $L(A) = L(G)$

Teacher's Signature

each is in
sentential
form

$$S \xrightarrow{\text{derives}} \alpha A \beta, \quad \alpha, \beta \in (V \cup T)^*$$

$$\rightarrow \alpha \gamma \beta$$

$$\rightarrow \alpha B \gamma' \beta$$

$$\rightarrow \alpha \gamma'' \gamma' \beta$$

$$\rightarrow aabbaa--$$

$$A \rightarrow \gamma$$

→ producing sentential form

by applying rules in
grammar

$$A \ B$$

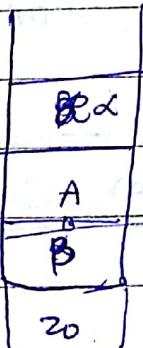
- we'll construct PDA which accepts by empty store
- * we use CFG with ^{left} most derivation.

$$S \rightarrow \alpha A \beta$$

↓

Pop NT & push its rule.

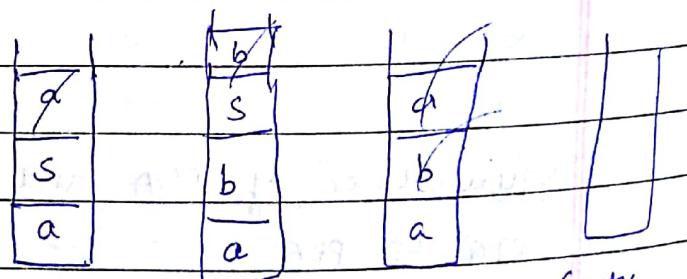
Pop T.



$$S \rightarrow a S a$$

$$S \rightarrow b S b$$

$$S \rightarrow a$$



[a | b | a | b | a]
↑ ↑ ↑ ↑ ↑ ↑

(make sure
i/p symbol & tos
are same)

→ It'll accept by PDA.

$$P = (S, Q, \Gamma, V, U, T, S_0, q_0, S, \phi)$$

f.s. not needed

S transition :-

- 1) \forall non-terminals $A \in V$
(variables)

$$s(q, \epsilon, A) = \{ (q, B) \mid A \rightarrow B \text{ is a prod' rule in } P \}$$

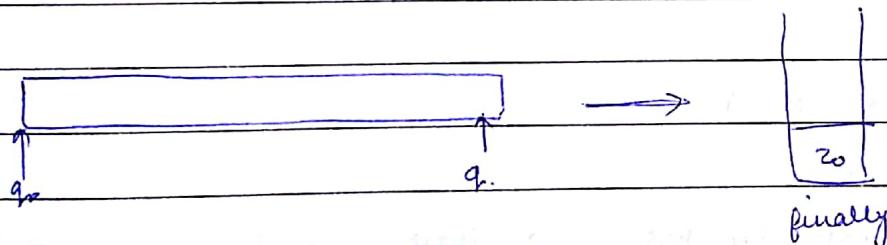
\downarrow
E move

- 2) \forall terminals $a \in T$

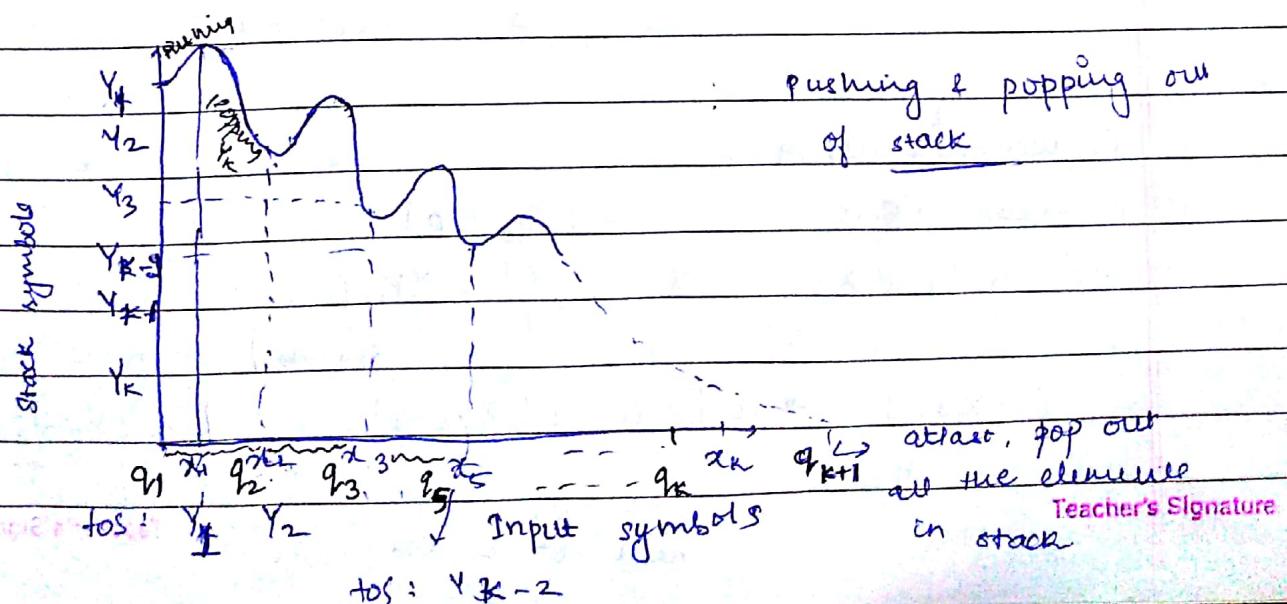
$$s(q, a, a) = \{ (q, \epsilon) \}$$

$$\text{No. of rules} \equiv V + T$$

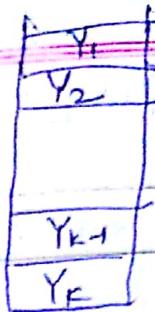
- 2) PDA \Rightarrow CFG (PDA : acceptance by empty store)
- PDA $P = (Q, \Sigma, \Gamma, S, q_0, z_0)$



Graph :



- b/w $q_1 - q_2$: read x_1
 $q_2 - q_3$: read x_2
 \vdots
 $q_k - q_{k+1}$: read x_k



→ if I want to pop $Y_k \Rightarrow$ should've popped out Y_1, Y_2, \dots, Y_{k-1} symbols already.

from q_1 to q_2 : we are popping Y_1

→ state : $[q_1, Y_1, q_2]$

similar from q_2 to q_3 : $[q_2, Y_2, q_3]$

→ popping each symbol while reading some input symbol

→ $[p X q]$: p is a state
 q is a state

$[p X q] \&$ is a NT to be popped out.

$[q_0 Z_0 q]$: $q_0 \rightarrow q$. In meanwhile, pop Z_0 : empty stack



$$G = (V, \Sigma, R, S)$$

V consists of

(1) Special symbol S → Start symbol

(2) All symbols of the form $[p X q]$

where $p, q \in Q$ and $X \in T$

Production rules of P

(1) \forall states $p \in Q$ $S \rightarrow [q_0 Z_0 p]$

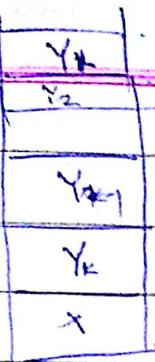
(2) let $S(q, q \notin X)$ contains $(\tau, Y_1 Y_2 Y_3 \dots Y_k)$

pushing k symbols

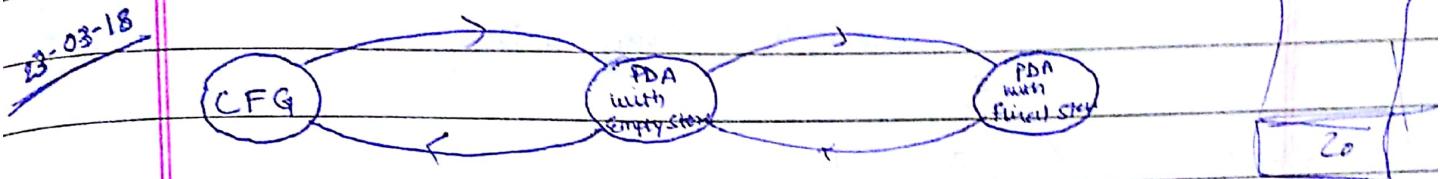
$[q, X \tau_k] \rightarrow a [\tau Y_1 \tau_1] [\tau Y_2 \tau_2] \dots [\tau Y_{k-1} \tau_{k-1}] [\tau Y_k \tau_k]$

need not be the same

Teacher's Signature



\Rightarrow Pop out all $Y_1, Y_2, Y_3, \dots, Y_k$



e.g. PDA : $\underline{i^n e^{n+1}}$ \rightarrow This is what we need

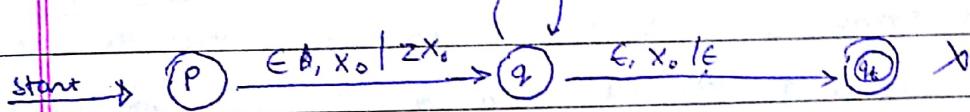
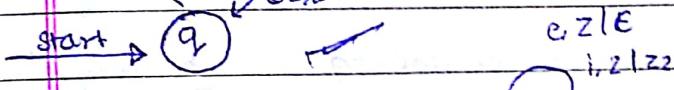
e.g. ~~If better words~~ $i \rightarrow i$ pushing Z when i comes

$e, Z | \epsilon$

$e \rightarrow \epsilon ee$

$i, Z | ZZ$

~~exists~~ : By empty store



Z	\xrightarrow{f}
Z	\xleftarrow{f}
Z	\xleftarrow{f}
X_0	

\Rightarrow No. of e = No. of i + 1 \Rightarrow violates rule

PDA \Rightarrow CFG \therefore 1

S for PDA :

(1) $S \rightarrow [q, Z_q]$

$S(q, e, z) \Rightarrow$ has (q, ϵ)

(2) $[q, Z_q] \xrightarrow{A} [q, Z_q]$

$S(q, i, z) \Rightarrow$ has (q, ZZ)

