DATE : / /
PAGE NO. :

Books:

- Theory of Computation by Michael Sipser
- Modern Approach to Computational Complexity by Sanjeev Arora & Boaz Boruk

Marks :

Assignments

Quizes

Exams

Pre - requisite :

Formal language & Automata

Discrete Mathematics

DSA

I. Computability — Sipser

II. Complexity — Sanjeev

DATE : 4 / 1 / 19
PAGE NO. :

Turing Machine

$M = (\Gamma, \Sigma, \gamma, q_0, q_{accept}, q_{reject}, \delta)$

- Γ: tape alphabet (blank)
- Σ: input alphabets
- δ: transition func^n

$\Sigma \cup \gamma \subseteq \Gamma$

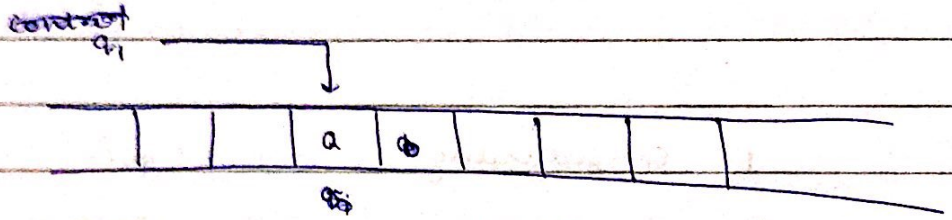
If T.M. doesn't go to any q_{accept} or q_{reject} → It's looping

Recursive : T.M. has to halt

T.M. Recognizable lang. : don't know whether it'll halt or not

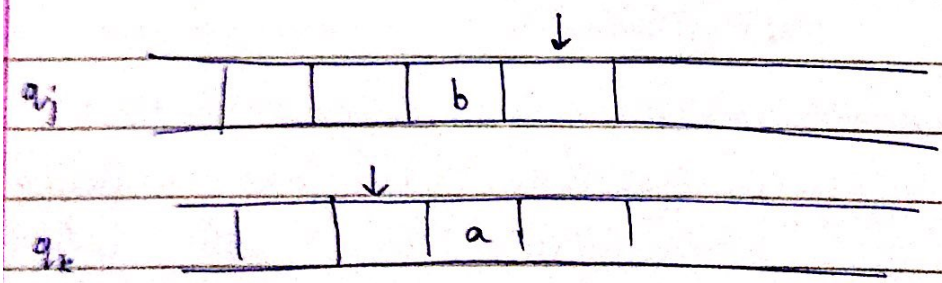
$\delta : (Q \times \Gamma^*) \longrightarrow (Q \times \Gamma^* \times \{L, R\})$

(string of alphabets)



$\delta(q_i, a) \longrightarrow (q_j, b, R)$

OR → (q_k, a, L)



DATE : / /
PAGE NO. :

Non-deterministic : Have multiple choices

↳ $\delta(q_i, a) \in \{ (q_1, b, R), (q_k, a, L) \}$

→ Deterministic language $\subseteq$ Non-deterministic.

DFSA $(\equiv)$ NDFA

DCFL $(\not\equiv)$ NCFL

D T.M. $(\equiv)$ N T.M. (Time might ↑) (Read how to convert)

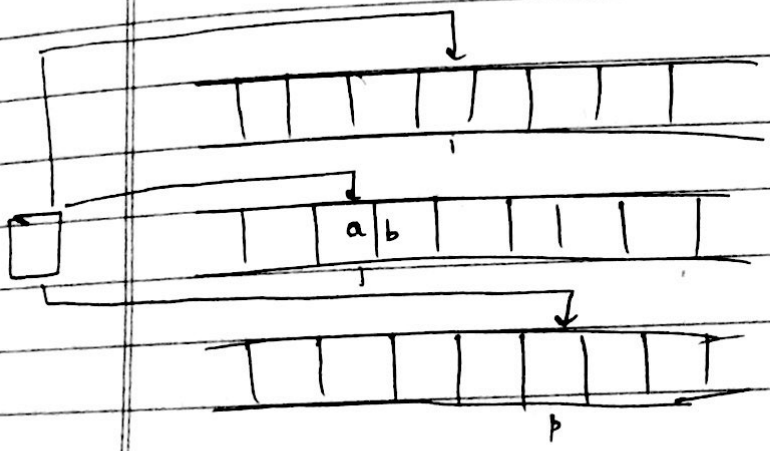
- Many variants of T.M. :

→ Multitape

→ One side opened , etc.

① Multitape T.M. $\Leftrightarrow$ single tape T.M.

k = 3



k tape

Model s.t. input string always comes in 1st ~~str~~ tape

pointer here

w # $\bar{a}$ # . . . . .

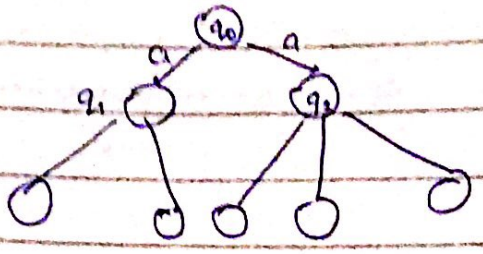
tape 1

If we need some more space here, can shift this part to right

DATE : / /
PAGE NO. :

Add more states (like NFA → DFA)

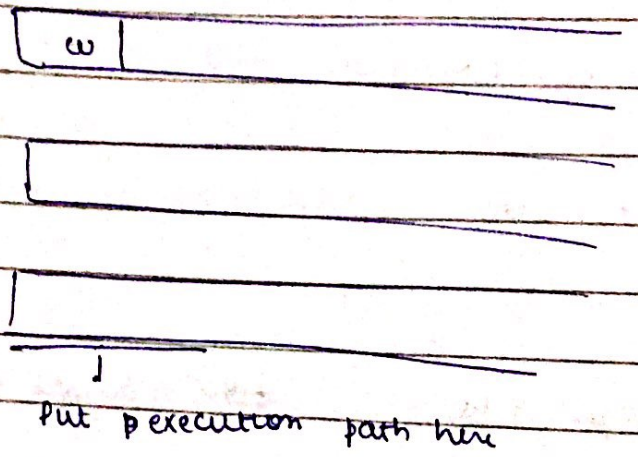
③ Non-Deterministic T.M. ⟺ Deterministic T.M.



→ Tree-like structure

→ Each path is unique in this tree.

Convert to | Multitape T.M.



Put execution path here

→ We read w & the path from 3, we'll execute it here & find whether it is acceptable or rejectable.

Acceptance & Rejec" will occur at leaf nodes.