(2) $[q, Z_q] \rightarrow i [q, Z_q] [q, Z_q]$ { have 2 more Z to be popped but S

\leftarrow 2nd Prod for this

(3) $[q, Z_q] \rightarrow e$ already popping it out
, don't have to capture anything else

$S \rightarrow A$
 $A \rightarrow i A A$
 $A \rightarrow e$

$[q, Z_q] \rightarrow i [q, Z_q] [q, Z_q]$

\downarrow may vary from 1st to n
 $\Rightarrow n^2$ rules

Teacher's Signature

Chromatography: $A \rightarrow BC$: ↑ NT by only 1
 $A \rightarrow a$ + " "

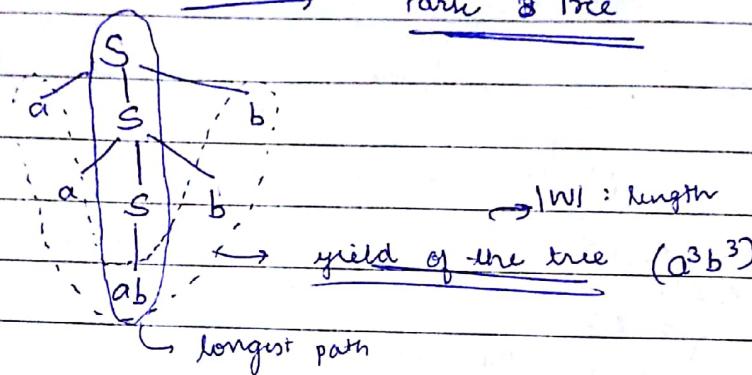
DATE : / /
PAGE NO. :

CFG

$$S \rightarrow aSb \rightarrow a^n b^m$$

$$\underline{S \rightarrow ab}$$

Park & Tree



Theorem: Suppose we have a parse tree according to Chomsky.

Normal Form. $G = (V; T, P, S)$ and suppose that the yield of the tree is a terminal string w . If the length of the longest path is n , then

$$|w| \leq 2^{n-1}$$

28-03-18

Proof: $n = 1$

$$A \rightarrow a \quad |w| = 1$$

$$1 \in \mathbb{Z}^{1-1} \quad \checkmark$$

\Rightarrow True for the basis $n = 1$

Hypothesis : result is true for $n-1$

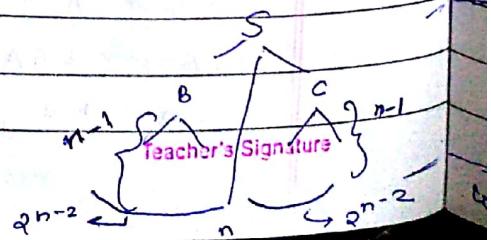
$$\Rightarrow |w| \leq g^{n-2}$$

Show for n : $(n > 1)$ \Rightarrow need to start in form $A \rightarrow a (n=1)$ (can't have $g \rightarrow BC$ (Chomsky))

length of longest path = n

$$|w| \leq 2^{n-2} + 2^{n-2}$$

$$|\omega| \leq 2^{n-1}$$



Pumping Lemma for CFL

let L be a CFL. Then there exists a constant n s.t. if $\exists z \in L$ s.t. $|z| \geq n$, then we can write

$z = uvwxy$, subject to following conditions :

(1) $|vwx| \leq n$

(2) $\#vx \neq 0$

(3) $\forall i \geq 0 \quad uv^iwx^i y \in L \quad (v \& x \text{ can be pumped any no. of times})$

* grammar should be in Chomsky Normal Form

(→ fed by 1 terminal only at 1 step)

(If language is empty / $\epsilon \Rightarrow$ CNF can't exist for L)

Lemma always satisfied if this case is considered.

Suppose G is a CFG in CNF. Let G has m variables.

Let $n = 2^m$

Let $|z| \geq n = 2^m$

Using previous theorem,

longest path should have length $> (m+1)$
in parse tree
of z

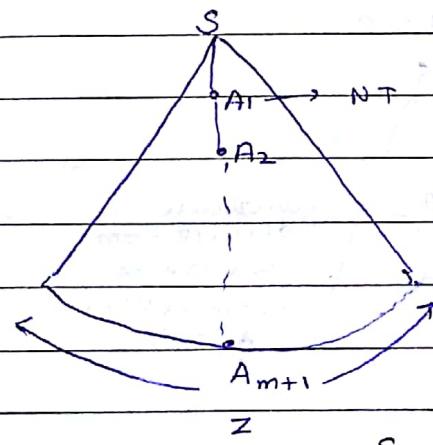
m variables, hence

at least $m+1$ NT



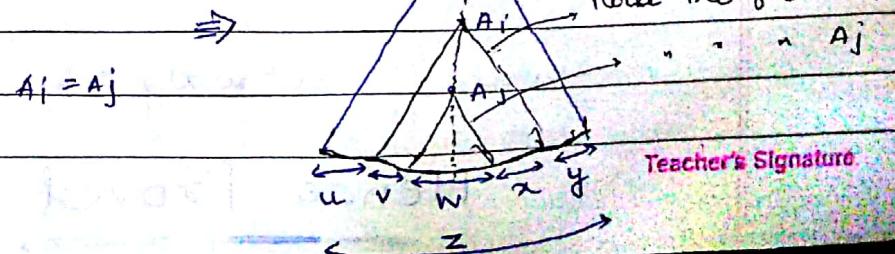
Using Pigeonhole.

at least 2^m NT are
same



yield of S
 \geq yield
of $B + C$

can't
have
longest path $> n-1$



1.) Max. length b/w A_i & $A_j = m$
 \Rightarrow yield : $2^{m-1} \leq n^{\rightarrow q^m}$

$$\Rightarrow |vwxi| \leq n$$

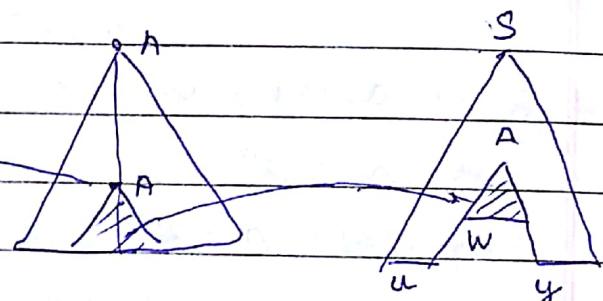
2) either v or x should not be \in .

if $vn = e \Rightarrow$ both yields of A_i & A_j are same which is
 (both are e) not possible because we have atleast 1 derivative
 in b/w. (atleast of length 1)

3.) A_i & A_j are one & the same

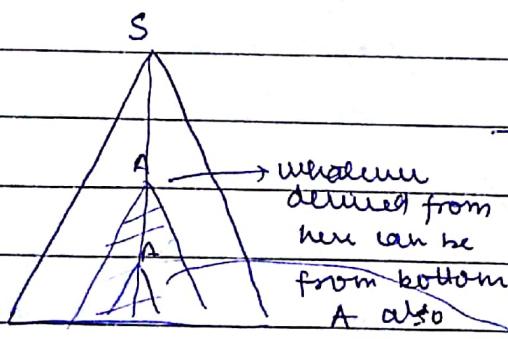
$$\text{let } A_i = A_j = A$$

whatever is derived
 from here can be
 derived from above A
 also



$$\Rightarrow \underline{uvwxy \text{ EL}}$$

→ Vice - versa



$$\Rightarrow \underline{uv^2wx^2y \text{ EL}}$$

it loses hold on
 $\#x$ & $\#v$

$$\text{likewise, } \underline{uv^ix^iw^jy \text{ EL}} \quad i \geq 0$$

Hence Proved

Teacher's Signature

~~to prove~~

$$(1) L = \{a^n b^n c^n \mid n \geq 1\} \rightarrow \text{not CFL}$$

let us assume L is context free language.

by pumping lemma, $\exists m \text{ s.t.}$

$$z \in L \text{ st } |z| \geq m \text{ then } z = uvwxy \text{ &}$$

all 3 conditions are satisfied.

~~but not taken~~

Let us take the string $a^m b^m c^m$

$$a^m b^m c^m = uvwxy$$

$$\text{s.t. } |vwx| \leq m$$

$\Rightarrow x$ can consist of only a 's & b 's (not c 's)

$x =$

comb' of

b 's & c 's.

$\Rightarrow v$ can consist of only a 's / only b 's / only c 's / a 's & b 's / or
 b 's & c 's
 (can't have all 3 symbols since $|v| \leq m$)

$\Rightarrow x$ can also consist of same values.

$$\Rightarrow v = a^i \quad x = a^j$$

$$(i=0) \rightarrow uvwxy : \Rightarrow uw = a^{m-i-j} b^m c^m \notin L \quad X^{\text{NOT true}}$$

violate cond' of Pumping Lemma

$$(i \neq j \neq 0 : \text{cond' } vx \neq \epsilon)$$

\Rightarrow Not a CFL.

$\hookrightarrow a$'s & b 's : $(a^i b^i)^i$: can't be accepted in given
 language (a can't come after b)

\rightarrow have to tell for all comb's

show for 1 & show it is similar for others.

Teacher's Signature:

Using pumping lemma, p.t. following 1 are not CFL.

(2) $L = \{a^i b^j c^i d^j\}$

(3) $L = \{ww \mid w \in \{a, b\}^*\}$

w $a^i b^j c^i d^j = \cancel{a^i} uvwxy$

why maths not included in noble prize

→ every problem can be related to decision problem

DATE:	/ /
PAGE NO.:	

TURING MACHINES

→ infinite tape :

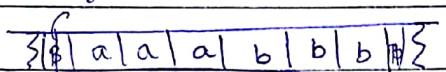


↑ can move to the right as well as to the left → role of flexibility

Δ replacement of symbol is also allowed

e.g. $L = a^n b^n$

blank



head
a, replace
with x

escape all
a, find 1's b & replace with Y & so on

3 | x | x | x | Y | Y | Y | S : find no a & b : reach to final state

→ here, counting is done using matching.

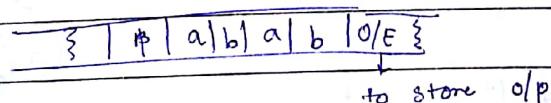
4/4/18

Turing machine can be treated as : Automata

(1) Accepting Device (like PA, PDA) → Decision Problem (Y/N)

(2) I/O Device

count b odd/even



to store o/p

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

Q = set of finite no. of states

Σ : input alphabet

Γ : tape alphabet (includes Σ also) $\Sigma \subset \Gamma$

q_0 : start state

δ : blank symbol

F : set of final states → makes sense only in case of accepting

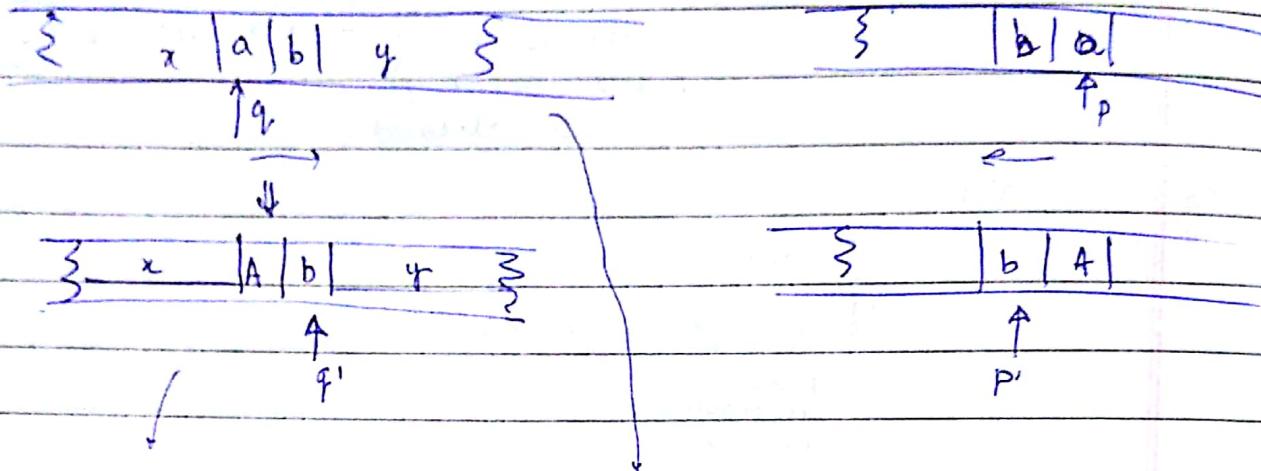
Non-deterministic

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

Teacher's Signature

moving left / Right

$s(q, a)$ includes (q', A, R) , $s(p, a)$ includes (p', A, L)



Instantaneous
Description : $x A q' b y$

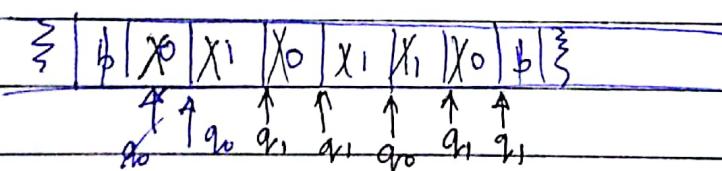
ID: $xqaby$

$x A q' b y \rightarrow xqaby$ after applying above rule.

Turing Machine as I/O device

1) Parity checker : check whether even no. of 0's or
odd no. of 1's

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



x : already read that
symbol

read 1 \rightarrow move next state

$(q_1, B) \rightarrow \text{odd}$

$(q_0, B) \rightarrow \text{even}$

$S(q_0, 0)$ includes $\$ (q_0, x, R)$

$S(q_0, 1)$ includes $\$ (q_1, x, R)$

$S(q_1, 0)$ includes $\$ (q_1, x, R)$

$\$ (q_1, 1)$ includes $\$ (q_0, x, R)$

$S(q_0, \#)$ " $(\text{Halt}, \epsilon, -)$
↓
Halt

$S(q_1, \#)$ " $(\text{Halt}, 0, -)$

→ Halt always

in this case.

But in some cases,

it may not halt.
if not accepted.

Turing machine

→ procedure need not halt in case of all i/p (O/P Yes: halt
No: may not halt)
Algo needs to " " " "

Perfect no. : n is perfect if sum of all divisors except n = n

$$\text{Eg. } 6 = 3 + 2 + 1$$

$$28 = 1 + 2 + 4 + 7 + 14$$

Q: Is there a perfect no. > n ? \Rightarrow Check $n+1, \xrightarrow{\text{NO}} n+2 \xrightarrow{\text{No}} n+3$.

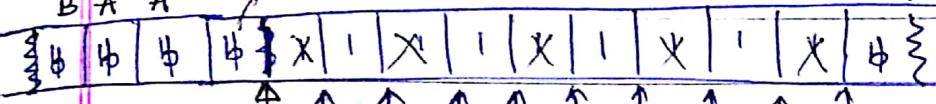
↓
may give procedure but not algo.

may not end also

118

example 3 Unary to Binary Conversion.

$$\begin{array}{r} \text{B A A} \\ \text{B A A} \\ \hline \text{B A A} \end{array} \quad \begin{array}{l} \text{9 1's} \\ \Rightarrow \end{array} \begin{array}{l} \text{Binary} \\ \text{1001} \\ \text{BAAB} \end{array}$$



$q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9 q_{10} q_{11}$: remainders

to read all symbols : if odd no. of $1^s \Rightarrow$ 1 null com.
at last
Position

$$\begin{array}{r}
 2 | 9 \\
 2 | 4 \\
 \hline
 2 | 2 \\
 2 | 1 \\
 \hline
 0 | 1
 \end{array}$$

for quotient: alternate x : count 1's

Transitions :

$$s(q_0, 1) \rightarrow_{\text{def}} = (q_1, x, R)$$

$$S(q_1, 1) = (q_0, 1, R)$$

$$s(a_1, \Phi) = (a_2, \phi, l) \xrightarrow{\text{to add remainders}} \begin{matrix} a_1 + b \\ \oplus \\ a_2 \end{matrix}$$

$s(q_2, 1/x) = (q_2, 1/x, \omega) \rightarrow$ don't do anything, just

$$S(q_2, \mathbb{H}) = (q_2, B, R) \rightarrow \text{put } B$$

$$S(q_4, z_1) = (q_4, z_1, R), \quad z_1 \in \{A, B, Z\}$$

$$S(q_1, x) = (q_1, x, R)$$

$$s(q_0, \Phi) = (q_2, \Phi, L) \Rightarrow \text{even no. of 1's} \Rightarrow A$$

$$g \circ (q_3, 1/x) = (q_3, 1/x, L) \rightarrow \text{just move to left}$$

$$S(q_3, A/B) = (q_3, A/B, L)$$

$$S(q_3, z) = (q_3, z, L) ; \quad z \in \{1, x, A, B\}$$

$$S(q_3, \psi) = (q_4, A, R) \Rightarrow \text{put } A$$

$$s(q_0, x) = (q_0, x, R)$$

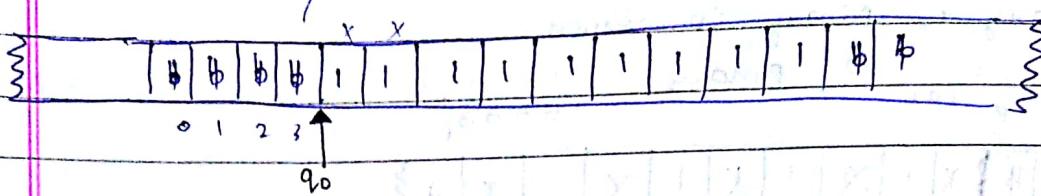
$$S(a_1, x) \models (a_1, x, R)$$

$$s(q_+, \mathbb{H}) = (\text{Hart}, -, -)$$

$$\Omega = \{q_0, q_1, q_2, q_3, q_4\}$$

DATE: / /
PAGE NO.:

Other way: treat this as a binary counter : incrementing binary no by 1



Encounter 1 \rightarrow q_1, B, R at B (3) (1)

2nd 1 \rightarrow move L: see B $\Rightarrow B \rightarrow A$ (10)
(2) $B \rightarrow B$

3rd 1 \rightarrow move L: see A $\Rightarrow A \rightarrow B$ (11)

$q_0 \rightarrow$ search 1's (A/B)

$q_1 \rightarrow$ move to left & write A/B

if B $\rightarrow B \rightarrow A$

go left $\rightarrow B \rightarrow A$

go left $\rightarrow B \rightarrow B$ (100)

$s(q_0, 1) = (q_1, x, L) \Rightarrow$ you want to count no. of 1's

$s(q_1, B) = (q_0, B, R) \Rightarrow$ (skip)

$s(q_0, x, R) = (q_0, x, R) \Rightarrow$ skip

$s(q_1, x) = (q_1, x, L)$

$s(q_1, A) = (q_1, A, L) \Rightarrow$ (skip)

~~$s(q_1, B) = (q_1, B, R)$~~ $s(q_0, B) = (q_0, -, -)$

$s(q_1, A) = (q_0, B, R) \Rightarrow$ (skip)

\rightarrow Only 2 states are used here.

(1) \rightarrow vs (2). $\left\{ \begin{array}{l} \text{like} \\ \text{running time} \end{array} \right\}$

No. of cells \sim

No. of cells you have to traverse

\sim will be less

is more (skip ones) (hub ones)

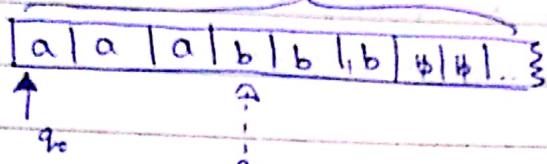
①: \uparrow \circ .