~~For Zero~~

→ ~~We should know that Tree may be~~

→ Problem : If have an infinite path

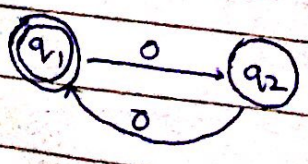
④

Language accepted by TM M : $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$

Ex. $L = \{0^{2k} \mid k \geq 0\}$

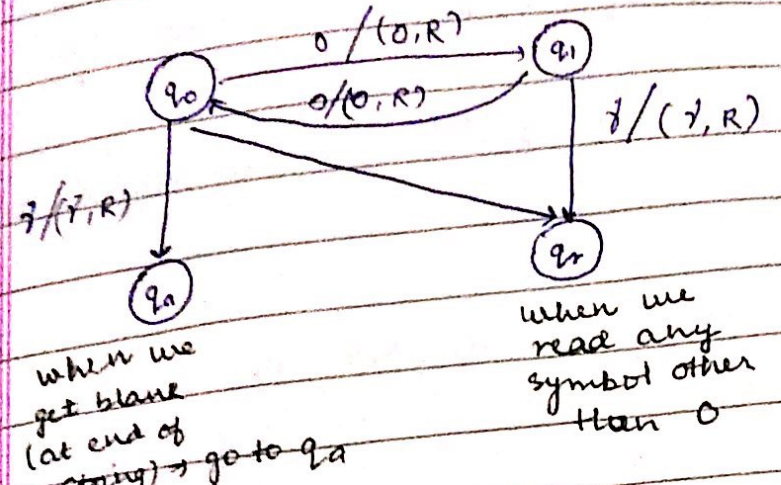
$L = \{00, 0000, \ldots\}$ $k=0 \Rightarrow$ Blank String

DFA :



DATE : / /
PAGE NO. :

In case of TM :

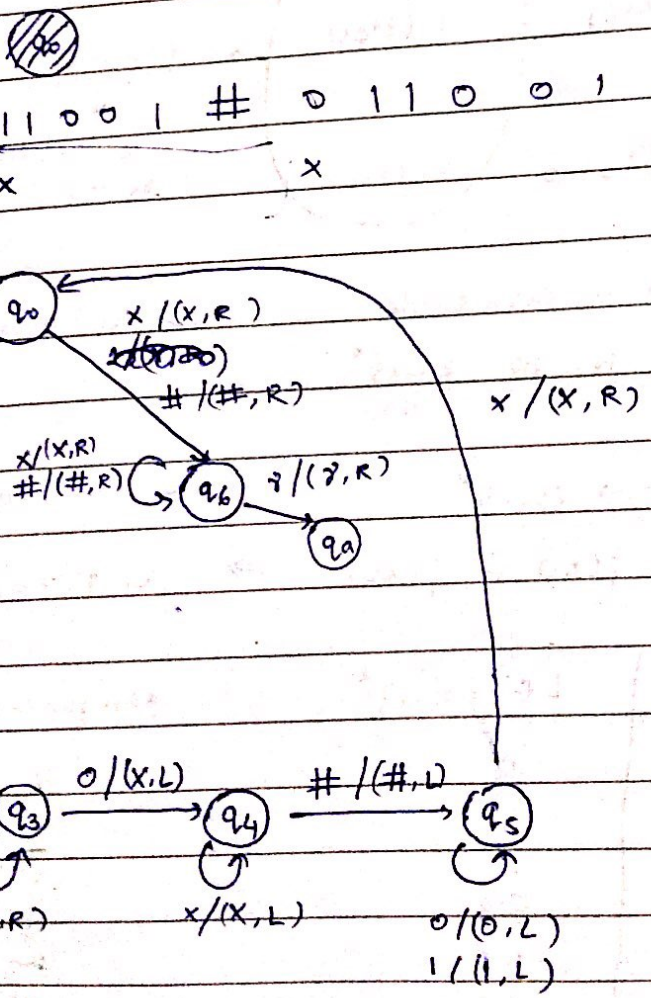


when we get blank (at end of string) → go to qa

when we read any symbol other than 0

$M = (\{q_0, q_1, q_a, q_r\}, \{0\}, \{0\}, q_0, q_a, q_r, \delta_1)$

Ex. $L = \{ w \# w \mid w \in \{0,1\}^* \}$ abb#abb



. Do the same for 1 also

. Not showing

DATE : / /
PAGE NO. :

5/11/19

$M = (Q, \Gamma, \Sigma, \delta, q_0, q_a, q_r, \$)$

(Binary) encoding of TM : [can be done in many ways]

$Q = \{q_0, q_1, \ldots q_m\}$ $\quad \Gamma = \{a_0, a_1, \ldots a_n\}$

$\$$: $t=0$: L, $t=1$: R

$\delta(q_i, a_j) \longrightarrow (q_k, a_l, \Delta_t)$

$c_{ij} = 0^{i+1} 1 0^{j+1} 1 0^{k+1} 1 0^{l+1} 1 0^{t+1}$ → One way of encoding

→ Description of TM M can be denoted by :

$\langle M \rangle = 111 C_{01} 11 C_{02} 11 C_{03} 11 \ldots\ldots 11 C_{mn} 111$

- For each TM, we get 1 binary encoding
- We will take care of only those binary no. which are codes for TM.

→ $M_b \rightarrow L(M_b)$ → set of encoded $\quad b \in \{0,1\}^*$

b	M_b	$L(M_b)$
$b=00$	M_{00}	$L(M_{00})$
$b=01$	M_{01}	$L(M_{01})$
⋮	⋮	⋮
$b=0110$	M_{0110}	$L(M_{0110})$

→ Countable

⇓

S is countable iff ∃ bijection from $\mathbb{N}$ to S

$\mathbb{N} \longrightarrow S$

- To prove $L(M_b)$ is countable, we have to prove there's a bijection from N to $\underset{(b)}{\{0,1\}^*}$

bijective

$f : \mathbb{N} \longrightarrow \{0,1\}^*$ (One-One and Onto)

One-One : $f(x_1) \neq f(x_2) \Rightarrow x_1 \neq x_2$

N	$b \in \{0,1\}^*$
1	0
	1
2	00
	01
	10
	11

assignment to N

1 → λ

2

3 → mapping $2^1 + 2^0$

4

5

DATE : / /
PAGE NO. :

. In case of Real no. ⇒ Uncountable : Proof by Diagonalization

$\mathbb{N}$	$\mathbb{R}$
1	$a_{11}\ a_{21}\ a_{31} \ldots$
2	$a_{12}\ a_{22}\ a_{32} \ldots$
3	$a_{13}\ a_{23}\ a_{33} \ldots$
⋮	⋮
	$x_{1k}\ x_{2k} \ldots x_{kk} \ldots$
⋮	⋮
j	$a_{1j}\ a_{2j}\ a_{3j} \ldots$

$a_{ij} \in \{0, \ldots 9\}$

Aim is to construct a no. $\in \mathbb{R}$ but $\notin$ above a_{ij} set.

⇓

we can select any no. ~~from~~ s.t.

$b_{ii} \in \{0, 1, 2, \ldots 9\} \setminus \{a_{ii}\}$ [no. b/w 0 to 9 which is not equal to a_{ii}]

↓ set minus

Now, we have to prove b_{ii} doesn't lie anywhere in the set $\{a_{ij}\}$.

Proof by Contradiction :

let us assume that $b_{ii} \in$ set (above defined)

k^{th} position of k^{th} no.

$x_{1k}\ x_{2k} \ldots$ x_{kk} [ ]

↓ we don't know about these no.

But we can say that

$b_{kk} \neq x_{kk}$

Hence Proved.

↳ In general, Language is set of strings (binary) : Uncountable
(no. of infinite binary strings)

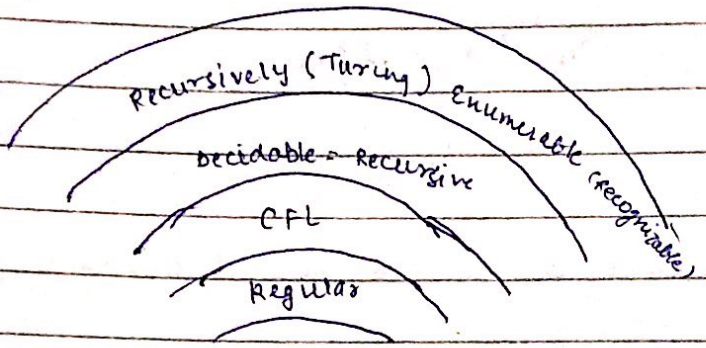
But set of lang. accepted by T.M. are countable

11/1/19

DATE : / /
PAGE NO. :

(Proof) : ① $L' \triangleq$ set of languages recognized by TM → Countable
② $L \triangleq$ set of languages on alphabets $\{0,1\}$ → Uncountable
③ $B \triangleq$ set of Binary strings → Uncountable

$L' \subset L$ (There are lang. not recognized by T.M.) (uncomputable)



Proof of ② :

We know ① & ③, we have to show ② & ③ are of same size, ie, define a bijective func^n

bijection $f : L \longrightarrow B$

Assume $A \in L$ $\quad A = \{0, 00, 001, \ldots\}$

$\Sigma^* = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

Construct Characteristic sequence of A $\quad X_A = \{1, 0, 1, 0, 0, 0, \ldots\}$

(Σ^* of A: if 0 is +nt in A; if −nt in A)

$\Rightarrow f(A) = X_A \in B$

↓

well defined func^n.

Now have to prove this is one-one & onto.

By seeing the construc^n :

if A & A' are diff, X_A & $X_{A'}$ will also be diff. (One-One)

Onto : whenever we've 1 in X_A : put in A

0 — : not —

$\therefore$ B is uncountable $\Rightarrow$ L is also uncountable (due to bijective mapping)

Hence Proved

DATE : / /
PAGE NO. :

Ex: $D = \{ b \in \{0,1\}^* \mid b \notin L(M_b) \}$ → lang. recognized by TM M_b (machine encoding b)

taking string $\{0,1\}^*$ which is not accepted by its own TM

D : non-Turing recognizable

Proof by Contradiction :

Assume that D is recognized by some T.M. M

$d = \langle M \rangle$ (encoding) $\qquad D = L(M_d)$

$d \in D \iff d \notin L(M_d)$ ← from language D's definition

By contradiction, ...

Hence Proved.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	. . . . .	$\langle M_n \rangle$	. . .
M_1	a → dna	dna			a	
M_2	dna	a → dna			dna	
M_3	a	dna			a	
⋮						
M_n	dna	a			a → dna	
⋮						

If a TM accepts its own description ($\langle M \rangle$) : write A, else : dNA

→ We'll create a behavior which is not accepted by any TM (using diagonalization)

In diagonal : replace a ⟶ dna & vice-versa

This behavior didn't exist in the table originally, so, no TM can accept this behavior. (we did this in last class $b_{kk} \neq a_{kk}$)

Deterministic FA, CFL & Decidability

DATE : / /
PAGE NO. :

Universal language

encoding ($\langle M \rangle w$)

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$ → Turing Recognizable

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

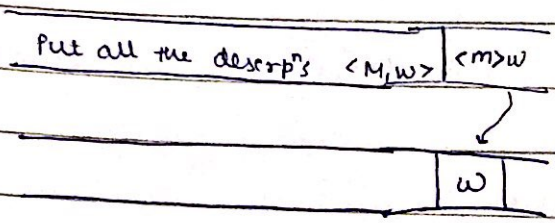
$HALT_{TM} \subset A_{TM}$

Prove that A_{TM} is recognized by ATM but it is undecidable

→ Universal TM recognizes A_{TM}

We will take 3-tape T.M.

UTM



1st have to check if proper encoding (Finite string)

Yes

Put w here

Can simulate now (in 3rd tape)

* UTM can simulate any type of string w

→ A_{TM} is not decidable.

Contradiction ;

Let us assume A_{TM} is decidable.

1/1/19

Empty TM

Q. Prove that $E_{TM} = \{ \langle M \rangle \mid L(M) = \phi \}$ → undecidable

Proof by contradiction :

Assume that E_{TM} is decidable.

R is a decider TM for E_{TM}. (R decides whether language recognized by M is empty or not)

from R, we construct decider S of A_{TM} (Imp. to learn how to convert)

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$

DATE : / /
PAGE NO. :

We define : (modified TM of A_{TM})
[M_1 accepts only w]

$M_1(x)$:
if $x \neq w$ reject —①
if $x = w$ run M on w & do according to M. —②

(We are converting M into TM M_1) ↳ of A_{TM}.
M_1 will reject every string other than w
So, $L(M_1) = \{w\}$
$\Rightarrow L(M_1) = L(M)$

Decider for A_{TM} (S)

Run $R(\langle M_1 \rangle)$
if R accepts $\Rightarrow$ $L(M_1)$ is empty
(this means M doesn't accept w) (from ① & ②)
↳ rejects (S)

We're running R as a sub-routine for S.

if R rejects $\Rightarrow$ M accepts w $\Rightarrow$ accepts

~~for a given M & w~~

→ In b/w, we're running a TM which is converting M to M_1.

reduced E_{TM} to A_{TM}

$\Rightarrow A_{TM} \subseteq \bar{E}_{TM}$ { We've reduced to $\bar{E}_{TM}$, reject → accept
but E_{TM}: decidable $\Rightarrow$ $\bar{E}_{TM}$: decidable
& E_{TM}: un " $\Rightarrow$ " : un -
(wrt decidability, they're equivalent)

A_{TM} undecidable

By Rice Theorem, it is undecidable.

Ex. $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid$ TMs M_1 & M_2 s.t. $L(M_1) = L(M_2)\}$ ↳ Undecidable
↳ equivalent TM

↓ we want to change this & bring some emptiness cond" here (so as to apply decider of E_{TM})

$\{\langle M', M'' \rangle \mid L(M') = L(M_1) \overset{\text{set diff}}{\setminus} L(M_2)$ & $L(M'') = \phi\}$

↳ NOW, we'll check whether $L(M')$ & $L(M'')$ are equivalent
$L(M')$ exists (because ~~state~~ L(M) is closed under union, ...)

Logical formula : words on symbols (given below)

$\{\neg, \vee, \wedge, \rightarrow, \forall, \exists, f_i, 1 \le i \le N, (,)\}$

$\{p, p \rightarrow q\} \vdash q$ (If given p & $p \rightarrow q$, we can determine q)

set of premises ↝ conclusion

→ $\{P_1, P_2, \ldots P_n\} \vdash Q$

P_1

S_1 ⟶ set of statements : **Proof**.

S_2

P_2

⋮

Q (finally get this)

→ $x^n + y^n = z^n$; $n \ge 3$: No sol^n in $x, y, z \in \mathbb{N}$

Rice Theorem

Ex. · C = set of languages

$L_C = \{\langle M \rangle \mid L(M) \in C\}$ ⟵ empty / it contains all $\langle M \rangle$ / undecidable ⟵ can be concluded

if C : set of CFL

L_C : set of encoding of all TM which accept CFL.

Proof by contradiction :

Assume : L_C is decidable, non-empty, and doesn't contain all $\langle M \rangle$.