②: $\downarrow \rightarrow$ {like memory}

Teacher's Signature

Turing Machine as an Accepting Device

- we assume tape is finite on left side, infinite on right side
- output is from left, state side
- head is at left side in the starting.



$$= (\emptyset, \Sigma, \Gamma, S, q_0, \#, F)$$

Instantaneous Description = $a a a q_0 b b b$ $\xrightarrow{S \text{ in state } q_0 \text{ & going to read } b}$

Starting " " : $q_0 w$

$$q_0 w t^* \alpha_1 q_f \alpha_2, \quad \alpha_1, \alpha_2 \in \Gamma^*, \quad w \in \Sigma^*, \quad q_f \in F$$

→ tape alphabets \neq input alphabets
 If this is possible
 w is accepted by TM

*. The head may not even need all the string. ($\alpha_1 q_f \alpha_2$)

↳ if you want to read all i/p symbols : you've to take care

$w \in \Sigma^*$

$$L(M) = \{ w \mid q_0 w t^* \alpha_1 q_f \alpha_2 \text{ s.t. } \alpha_1, \alpha_2 \in \Gamma^*, q_f \in F \}$$

↳ Type 0 language /

Recursively Enumerable

language (RE) ↳ can be counted

↳ using same set of transition table again & again

Teacher's Signature

11/4/18

PAGE NO.:

Turing Machine :- accesses i/p (ω) & gives o/p ($f(x)$)
↓
defines a func'

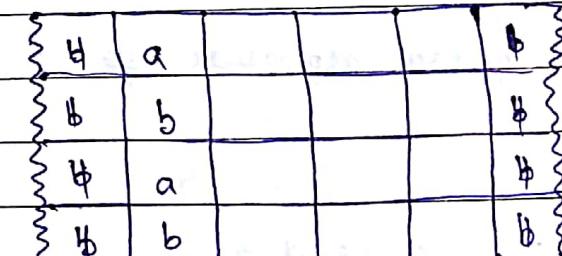
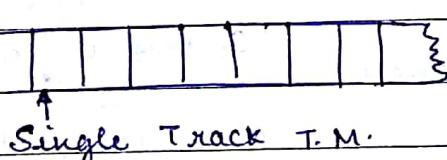
T.M. not only defines a func', but also tells how that func'
is derived \Rightarrow T.M. is beyond a func' \Rightarrow gives algo also,

↳ set of all T.M. which defines func's \rightarrow Turing Computable Func's

features of T.M.

- (1) Finite Control (can store some info.)
set of states (e.g. even / odd no. : 2 states q_0 and q_1)
- (2) Marking some symbols on tape is like storing some info.
(e.g. : Quotient : 'x' se 1 ko alternatively replace kr rhe the)

(3)



similar to

Multi-track T.M.

(different from multtape)

$\{ b \mid abab \mid abaa \}$

(1 Head)

can read all symbols in
that col.

↳ multitrack can be converted into single track

↳ single track is always multitrack

↳ Both are equivalent

$$MT \cap TM \equiv ST \cap TM$$

good when don't
want to have clumsiness

Teacher's Signature

Accepting : semi infinite tape

I/O : infinite tape

PAGE NO.:

DATE: 11

Eg: → In many eg. :



(3) : Single track TM \Leftrightarrow multitrack TM

(4) : Checking of symbols :

Eg: → WCW as an acceptor device $w \in S$

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

↑ ↑
q_a after c,
(have read
a : go
to q_a)
is q_a : go
to q'_a
then start
moving left
until x

Halt : when only c & all
other x.

go to left till get x : move R
if get c : also check if all are
x in Right. → Accept. (Halt)

(5)

Generalised TM

$\Sigma A \Gamma$ → I/O device ①

$\boxed{\quad \quad \quad \quad \quad} \Sigma$ → accepting device ②

If you move
left : TM will abruptly stop : won't accept string

whatever you can do in ② \Rightarrow you can do in ①

in ① \rightarrow Put a marker before that so that TM won't go beyond
that ~~cell~~ cell. By that way : it becomes one side finite

but this was ~~one side~~ true. (O.K.)

For nice view

② → Take 2 track

take i/p
in 1st track

A ₁	A ₂	A ₃	A ₄	A ₅	A ₆
B	C				

whatever will be in left to head : bottom track
right : top track

but in infinite tape = $(R \rightarrow R) \{B, C\}$
but here = $(L \rightarrow R) \{B, C\}$

III

A ₁ B	A ₂ C	..			
------------------	------------------	----	--	--	--

1.) One-sided Infinite tape = Both sided Infinite tape

2) Multi tape TM \Rightarrow Single Tape Turing Machine

↓

A									
---	--	--	--	--	--	--	--	--	--

: each tape has its own two heads

B									
---	--	--	--	--	--	--	--	--	--

--	--	--	--	--	--

A									
---	--	--	--	--	--	--	--	--	--

multi-track

All 3 made are independent

∴ state will be same

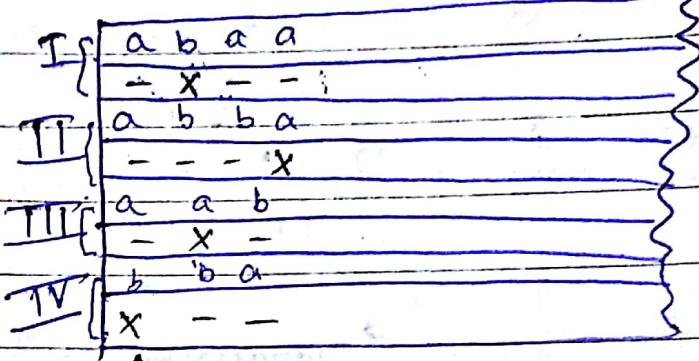
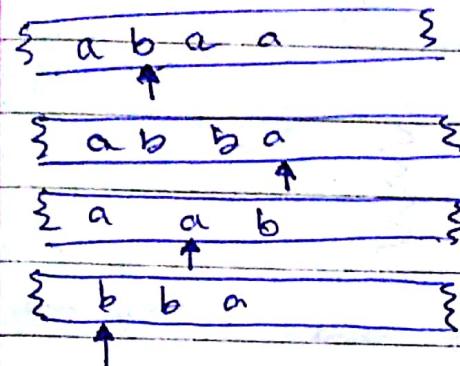
$$S(q_0, A, B, A) \rightarrow (q', A', B', A'', \underbrace{R, L, R}_{\text{define movement}})$$

↑
state
is same

at each state
for each tape

Multitape

for every tape, I'll have
a tracks



top track : i/p symbol

if head
is here, it has to see where is X.

bottom track : mark X where header is

$(q_r, \underline{b} \underline{a} \underline{a} \underline{b}, q_f, f)$

where X is counter, how many are filled

Now, it has to simulate after reading b, what 1st tape will read. + so on for each tape

suppose 2nd tape : $a \rightarrow A$ & moves to left

say q_r ,

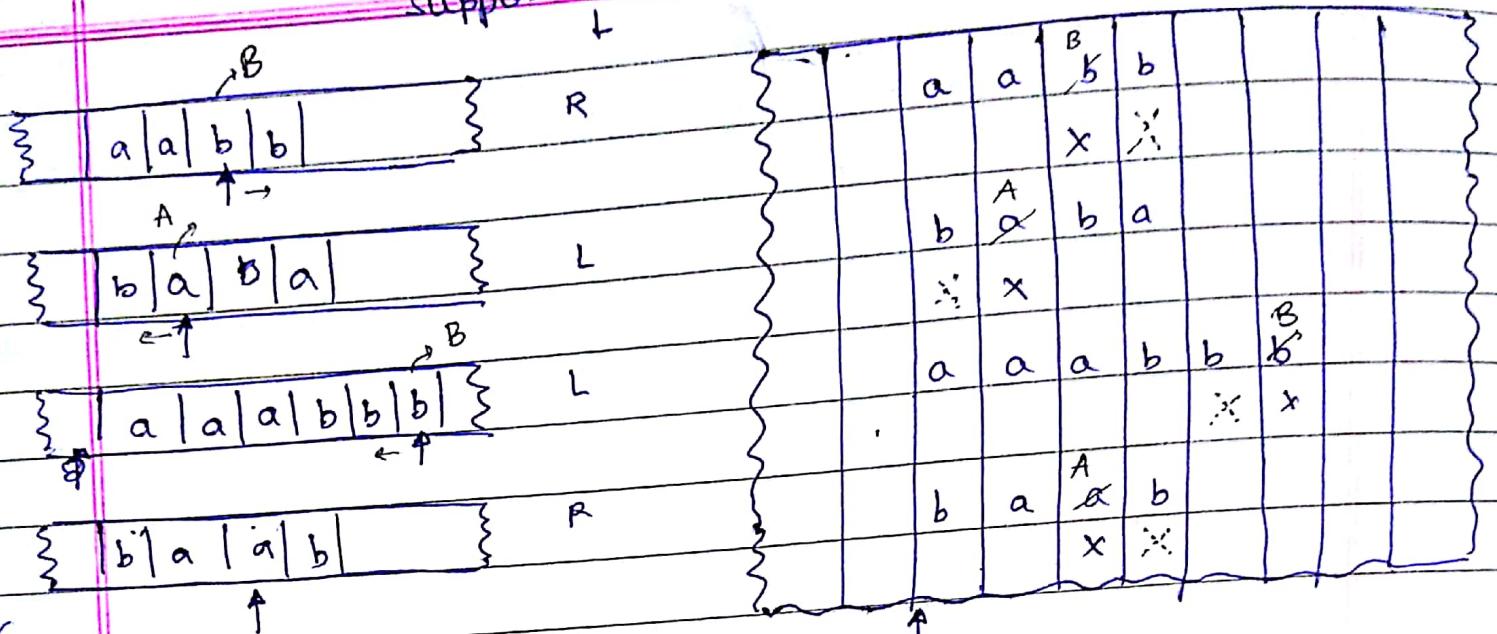
For 1 transition,

$\xrightarrow{(q_r, \underline{\quad \quad \quad}), \quad \quad \quad}$

for this, you may've to move full length of 1st tape

may have to move back in left full length

4/18
Suppose movement:



$$S(q_r, baba) = q^r (BABA, RLRL)$$

First, it need to move through all strings to get x .

$$S(q_r, \underline{b} \underline{a} \underline{b} \underline{a}, \text{ } \text{ })$$

counter (increment as you get x)

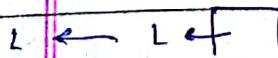
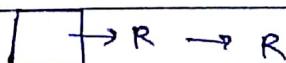
1st pass: $L \rightarrow R$ (to get x)

2nd pass: $R \rightarrow L$ (replace $b \rightarrow B$ & move to right)

1st x will come for 3rd tape: have to store which one is finished also
replace by B & move to L

\Rightarrow If 2 passes.