(We'll reduce this TM into A_{TM})

→

$\langle M, w \rangle \in A_{TM} \Leftrightarrow \langle M' \rangle \in L_C$

Luca Trevisan
computability & complexity → Read Rice Theorem

DATE : / /
PAGE NO. :

Define $M''(x)$ → Run M on w if M ~~accepts~~ doesn't accept w → ~~accepts~~ reject
→ if M accept w → run M' on x & do accordingly

M accept w
$L(M'') = L(M') \notin M''(x)$

M doesn't accept w
$L(M'') = \phi \notin C \Rightarrow \langle M \rangle \notin C$

23/1/19

$M_{in} \in L_C$

Let * be a decider of L_C.
We've $A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$, without loss of generality
We need one more assump" that $\phi \notin C$. (w.l.g.)

Define :
$M_w(x)$ → M doesn't accept w → does not accept
↓ S
↳ M accepts w, Pick $\langle M_{in} \rangle \in L_C$ & run $M_{in}(x)$ & o/p accordingly
$w, x \in \Sigma^*$ ↓ one element in set L_C

We've to reduce : $\langle M, w \rangle \in A_{TM} \iff \langle M_{in} \rangle \in L_C$

→ Our aim is to ~~the~~ pick a string in A_{TM} & a string in L_C

- if M accepts w and on running x on M_{in}, if M_{in} accepts x → then S M_w accepts x.
whatever o/p Min gives → same o/p is given by S.

↳ M accepts w :
$L(S) = L(M_{in}) \in C \quad \{ \because M_{in} \in C \} \Rightarrow S \in L_C$

↳ M doesn't accept w : S(x) won't accept any x (string)
$L(S) = \phi \notin C \Rightarrow S \notin L_C$

→ L_C : undecidable.

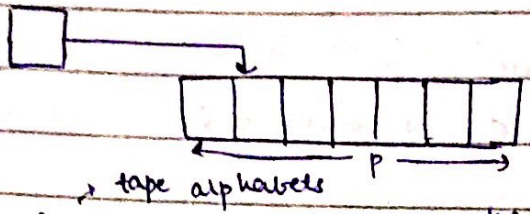
- $\langle M, w \rangle \in A_{TM} \Rightarrow \langle S \rangle \in L_c$

  $\langle M, w \rangle \notin A_{TM} \Rightarrow \langle S \rangle \notin L_c$

$\therefore A_{TM}$ is undecidable
$\Rightarrow L_c$ is also undecidable

Linear Bounded Automata

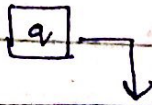
(TM with finite tape)



↳ tape alphabets

$|\Gamma| = m$ → Total no. of strings possible = m^p

$|Q| = s$



→ aXcXqabc ≡ a | X | c | X | a | b | c

↳ a configuration of TM M on some input

$\therefore \delta(q, a) = (q', b, R)$

aXcXbq'bc → Do this to find all possible transitions in LBA (finite tape)

Q. How many such config^{n}s can we create on tape of length p?

$s \, m^p \, p^{p-1}$

↳ state can be at p positions

Halting LBA

No. of config^{n}s ← finite

→ All the problems we had defined for TM can be applicable here.

$A_{TM} \rightarrow A_{LBA}$

$E_{TM} \rightarrow E_{LBA}$

$HALT_{TM} \rightarrow HALT_{LBA}$

* LBA is more powerful than Regular Language & CFL
(have to find a lang. $\in$ LBA but $\notin$ RL / CFL)

* TM M > LBA

18/1/19

Ex. $A_{LBA} = \{ \langle M, w \rangle \mid \text{LBA } M \text{ accepts } w \}$ → decidable

Possible configr's : $q\,m^{p-1}\,p$ (last class)

$|Q| = q$

$|\Gamma| = g$

length of tape : p

Run M on w $q\,g^{p-1}\,p + 1$ steps

(→ decider for A_{LBA}

$S(\langle M, w \rangle)$

↳ this means atleast 1 config" will be repeated

↳ if M accepts w within $q\,g^{p-1}\,p$ steps : accept

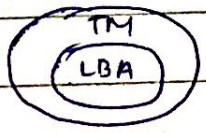
↳ if rejects : reject

↳ otherwise : reject
(looping)

crossing $q\,g^{p-1}\,p$ without accepting/rejecting ⇒ a config" is repeated ⇒ going to a loop

Acceptance of TM : undecidable

" " LBA : decidable



Ex. $E_{LBA} = \{ \langle M \rangle \mid \text{LBA } M \text{ such that } L(M) = \phi \}$ → undecidable

$\langle M, w \rangle \in A_{TM} \iff \langle B \rangle \in \overline{E}_{LBA}$ (we've to reduce it to B & $\overline{E}_{LBA}$: undecidable)

decidability is closed under complement.

$\overline{E}_{LBA}$: und. ⇒ E_{LBA} : und..

B is a LBA s.t. accepting

$L(B) = \{ x \mid x$ is an computation history of M on w$\}$

accepting computation history of M on w

how ?

baqca → baaq'a → ... q_{accept}

$\delta(q, c) \rightarrow (q', a, R)$ config 1 → config 2

(changing config's based on transition fnctn)

$B(x)$: x is input history

$x \Rightarrow \# x_1 \# x_2 \ldots / \# x_{11} \# x_{12} \ldots$ → many possible combi.

B will convert x into

1.) this form & check for any possible modif of x whether x_1 is like $q_0 w$ or not

2) And for 2 consecutive config's, it will check whether it's happening acc. to δ or not

3) will check last config (qaccept)

* We are not giving importance to complexity here. Just on constructing a LBA.

Now we will find whether $\langle B \rangle \in \overline{E}_{LBA}$ or not.

R is a decider of E_{LBA}

R(<B>) { Run R on B }

if R accepts ⇒ L(B) is empty (no string accepted by B) $L(B) = \phi$ → reject (M has rejected w) $\langle M, w \rangle \notin A_{TM}$

if R rejects ⇒ $L(B) \neq \phi$ (atleast a string accepting comput" history of M on w) → accept $\langle M, w \rangle \in A_{TM}$

↓ Reduced to A_{TM}

Since A_{TM} is undecidable ⇒ $\overline{E}_{LBA}$ also undecidable

PAGE NO. :

Post Correspondence Problem (PCP)

dominos $t_i, b_i \in \Sigma^*$

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \ldots, \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

input

- have to arrange dominos in P s.t. we have a matching in t_i's and b_i's
- length of t_i & b_i may be diff.

$$\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \ldots \begin{bmatrix} t_{i_m} \\ b_{i_m} \end{bmatrix} \quad \text{s.t.} \quad t_{i_1} t_{i_2} t_{i_3} \ldots t_{i_m} = b_{i_1} b_{i_2} \ldots b_{i_m}$$

if we get a matching : Answer : Yes

else : Answer : No.

Ex. $|t_i| > |b_i|$ ⇒ Not possible

Ex. $\left\{ \begin{bmatrix} ab \\ ca \end{bmatrix}, \begin{bmatrix} a \\ ab \end{bmatrix}, \begin{bmatrix} ca \\ a \end{bmatrix}, \begin{bmatrix} abc \\ c \end{bmatrix} \right\}$

↓

~~$\left\{ \begin{bmatrix} a \\ ab \end{bmatrix}, \begin{bmatrix} b \\ ca \end{bmatrix}, \begin{bmatrix} abc \\ c \end{bmatrix}, \begin{bmatrix} ca \\ a \end{bmatrix}, \begin{bmatrix} abc \\ c \end{bmatrix} \right\}$~~

$\begin{bmatrix} a \\ ab \end{bmatrix} \begin{bmatrix} b \\ ca \end{bmatrix} \begin{bmatrix} ca \\ a \end{bmatrix}$ — start from same

$\begin{bmatrix} a \\ ab \end{bmatrix} - \begin{bmatrix} abc \\ c \end{bmatrix}$ ✓

$\begin{bmatrix} abc \\ c \end{bmatrix}$ ✗

* Each domino can repeat any no. of times.

Sipser book Exercises Problems

DATE : / /
PAGE NO. :

* This PCP is an undecidable problem.

$t = \#\, C_1\, \#\, C_2 \ldots \#\, q_{accept}$

$b = \#\, C_1\, \#\, C_2 \ldots \#\, q_{accept}$

dominos

↳ if M accepts w : we get a match

else : " don't " "

30-01-19

Proof : Assume match in P is decidable.

(Based on decider of P, we'll try to construct decider S for A_{TM})

$(M, w) \longrightarrow P = \{ [\,], [\,], \ldots [\,] \}$

Let R be a decider of P

→ we'll again look at accepting comput^n history of M w on M.

$S \in M$

$\#\, C_1\, \#\, C_2 \ldots C_i \# C_{i+1} \ldots \#\, C_{accept}\, \#\, (M, w)$

$C_1 = q_0 w$; $C_{accept} = q_{accept}$; (same as in last problem)

→ We want to construct a domino

$\#\, C_1\, \#\, C_2\, \#\, C_3 \ldots \#\, C_{accept}\, \#$

$\#\, C_1\, \#\, C_2\, \#\, C_3 \ldots \#\, C_{accept}\, \#$

$\begin{bmatrix} t_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$

$\begin{bmatrix} \# \\ \# q_0 w \end{bmatrix}$

first dominos

DATE : / /
PAGE NO. :

replacing a with b & moving right

** ∀ q reject / q accept

∀ a, b ∈ Γ q, r ∈ Q

if $\delta(q, a) \rightarrow (r, b, R)$ dominos →

q	a
b	r

if $\delta(q, a) \rightarrow (r, b, L)$ dominos →

cq	a
βrc	b

precious tape alphabet (on left) of a

↓

we have to do it ∀ c ∈ Γ (for each c)

Ex Γ = {1, 2, 3}

dominos : (for left case)

1q	a
r1	b

2q	a
r2	b

3q	a
r3	b

→ we'll add more dominos :

∀ x ∈ Γ

x
x

#
#

↑ for hashes we get in between

#
~~#~~

↑ blank symbol

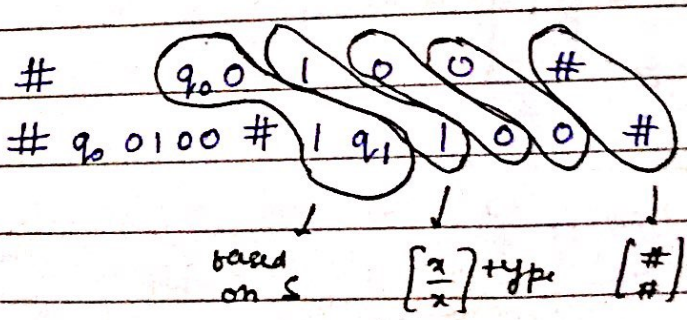
→ ω = 0100

$(q_0, 0) \rightarrow (q_1, 1, R)$

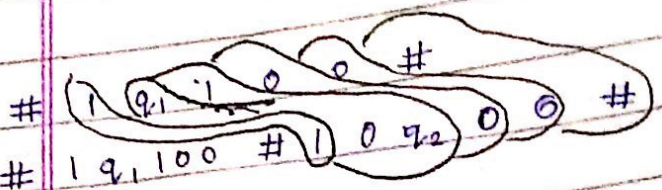
#
q_0 0100

↓

q_0	0
1	q_1



$(q_1, 1) \rightarrow (q_2, 0, R)$



#1q₁100# → #10q₂00#

like this, the string we want can be modified.

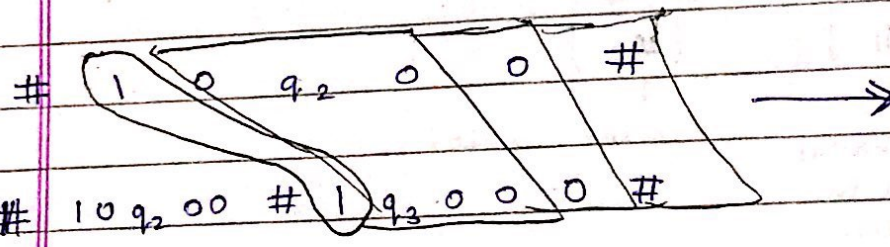
4) If we're left transition:

$\Gamma = \{0, 1, \sqcup\}$ ← blank

$(q_2, 0) \rightarrow (q_3, 0, L)$

$$\begin{bmatrix} cq_2 & 0 \\ q_3c & 0 \end{bmatrix} \rightarrow \left\{ \begin{bmatrix} 0q_2 & 0 \\ q_30 & 0 \end{bmatrix}, \begin{bmatrix} 1q_2 & 0 \\ q_31 & 0 \end{bmatrix}, \begin{bmatrix} \sqcup q_2 & 0 \\ q_3\sqcup & 0 \end{bmatrix} \right\}$$

↓ which domino to use?
have 0 before q_2 (match with $(q_2, 0)$)
↓ previous state



#10q₂00# → #1q₃000#

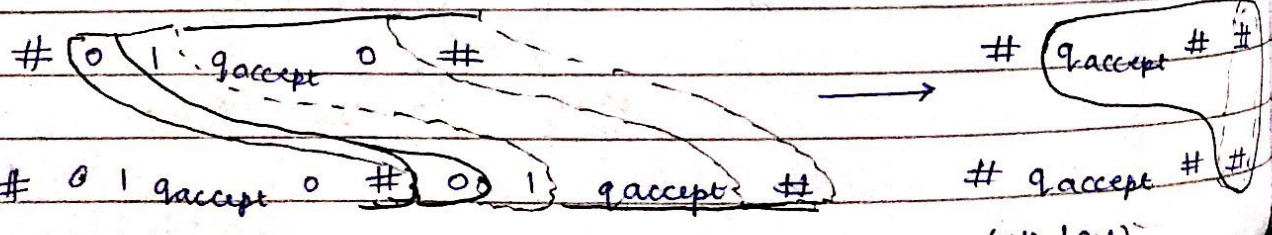
#01q_{accept}0#

(assume this is the string we get at last)

we want to reach on # q_{accept} # string.

so, we'll have to add dominos for this:

$\forall a \in \Gamma$ $\begin{bmatrix} q_{accept}\, a \\ q_{accept} \end{bmatrix}, \begin{bmatrix} a\, q_{accept} \\ q_{accept} \end{bmatrix}$



(at last)

DATE : / /
PAGE NO. :

$\langle M, w \rangle \in A_{TM} \iff$ match in P

$$P = \left\{ \begin{bmatrix} \# \\ \# q_0 w \end{bmatrix}, \begin{bmatrix} q_i & a \\ b & q_j \end{bmatrix}, \begin{bmatrix} Cq_i & a \\ q_j c & b \end{bmatrix} \begin{bmatrix} \# \\ \sqcup \# \end{bmatrix}, \begin{bmatrix} x \\ x \end{bmatrix}, \right.$$

$$\left. \begin{bmatrix} q_{accept} & a \\ q_{accept} & \end{bmatrix}, \begin{bmatrix} a & q_{accept} \\ & q_{accept} \end{bmatrix} \begin{bmatrix} q_{accept} & \# & \# \\ & \# & \end{bmatrix} \right\}$$

↳ Reduces to A_{TM}

$\therefore A_{TM}$ is undecidable $\Rightarrow$ P is also undecidable

4/2/19

Decidability of Logical Theorems

→ Given a statement — True / Provable (Provable has a proof)

Logical symbols : $\exists, \forall, \vee, \wedge, \rightarrow, \leftrightarrow, R, *, ...$

↳ Given a statement, is it true ?
↳ " " is it provable ? } → 2 kind of features

→ Provability → True (If provable → will be true)
But vise-versa is not true (If true → might have a proof)

Ex. A = { } → ~~Probability~~ sample

and mean = 5.