\Rightarrow if there are n movements in multitape, no. of movements in multitrack \Rightarrow "Polynomial" (not exponential)



Distance = 2 4

Deterministic T.M. and Non-Deterministic T.M.

→ NTM: $s(q, a) = \{ (q', A, R), (q'', B, L), (r, C, R) \}$

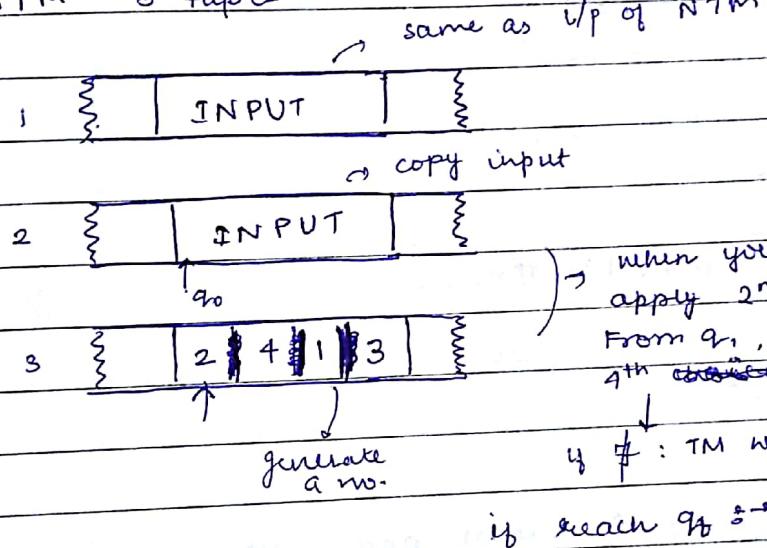
$$s(p, b) = \{ (p', B, R), (r, B, L) \}$$

1 2 3
1 2 3 } INPUT }

Have to simulate this NTM using DTM.

↳ find transition which has max^m tuples. → label them

↳ DTM: 3 tapes



) when you're in q_0 , you've to apply 2nd tuple for that → q_1 . From q_1 , you've to ~~get~~ apply 4th choice

if \neq : TM will stop.

if reach $q_f \Rightarrow$ accepted.

If you don't accept: generate another no. & check if it accept or not

exam 2nd & 3rd & generate another no. → have to repeat many times for same input

↳ No. of transitions will be: "Exponential"

NTM \Leftrightarrow DTM

(Power is same)
(^{no. of options} ~~no. of transitions~~ will be exponential)

→ $s(q_0, \#) = \{ (q_0, 1, R), (q_1, 1, R) \} \quad q_1 \in F$

$q_0 \xrightarrow{1} q_1$: can generate any no. of 1's (unary no.)

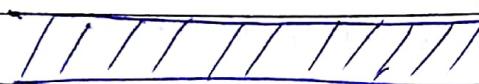
Teacher's Signature

→ if NTM ends : DTM also ends

if NTM goes to ∞ loop , DTM also goes to ∞ loop

3) Turing Machine as a Generating Device : similar to i/o device somewhat

M



$w_1 \# w_2 \# w_3 \# w_4 \# w_5 \# w_6 \dots$

marker

* M generates
 w_2

if $\exists i$ s.t. $w_i \in \text{Output tape}$, we say that 'TM M generates w_i ' ($w_i \in G(M)$)

Thm.

$L = T(M_1)$: (language accepted by T.M. M_1) for some

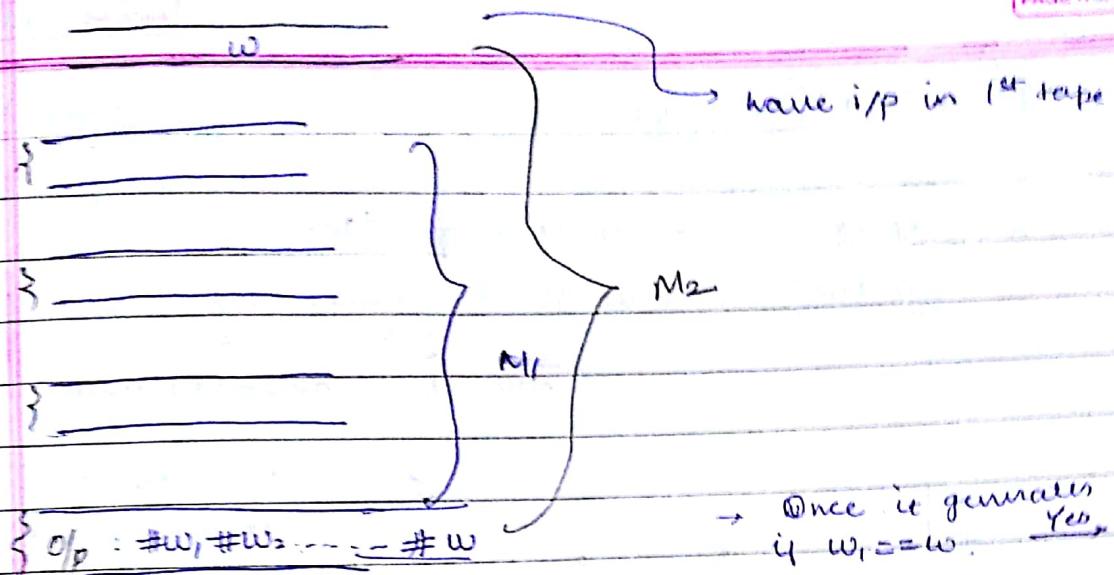
T.M. M_1 . iff $L = G(M_2)$ for some TM M_2 .

(language accepted by TM \equiv language generated by TM)

$$L = G(M_1) \quad \text{--- (1)}$$

$$L = T(M_2) \quad \text{--- (2)}$$

1) (1) \Rightarrow (2) ; using generating TM.



→ Once it generates & checks
if w₁ = w. Yes, Halt

↓ No

Move to w₂

- when generate in M₁ → can get accepted in M₂

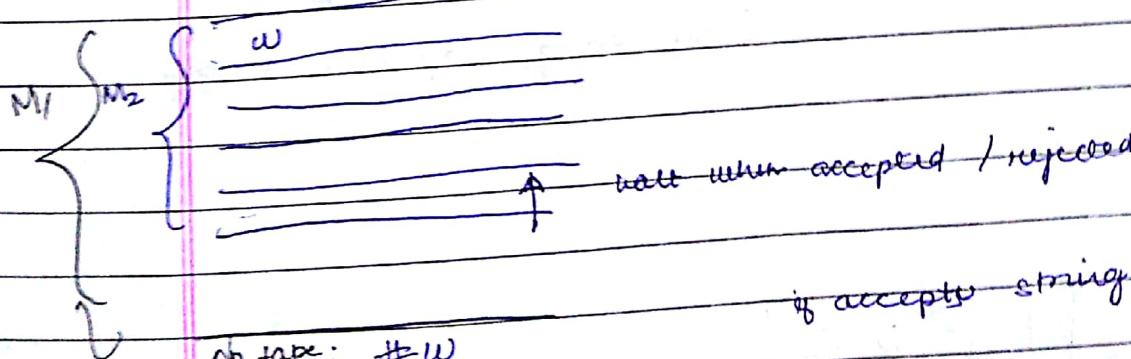
2) ② → ①

~~non recursive~~ If accepted : Halt

If not accepted : Infinite loop may go on

if we assume it is recursive language L. (even if not accepted)
L is a recursive language if & a TM, M s.t. L(M) = L
and L halts on all inputs

Suppose : M₂ is recursive



if accept string w: write in
output tape

↓

include this also
in M₁

Teacher's Signature

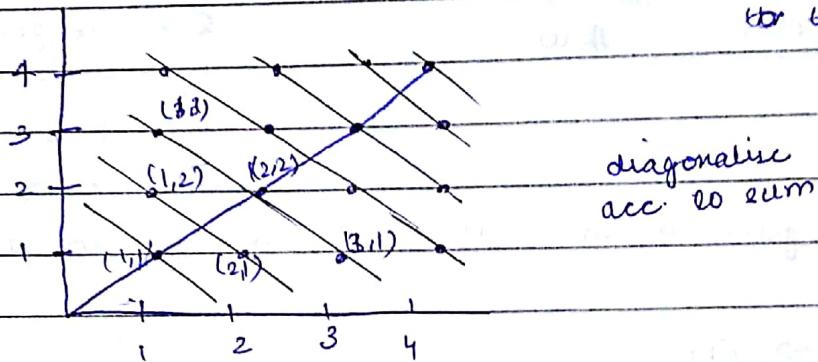
b) M_2 is non-recursive

if accepted : can write in o/p tape

rejected : will go in infinite loop
↓ now to check

. will use : "diagonalization"

like proving \mathbb{Q} is
not uncountably infinite



here,

→ include

can use many no. here

$\langle i, j \rangle$

: generate pair (i, j) (randomly)

will take w_i (i^{th} string accepted by language)

will generate it for j steps

(generate w_i nothing
j steps)

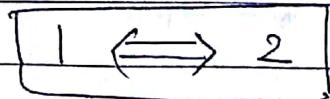
| if not generated in
j steps

↓↓↓

(Can change i & j accordingly)