We can guarantee :

$x \in A$, $x \leq 5$

$y \in A$, $y \geq 5$

↳ Provabile → talks about existence of proof of a statement

Ex. $\{p \to (q \to r), p, \neg r\}$ — given; $\neg q$ — want to conclude this

$\{\ \ \ \} \to \neg q$ ① : can check by putting values in truth table & Trueness can be checked this way.

OR

can use existing formulae to derive $\neg q$ → PROOF

$p \to (q \to r)$, p given

we know : $\{p \to q, p\} \to q$

$\{p \to (q \to r), p\} \to (q \to r)$

Also, $q \to r \Rightarrow \neg r \to \neg q$

$\to \neg r \to \neg q$

Proof :

1.	$p \to (q \to r)$	given	Proof of given Statement ①
2.	p	given	
3	$q \to r$		
4.	$\neg r \to \neg q$		
5.	$\neg r$	given	
	$\neg q$		

(3) & (4) are equivalent

→ Statement ① can also be written as;

$\{p \to (q \to r)\} \wedge \{p\} \wedge \{\neg r\} \to \{\neg q\}$

OR

$(p \to (q \to r)) \to (p \to (\neg r \to \neg q))$

OR

- $\{p \to (q \to r), p, \neg r\} \vdash \neg q$ ② (talks about provability)
  - $\vdash$: syntactic entailment
- $\{p \to (q \to r), p, \neg r\} \models \neg q$ (talks about trueness)
  - $\models$: semantic entailment

we can't prove consistency of a system within that system

DATE : / /
PAGE NO. :

Another way to write statement ① :

$\vdash \{ p \to (q \to r), p, \neg r \} \to \neg q$ (If ② has a proof, we can go for this statement)

Refer :

* If a statement has a proof, then it is true (Soundness for Logical systems)
* All true statements are provable & it has a proof (Completeness)

→ 2 types of logics :

1) Propositional : $\vee, \wedge, \to, \leftrightarrow$

2) Predicate : $\forall, \exists$

~~These two logics hold simultaneously.~~

Soundness : $X \vdash Y \longrightarrow X \models Y$

Completeness : $X \models Y \longrightarrow X \vdash Y$

* In both 1) & 2) these above 2 statements hold true.

→ Consistency :

$\vdash A$ (A has a proof) → If both hold true, then our system is inconsistent

$\vdash \neg A$ ($\neg A$ " ")

$\vdash A \wedge \vdash \neg A \equiv \vdash (0 = 1)$ → Inconsistency

$\nvdash (0 = 1)$ → Consistency

$0 = 1$	provable	(I)	$0 \neq 1$	provable	(C)
$0 = 1$	not "	(C)	$0 \neq 1$	not "	~~(I)~~ → Can't say anything (dilemma)

I : Inconsistent , C : Consistent

→ If we move from Logical System → Mathematical System → Logical system

have $\mathbb{N}$ here (unlike earlier)

has many properties $(+, -, *, /, \ldots)$

⇓

This system is more bigger.

- simultaneously, soundness, completeness & consistency can't hold together in ~~logical~~ mathematical systems.

- Consistency : verifiability / checkability

algorithmic sense of Consistency.

$(\langle \phi, \Pi \rangle)$, proof for ϕ

↑ Logical formula / statement

if Π is a proof of ϕ → Accept

else → Reject.

Ex. $\phi = \forall x \exists y \, [y = x + x]$ –

for what values of (x, y) above proof is true ?

because it's used above.

$(S, +)$ → model

$S = \mathbb{C}, \mathbb{R}, \ldots$

Ex $\phi' = \forall x \forall y \, [x \geq y \text{ or } x \leq y]$ → $(R, \geq)$

⇒ $Th(M)$: set of all true statements within the model M.

(Logical formulas)

8/2/19

- $Pr(m, n)$: m is the godel no of a proof of a formula whose godel no is n.

- $P(\underbrace{g(x)}_{\text{godel no } (=n)}) = \exists x \, [Pr(x, n) \wedge x = m]$

- $\vdash x \rightarrow \vdash P(g(x))$ — (1)
  $\vdash P(g(x \rightarrow y)) \rightarrow P[P(g(x)) \rightarrow P(g(y))]$ — (2)
  $\vdash P(g(x)) \rightarrow P(g(P(g(x))))$ — (3)
  $\nvdash P(g(0=1))$: consistent — (4)
  $\vdash A \leftrightarrow \neg P(g(A))$ — (5)

Free variable

$B_0(n), B_1(n) \ldots B_j(n), \ldots B_n(n) \ldots$

$B_k(n) = \neg P(g(B_n(n))) = p$ [let RHS be equal to $B_k(n)$ → assumpⁿ]

↳ [we've proof for $p \rightarrow p \wedge p \leftarrow p$]

$\vdash B_k(n) \leftrightarrow \neg P(g(B_n(n)))$

∴ n is free variable, it can take any value $(\mathbb{N}, +, *)$

we can replace it with k.

→ $\vdash B_k(k) \leftrightarrow \neg P(g(B_k(k)))$

$\vdash A \leftrightarrow \neg P(g(A))$ (let $B_k(k) = A$)

→ $\vdash \underbrace{P(g(A))}_{x} \rightarrow \underbrace{\neg A}_{y}$ — (6) (Contrapositive of above one)

Applying (2), we get

(We have taken just → for simplicity)

(a) — $\vdash P(g(P(g(A)) \rightarrow \neg A))) \rightarrow [P(g(P(g(A)))) \rightarrow P(g(\neg A))]$

Apply (1) on (6)

(b) — ~~$\vdash P(g(P(g(A)) \rightarrow \neg A))$~~

$\vdash (P(g(A)) \rightarrow \neg A) \rightarrow \vdash P(g(P(g(A)) \rightarrow \neg A))$

DATE : / /
PAGE NO. :

(a) & (b) of form : $p \to q$, p ~~has~~ have proof → q also has proof

→ $\vdash P(g(P(g(A)) \to \neg A))$ —(c)→ again, taking (a) & (c),

→ $\vdash P(g(P(g(A))) \to P(g(\neg A))$ ← this has proof

← (d)

let $P(g(A))$ has a proof

$\vdash P(g(A))$

Apply (1) on this.