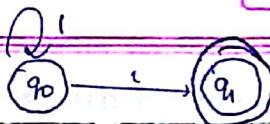
O/p tape

includes



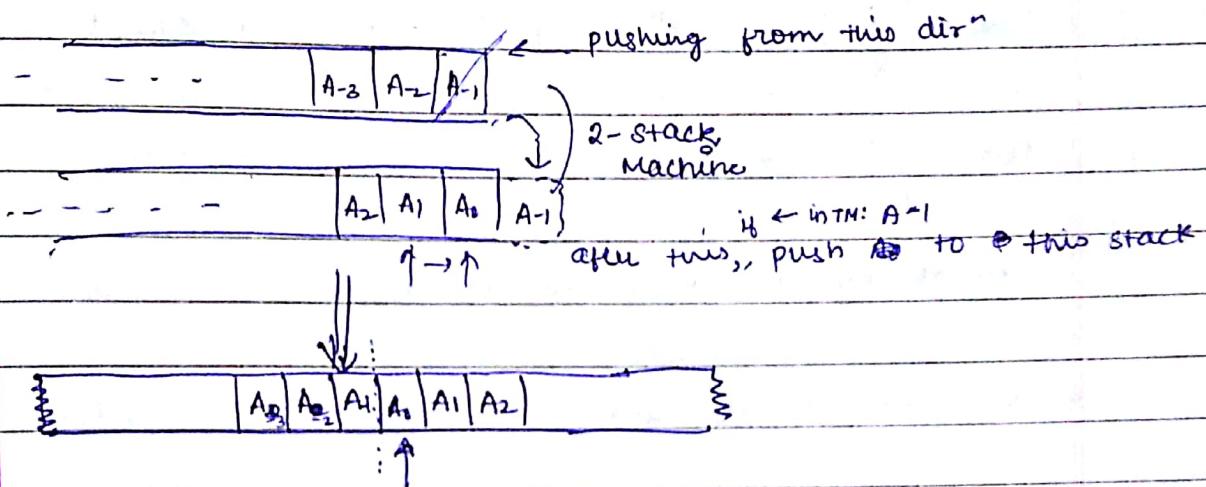
Teacher's Signature

→ can generate i & j easily using :



Restrictive Models of TM

1) 2-stack Machine = TM



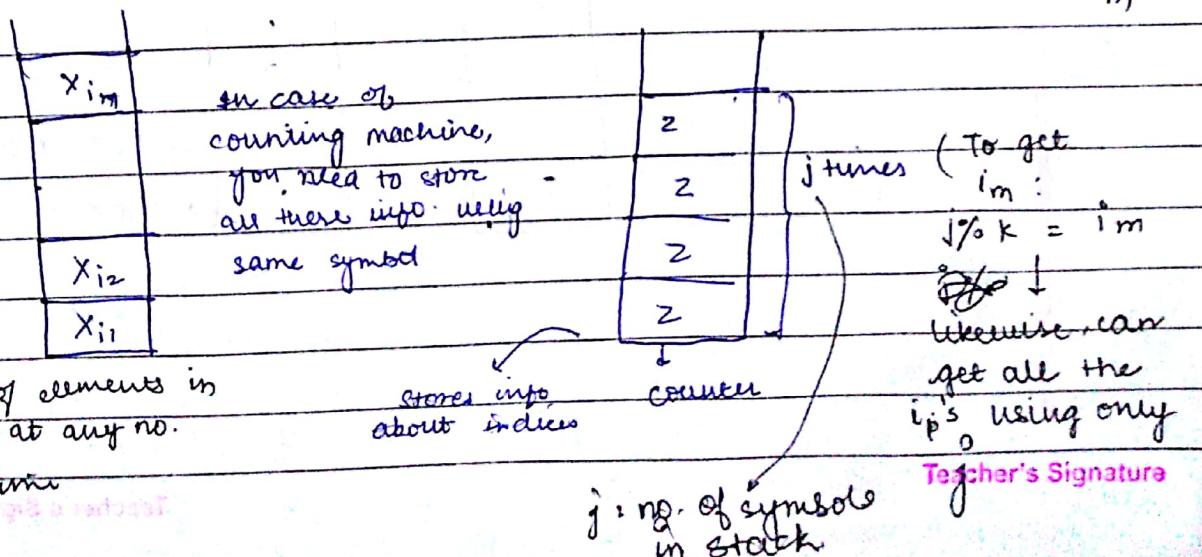
2) can have 1 symbol in stack and still get the T.M. using 4 counters. : Counting Machine (Counting no. of z's)

$[z|z|z|z] \dots$

1-stack by 2-counters

(2-stack : 4 counters \rightarrow TM)

$$\Gamma = \{x_1, x_2, \dots, x_{k-1}\}, j \Leftrightarrow k^{m-1}i_1 + \dots + k^2i_{m-2} + k^1i_{m-1} + i_m$$



→ we will require another counter in case of push/pop opn
↓
Stack with only
↓ symbol

18/4/18

$j \rightarrow$ unique representation

$$j \% K = i_m$$

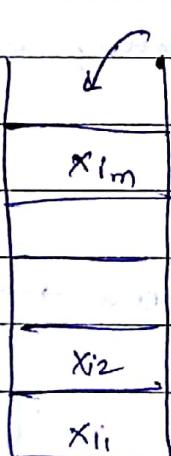
Quotient $\left(\frac{j}{k}\right)_{max} = k^{m-2}i_1 + k^{m-3}i_2 + \dots + k^{i_{m-2}} + i_{m-1}$

Rem $\left(\frac{j}{k}\right) = i_m$

divide by K , Rem will be i_{m-1}

→ You can get all indices $i_m, i_{m-1}, i_{m-2}, \dots, i_1$

↳ x_r needs to be pushed

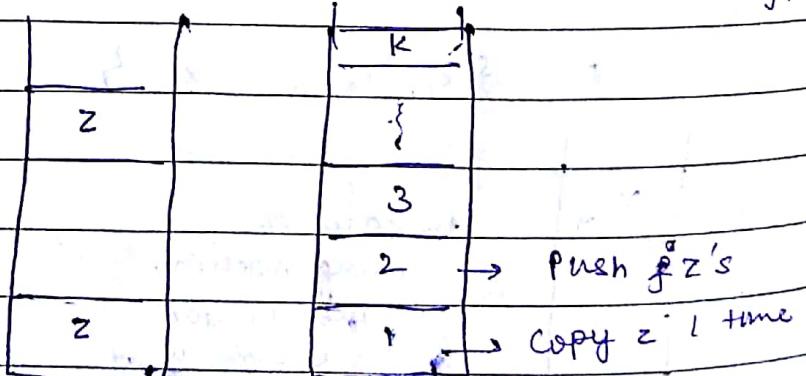


Represent for this

$$j \rightarrow jK + r$$

you need to know how many z you need to put

for $1 z \rightarrow$ you need to put $k z$ $\underbrace{z}_{j \text{ times}}$



→ Push z 's

→ Copy z 1 time

Teacher's Signature

↳ X_m needs to be popped

↳ divide by k :

1 stack \Leftrightarrow 2 cttrs

T.M. \leftarrow 2 stack \Leftrightarrow 4 cttrs

\Rightarrow 4.cttrs \rightarrow T.M.

Hence Proved

} Even 2 counters can form a T.M.

** For every T.M., \exists an equivalent T.M. that uses only 2 alphabets (symbols)

↓

Base of binary (Anything can be written as a comb' of 0's & 1's)
 becomes a language over 0 & 1.

Restrictions

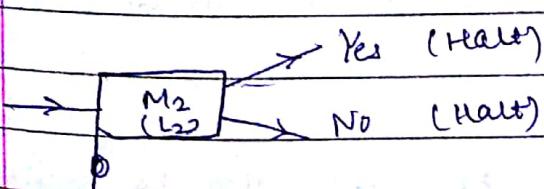
- 1) 2 - stack Machine = T.M.
- 2) 4 - counter = T.M.
- 3) T.M. can have 2 alphabets symbols
- 4) T.M. without rewriting

Recursive Sets

L_1, L_2 : Recursive languages

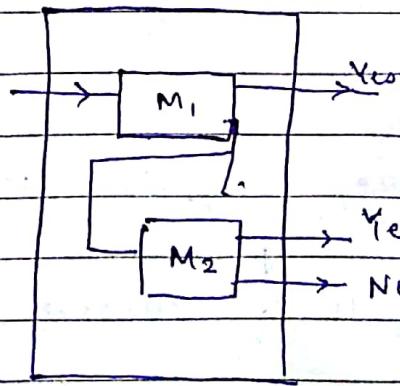
b) Is $L_1 \cup L_2$ recursive language?