→ $\vdash P(g(A)) \to \vdash P(g(P(g(A))))$ —— (e) $y \& z$, $x \& y$

Apply $[(x \to y) \wedge (y \to z) \to (x \to z)]$ on (d) & (e).

we conclude that

$$\boxed{\vdash P(g(A)) \to P(g(\neg A))} \quad \text{—— (6)}$$

→ $\vdash \underset{x}{A} \to (\underset{Y}{\neg A \to (0=1)})$ suppose it's true $[(A \wedge \neg A) = (0=1)]$

Apply (2) ← (7)

$\vdash P(g(A \to (\neg A \to (0=1)))) \to [P(g(A)) \to P(g(\neg A \to (0=1)))]$

Apply (1) ~~stigues~~, apply $p \to q$. q wala concept ~~factors~~ (like earlier), we get

$\vdash P(g(A \to (\neg A \to (0=1))))$

again, $p \to q$ & $p \to \vdash q$

$x \to Y$

→ $\vdash P(g(A)) \to P(g(\neg A \to (0=1)))$

Apply (2) on 2nd part $\{(x \to y) \wedge (y \wedge z) \to (x \to z)\}$

$\vdash P(g(A)) \to [P(g(\neg A)) \to P(g(0=1))]$

Using $[x \to (y \to z) \equiv (x \to y) \to (x \to z)]$

$\equiv \neg x \vee (y \to z)$

if $x : T$ → Trueness will be determined by $y \to z$

if $x : F$ → always will be true

same here.

→ same here

$\vdash (P(g(A)) \to P(g(\neg A))) \to (P(g(A)) \to P(g(0=1)))$

↓ have already derived this part

$p \to q$, p & q also has proof

$\vdash P(g(A)) \to P(g(0=1))$ — (7)

Ex. $\vdash A$ (A has proof)

$\vdash P(g(0=1))$ From (7) & (1)

↳ From (4) → there's a contradiction

$\to \not\vdash A$

$\not\vdash A$ — (8)

Ex. $\vdash \neg A$

$\vdash P(g(A))$ Apply (5) ka Contrapositive ($\neg A \to P(g(A))$)

$\vdash P(g(0=1))$

↳ Again, Contradiction

$\not\vdash \neg A$ — (9)

Consistency

Ex. $\vdash \neg P(g(0=1))$ (0=1 is provable doesn't has a proof)

Apply (7) [Contrapositive]

$\vdash \neg P(g(A))$ (Apply (5))

$\vdash A \to$ Contradiction

$\not\vdash \neg P(g(0=1))$ — (10)

For no. : Piano Axiom

for set theory : ZFC

DATE : / /

PAGE NO. :

Conclusions :

→ If some statement A has no proof, its negation also has no proof.

→ There exists a statement which has no proof (maybe T or F)

↓

No completeness

→ ~~If~~ we can't prove the consistency of system within system, if we assume (A) is consistent ∀ for the system (A0)

Axiom (assumed to be true) [don't have proofs]

→ Euclidean geometry : has 5 axioms

Set theory works on : Axiom of Choice

↓

Cartesian Product of 2 non-empty sets is always non-empty → contradicted by Russel Paradox

↓ modified

ZFC ~~Axiom~~

within ZFC, you can't prove consistency ~~by~~ but you may if you add something else in the system (ZFC)

↳ can add some axioms here.

Will Cross the system $(\mathbb{N}, +, *)$

→ $|\mathbb{R}| > |\mathbb{N}|$: Can we find an infinite set S which lies b/w $\mathbb{R}$ & $\mathbb{N}$? No → Continuum Hypothesis

$|\mathbb{N}| < S < \mathbb{R}$

11/02/19

DATE : / /
PAGE NO. :

(won't come in exam)

Recursion Theorem

→ can be programs or code..

theorem
1. Construct a T.M which prints its own description.
2. Compute : with the printed description (obtained above)

→ A code which prints its own code. Can we write such code?

For T.M.

1 $q : \Sigma^* \longrightarrow \Sigma^*$

$q(w)$ is the description of TM P_w which prints w.

P_w (for any input) : erase tape, print w

$q(w) = \langle P_w \rangle$

→ $A = P_{\langle B \rangle}$ A is TM which writes (print) descrip^n of B

B(M) : a TM 'M' ↵ Compute $\overrightarrow{q(\langle M \rangle)}$ → descrip^n of TM which prints descrip^n of M.

print $q(\langle M \rangle) \langle M \rangle$ & halt

$\langle S \rangle = \langle A B \rangle$

A will print : $\langle B \rangle$

B will take ↓ as input & $q(\langle B \rangle)$: TM which prints B + A

$B(\langle B \rangle) = q(\langle B \rangle)\langle B \rangle = \langle A \rangle \langle B \rangle$

→ Given a language, can we write a code that prints its own code?

A : prints <B> $\langle S \rangle = \langle AB \rangle$: S is a T.M. which prints the over description

B(<B>) : q(<B>)(<B>)
= <A><B>

DATE : / /
PAGE NO. :

```
int main () {
char f = " int main () { char f = %c%s%c; printf (f,34,f,34); }";
printf (f, 34, f, 34);
}
```

<B> (this is a string)

A B(Machine is which is executed)

→ This code will write ~~its~~ ~~of~~ print its own code

2. Now, we'll see how to compute something from above code

$\forall$ T : $\Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$

$\exists$ R : $\Sigma^* \longrightarrow \Sigma^*$

for all 2-input TM T, $\exists$ a TM R which takes same input as T st

$$R(w) = T(\langle R \rangle, w)$$

Ex: T: Universal TM
(simulate TM R on w) — ①

Ex → if T(s·t) = st (just concatenates input strings), then

R(t) = T(<R>, t)
= <R>t

R(w) = UTM(<R>, w) — special case of ①

Ex → if T(s,t) = Print s.

then R(t) = T'(<R>, t) → will print <R>

using this,

→
```
R(t) {
  get <R>        can get <R>
  T(<R>, t)      and work on R(t) { T(<R>, t) }
}
```

Sir simulation
Anderson

DATE: / /
PAGE NO.:

Ex. $T(a,b)$ = TM b simulates a (replaced w with TM) $b(\langle a \rangle)$.

$R(\langle M \rangle) = T(\langle R \rangle, \langle M \rangle) = M(\langle R \rangle)$: gof = fog

↓ according to our definition

Proof of 2 : $\langle A \rangle = P_{\langle BT \rangle}$

$B(\langle BT \rangle) = q_{\langle BT \rangle}\langle BT \rangle$

$= \langle A \rangle \langle BT \rangle$

Q Prove A_{TM} is undecidable using Recursion Theorem.

$\{\langle M, w \rangle : TM\ M \text{ accepts } w\}$ Assume decidable & H is a decider

$B(w)$ → (want to check whether it's decidable or not)

get $\langle B \rangle$

Run H on $\langle B, w \rangle$ & do opposite (if H accept : reject ; reject : accept)

Contradiction (same as older proof)

13/02/19

Kolmogorov Complexity

→ Given a string, how to compress it → model it into TM

→ Find min. length string which has info about original string

string u

$S_1 = 0111010011110110$ ... → For this, we can't find any pattern

$S_2 = 011011011011$ ... → $(011)^i$ copy i times

Kolmogorov Complexity of S_2

Can we find min length string from original string u.