Yes (Halt)



Halt in all cases

$M_1 \cup M_2$

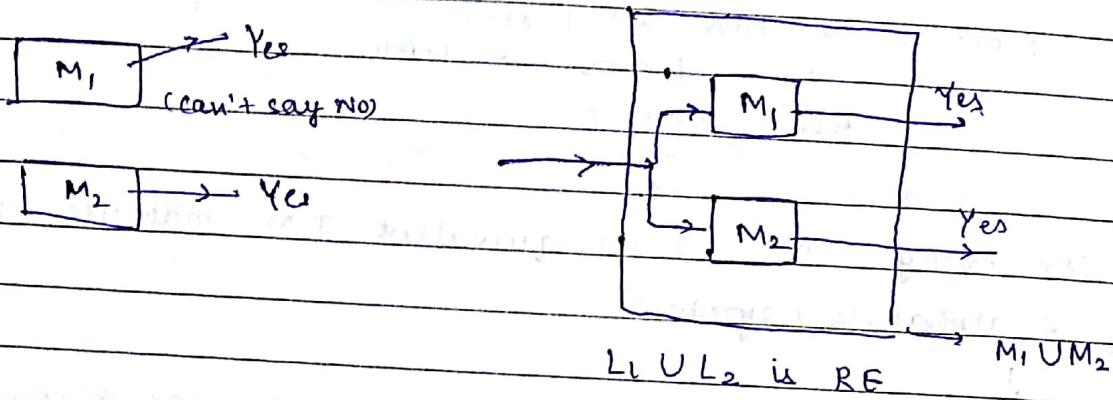


Halt in all cases
 $L_1 \cup L_2 \Leftarrow$ (Recursive) Teacher's Signature

if accepted : Halt

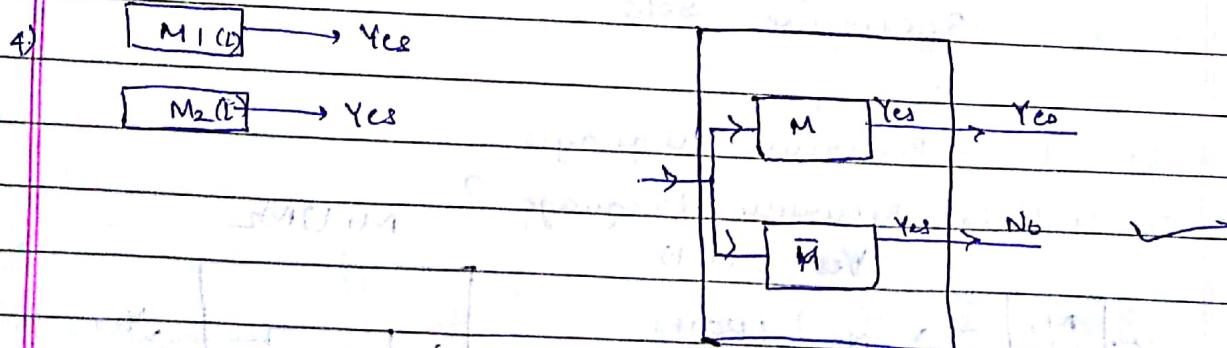
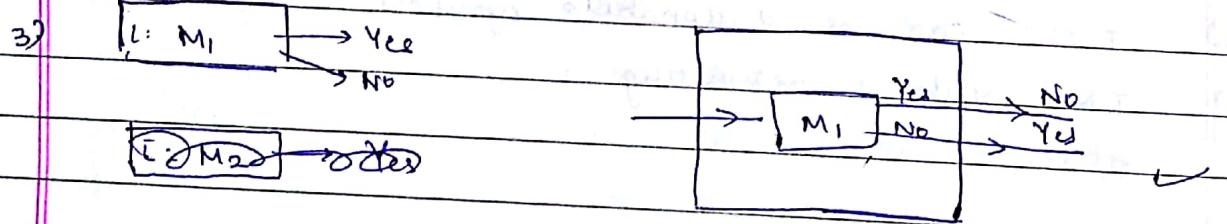
if not " : may not Halt

- 2) Is $L_1 \cup L_2$ Recursively Enumerable (RE), $L_1 \cap L_2 = ??$
 (Every recursive is recursively enumerable
 to do but not vice versa)



3) If L : recursive, is \bar{L} also recursive?

4) If L & \bar{L} are RE, then L is recursive?



$\Rightarrow L$ is recursive

Check for: $L_1 \cap L_2$, $L_1 - L_2$
 L_1, L_2 : Recursive

* If L_1 is recursive, its subset will also be recursive.

Teacher's Signature

Some definitions

1. Problem :

Given a graph, does it have Hamiltonian cycle?

I/P : G is graph

v_1, v_2, \dots, v_n are vertices vertex of graph.

Problem 1 : Does \nexists a hamiltonian circuit starting &

Decision problem \nexists ending with v_i .

(Yes / No)

If graph has Hamiltonian cycle \Rightarrow
it's \exists in graph.

(Yes / No)

Problem 2 : Find the Hamiltonian circuit in G starting

General problem \nexists with v_i

(Exact path)

(Decision + some other problems)

2. Instance (of a problem) : taking specific eg. (of graph)



: Instance of graph

G

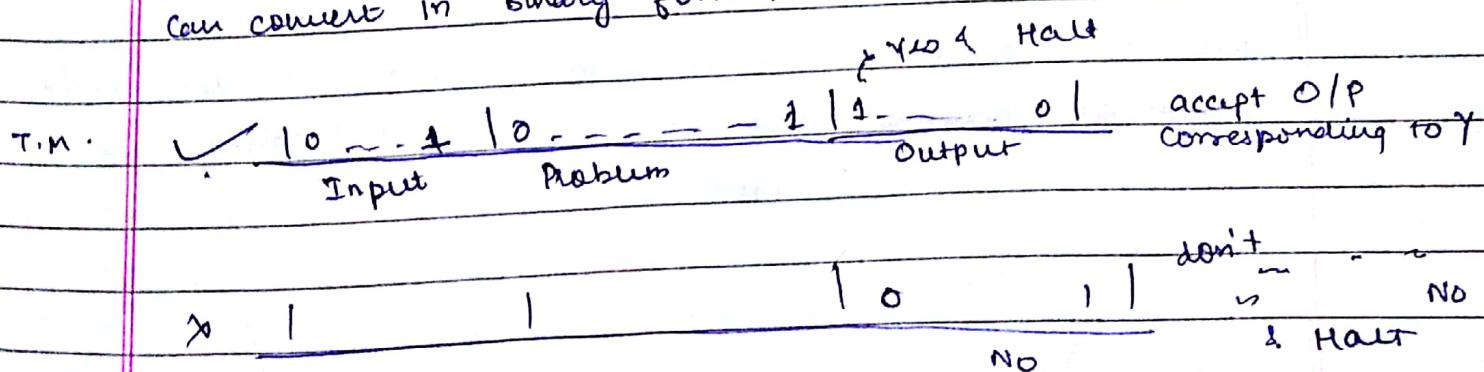
→ Consider problem 1 :

It'll end \Rightarrow halt with Y/N

decidable

TM that you can design
decision problem will
always end with Y or N.

Can convert in binary form :



This T.M. becomes recursive

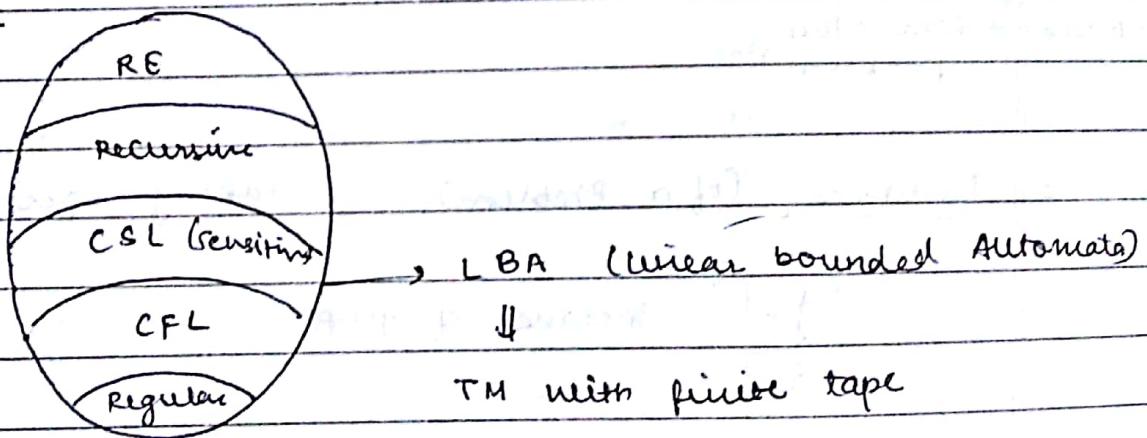
Teacher's Signature

T.M. : decidable
 Accept → don't accept
 Yes No : O/p (L is accepted or not)
 & halts in all cases

↪ suppose for Yes → halts in (T.M.)
 NO → goes to ∞ loop

⇒ Undecidable (Recursively Enumerable)

Hierarchy:



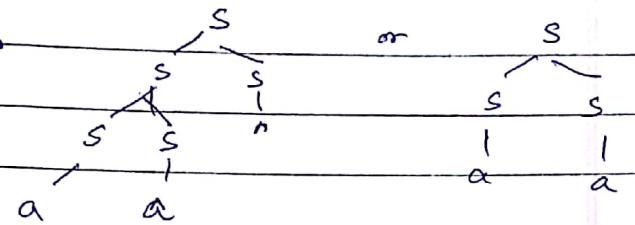
→ Linearly bounded: tape size can grow by linear function on I/p

You can't decide design an algorithm which decides whether another algorithm halts or not of Halt Problem
 Undecidable

Ambiguity

↳ $S \rightarrow aSb$) \Rightarrow Unambiguous
 $S \rightarrow ab$ \rightarrow n-1 times (to get $a^n b^n$)
 1 time

↳ $S \rightarrow SS$
 $S \rightarrow a$) \Rightarrow ambiguous



Ambiguity of CFL \Rightarrow Undecidable problem (AMB)

\rightarrow If it is ambiguous \Rightarrow it'll say Yes
 But can't say No : Undecidable.

G is a CFG

Whether G is ambiguous or not : Undecidable

Reducability :

P_{known} \rightarrow Undecidable (know that it's undecidable)
 \rightarrow To prove " (by reducing P_{known} to P_{unknow})
 P_{unknow} "

Reduction method

Instance of P_{known} $\xrightarrow{\text{Reduce it/}} \text{Instance of P}_{\text{unknow}}$ ①

If you're able to show this :

Unknown

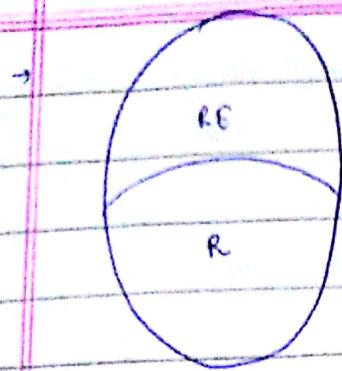
~~If P_{unknow} is Undecidable~~
~~exists ①~~

P_{known} \sim P_{unknow} (Reduce P_{known} to P_{unknow})
 \downarrow
 Undecidable \Rightarrow Undecidable

Teacher's Signature

25/9/18

DATE / /
PAGE NO. / /



There are cases that

L ⊂ RE

But $\overline{L} \notin RE$



RE is not closed under complement

R : is " " " "

(T)

Universal TM : can take another TM as an input & simulate same as the one taken as input (T)

(T)

L/R

$$s(q_i, x_j) = (q_k, x_k, z_1/z_2)$$

can be
written as

encoding of T.M.

$110^i 1 0^j 1 0^k 1 0^l 1 \overline{0} 0 \# 1$: in binary form

$T \leftarrow [t] \xrightarrow{\text{in/p}}$

$d_T \rightarrow \{0, 1\} \xrightarrow{\text{ }} \{0, 1\}$

so, whatever is accepted by t, can be accepted by T.

↳ all binary no. can encode a T.M.

↳ No. of T.M. in world : countably infinite

Table 1

	w_1	w_2	w_3	w_4	→ strings accepted by a T.M.
T_1	1 (Accepted)					
T_2		0 (Not accepted)	1			
T_3			0			
T_4				0		
⋮						
T_e						→ can be big matrix

↳ forming another T.M.

Define new language:

$L_d = \{w_i \mid w_i \text{ is not accepted by } T_i\}$

\Rightarrow NOT R.E.

↓
diagonal where 0 is +nt.

$\bar{L}_d = \{w_i \mid w_i \text{ is accepted by } T_i\}$

↓
diagonal with 1 as entry

i) claim : L_d is not R.E.

Let L_d be R.E.

$$\exists M \text{ s.t. } L(M) = L_d$$

\downarrow
T.M.

then by table T, $\exists \lambda$ s.t.

$$T_\lambda = M$$

{ Because table has list of all T.M.
& M is one of them }

$w_i \in L_d \Rightarrow w_i$ is accepted by $M(T_\lambda)$



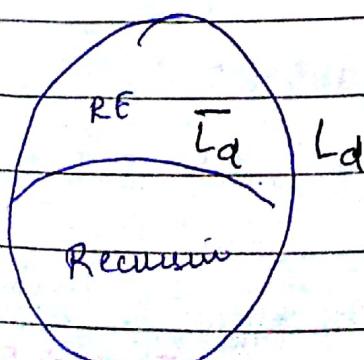
But by table T, w_i should not be accepted by T_λ
(by construction of L_d)

By contradiction,

$L_d \not\models \text{RE}$

Ld is not RE

2) claim : \bar{L}_d will be R.E.



Teacher's Signature

Halting problem (By contradiction)

Suppose \exists such a T.M. T

$d_T \mid t$

check if t halts at all i/p
as to input will halt or not

universal
check if T halts which

on
 t

claim: Given a T.M.
it'll halt.

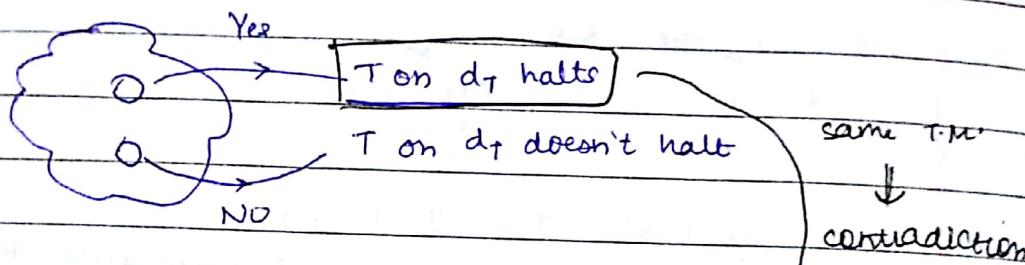
(t halts)

Yes Halts T on t halts

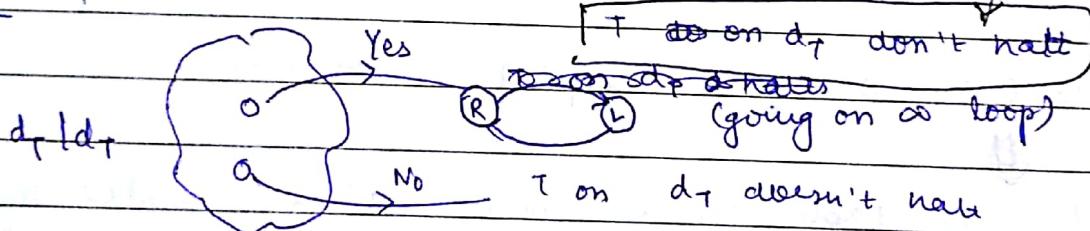
No Halts T on t doesn't halt
(t doesn't halt)

$t \rightarrow$ could be any T.M.

Copy-halt: $d_T \mid d_T \rightarrow$ consider t as d_T itself.



Contradict



* Halting Table is undecidable.

267/4/18

2 Post Correspondence Problem (PCP) \rightarrow also an undesirable problem.

2 sequences :

A = w_1, w_2, \dots, w_k

where $w_i, x_j \in \Sigma^+$

B = x_1, x_2, \dots, x_k

This
not

Ques? \rightarrow Is there a subseq. of $\{1, 2, \dots, k\}$:

i_1, i_2, \dots, i_m where $1 \leq i_j \leq k$ s.t.

$$w_{i_1} w_{i_2} w_{i_3} \dots w_{i_m} = x_{i_1} x_{i_2} \dots x_{i_m} \quad [\text{concatenate}]$$

Teacher's Signature

Eg.

	A	B	$k=3$ min
1	1	111	
2	10111	10	
3	10	0	

i₁ i₂ i₃ i₄

seq. : 2113

$$(10111)(1)(1)(10) = (10)(111)(111)(0)$$

Ⓐ ⓒ

? + a solⁿ for this

Eg.

	A	B
1	10	101
2	011	11
3	101	011

can't start with 0 or 3

0 1 1 0

? have to start with 1 only

should be 1 → 1 or 3.

→ greedy approach

4. 10 1 01 00

B 10 1 01 11 01
↓ ↓
fixed by 1

always you'll apply 3rd rule to get same strings but will always get 1 char.

extra w.r.t B.

This we can comment,
not system

? This problem has no solⁿ

? Does it have a solⁿ : is undecidable

Halt
N
Teacher's Signature
no loop

semantic :- applying rule on strings in last problem
→ system can't take inference from it
syntactic :-

DATE:	/ /
PAGE NO.:	

By red",

instance of Halt problem $\xrightarrow{\text{reduced}}$ instance of PCP

↓
since it is undecidable \rightarrow this'll also be undecidable

Other Undecidable Problems:

3. Checking Whether CFG is ambiguous
4. Equivalence of TM
5. emptiness : given a TM, language accepted by it is empty or not.

All these 5 are semantic properties.

⇒ TM can check syntactic, not semantic properties

Rice Thm :-

All non-trivial semantic properties of a TM are undecidable

property

Trivial Property

↳ won't be able to distinguish

Non-Trivial Properties

↳ if ∃ a T.M. which accepts it

& ∃ --- don't

↳ will be able to distinguish

↳ All above 5

S M : L(M) is RE ?

↓

we don't have any tm. which
won't accept it.

NP: } Problems for which if a non-deterministic TM
that solves in polynomial time }

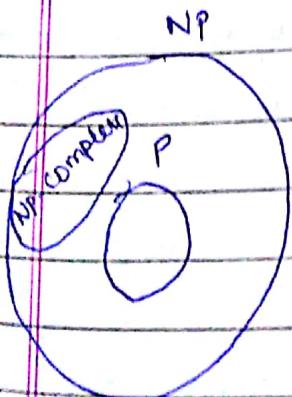
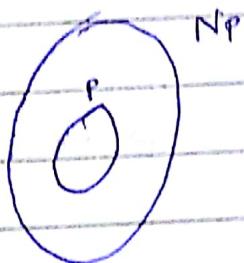
P: } Problems for which if a deterministic TM
that solves in polynomial time }

As per ability, if I can do \Rightarrow other can also do
 $\text{NDTM} \Leftrightarrow \text{DTM}$

Here, we're talking about how fast it can work.
 $\text{so, } \text{NDTM} \Leftrightarrow \text{DTM}$

$$P \subseteq NP \quad (\because DTM \subseteq NDTM)$$

$P = NP$ or $P \neq NP$?? : Open problem



NP-complete: Language L (or problem)

is said to be NP-complete if

(1) $L \in NP$

(2) $\forall L' \in NP$

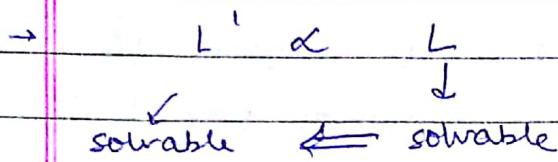
$L' \leq L$
can be
reduced
to
instance
of
language
in
polynomial
time & costed

Reduced
instance \rightarrow instance B

Teacher's Signature

decidability/un... : \rightarrow checking if solvable or not
P / NP : \rightarrow solvability time complexity

DATE:	1/1/1
SERIES	
PAGE NO.:	



NP-Complete : most diff. problem & interlinked

If you can solve 1 in polynomial time, then

$$\underline{P = NP}$$

i) SAT : satisfiability problem

\hookrightarrow Cook's theorem

$$(x_1 \vee x_2) \cap \overline{x}_3 \models T$$

\hookrightarrow Given a boolean formula, whether we get True as answer or not

\hookrightarrow " " with n variables, possible sol's = 2^n

if non-deterministic TM : can check in ND way

\hookrightarrow only verification needs to be done

Given a truth stat assignment : need only to check whether it satisfies or not (in polynomial time)

In DTM : haven't found till yet.

Cook's Thm : SAT is NP-complete

\downarrow
decidable

\uparrow
not time complexity

Given a problem Π , need to show prove : Π is NP-complete
steps :

$$1) \Pi \in NP$$

$$2) \exists L' \in NP \quad L' \leq \Pi$$

We already know SAT is NP-complete

\Rightarrow 2) cond'n becomes :

$$2) SAT \leq \Pi$$