DATE : / /
PAGE NO. :

Def^n → Given a binary string x_0, the minimum length description of string x_0, denoted by $d(x)$, s.t. ∃ a string $\langle M, w \rangle$, if M runs on w then it halts and puts x_0 in tape.

$$d(x) = \langle M \rangle w = \langle M, w \rangle$$

$$K(x) = |d(x)|$$

↑ Kolmogorov Complexity of string x

- $\langle M \rangle$: we always take it as constant. Only w is variable.
- M takes some input w & generates output x.
- We can say given $\langle M, w \rangle$, we can find x.

Q Can we compress every string into shorter length ?

$$M(w) = x$$

$$d(x) = \langle M, w \rangle \qquad K(x) = |d(x)|$$

Ex. $x = 011\ 011\ 011 \ldots$ i times

$$d(x) = |011| + |\langle M \rangle| + |\log_2 i|$$

↳ have to store value of i also

→ $$\boxed{K(x) \leq |x| + C} \quad \text{—— (1)}$$

→ to formalize wrt TM

C ↓

T.M. will have some of its own encoding (trans^n func^n in tape). ⇒ Cost of this encoding = C

If x It will put x in the tape & halt

If ($x = \epsilon$ → it will just halt) → (M, ϵ)

Ex. $$\boxed{K(xx) \leq K(x) + C}$$ find rel^n b/w these two. — (2)

minimum length string : w

$$M(w) = x$$

$$d(x) = \langle M, w \rangle$$

Let $x = 0110110111 \ldots$

→ $d(x) = (M, 011 \text{ i times}) \Rightarrow K(x) = |d(x)| = 3 + \log_2 i + |\langle M \rangle|$

→ ~~$d(xx) = (M, 011 \text{ 2i times}) \rightarrow K(xx) = |d(xx)| = 3 + \log_2 2i + |\langle M \rangle|$~~

will have to use new machine $N(w)$

$N(w)$:

1. $M(w)$ → put x in tape
2. & copy x again to get xx in tape.

$$|K(xx)| = |K(x)| + C$$

$|K(x)|$: 1st step; C : some transitions in TM N.

Ex. $K(xy) \leq 2K(x) + 2K(y) + C$ ——— ③

since we've added 2 bits in x part

we aren't able to identify the posⁿ when x ends & y starts (this was not needed in case of xx as we only used x there)

$x = 01101$

$x \uparrow y$

How to identify ?

We'll double each bit + nt in x

00 11 11 00 11

Then, we'll break this pairing by concatenating '01' & then put y

00 1111 00 11 01 y

(x = 00 1111 00 11; 01 ↑ 1st time pair have been broken)

↳ why we want to identify x & y ?

Because we only have $d(x)\,d(y)$ ($= d(xy)$) initially on LHS.

we've to get RHs

$d(x)$: access to TM which generates x ; $d(y)$: $\simeq y$ ⇒ Have to find out point where x & y meet. Else, we can't use the 2 TMs

ALSO, we can't do in $K(x) + K(y) + C$ length

because we need to identify the point $K(x,y)$

DATE : / /
PAGE NO. :

↳ How to improve ③ ?

also in binary

$|\overbrace{x}^{|x|}|\underbrace{x}_{x}|y$

↳ Put "length of x". Thus, we can divide x & y

$|x|\ x|y$

But again, we've to divide $|x|$ & x

$|x|\ |x|\ y$

↳ $2\log_2|x|$ ↳ for both parts

Total length = $2\log_2|K(x)| + K(y) + K(x)$

$K(xy) \leq 2\log_2|K(x)| + K(y) + K(x)$

15/02/19

Ex. $L = \{ (x,k) \mid K(x) = k \}$ → undecidable

(x: binary string, k: integer)

↳ (given a (x,k) tuple, we don't have programs that checks $K(x)=k$ & halts)

Proof by Contradn :

Assume that L is decidable.

∃ a program P which decides L.

(from P, we'll try to construct a program which will give the first string (y) which has $K(y) = k$)

[P(y) is program which tells $K(y)=k$ or not]

q : ① for all binary strings y in the lexicographic order, we check P(y,k)

② As we get first y s.t. $K(y) = k$,

③ O/p y.

constructed program with help of P.

* q : prints y s.t. halt
K(y) : min (<M,w>) : prints min. y

$(k =)\ K(y) \leq |q| \ (= |P| + \log_2 k + C)$ [assuming $K(y) < |q|$] ⇓ $K(y) < |q|$

for ① for ② ① ⇓

We have to choose k s.t. this inequality is contradicted.

We can find a k s.t.

$$k > \frac{c' \log_2 k + c'|p| + cc'}{\text{const}}$$ ($c' > 1$)

[atleast 1 k will exist that satisfies this inequality]

$\hookrightarrow$ ② { $k_0 > \log_2 k$ for $k \geq 2$ }

Putting ② in ①,

$$c' \log_2 k + c'|p| + cc' < k < |p| + \log_2 k + c$$

if $c' > 1$ ⇒ above inequality is contradicted.

⇒ By contradiction ∴ Hence Proved

↳ We know :

$$K(x) \leq |x| + c$$

We'll now prove :

$$K(x) \leq |x| - c$$ → some strings follow this ↓ "c compressible" binary strings

⇒ $M(y) = x$ and halt (Others strings : Incompressible by c)

$$|\langle M, y \rangle| \leq |x| - c$$

• $c+1$ compressible ⇒ c compressible but not vice versa

Existence Problem

Q Prove : $\exists$ 2^r binary strings which are Incompressible by c. for each length of binary string

Let us consider binary strings of length n.

If we can reduce $x \to y$ ($K(x) = y$) s.t. $|y| = n - c$ ⇒ string is c compressible

or $|y| = n - (c+i)$ ($c+i$) "

$(0 \leq n-(c+i) \leq n-c)$ [$0 \leq i \leq n-c$]

Proofs
- ↳ Induction ✓
- ↳ Contradiction ✓
- ↳ well Ordering → Extremal ~~problem~~ principal (try to bound → A state/both sides value & then contradict)
- ↳ Deduc" ✓
- ↳ Construc"

DATE : / /
PAGE NO.

Quest" is : how many such string 'y' we can come up with

we've to count all for $0 \le i \le n-c$

$2^0 + 2^1 + \dots + 2^{n-c}$

$= 2^{n-c+1} - 1$ ⟶ These many strings 'y' we can have

For each such y, we assume that we get a unique x.
(won't make a diff. in final result even if repeated)

$y \longrightarrow x$

Theorem

Atmost, no. of binary strings x with length n which are c compressible $= 2^{n-c+1} - 1$

Incompressible (atleast) $= 2^n - (2^{n-c+1} - 1)$

16-02-19

→ $f : \{0,1\}^* \longrightarrow \{\text{True, false}\}$ → Property of binary string (if property satisfied : True else : false)

Ex. even binary strings

if even → gives True

odd → " false

Property

Theorem → A property f holds for almost all binary strings

binary strings f True $= |\{x \in \{0,1\}^* \mid f(x) = \text{True}\}|$ — ①

binary strings f false $= |\{x \in \{0,1\}^* \mid f(x) = \text{false}\}|$ — ②

① + ② ≡ set of all binary strings ≡ 2^n (length of string = n)

Acc. to the statement,

$$\lim_{n \to \infty} \frac{\#\text{ binary strings of False}}{2^n} \longrightarrow 0 \qquad \text{— (2)}$$

Property statement: $\forall b > 0$.

For all sufficiently long binary string x s.t.

$f(x) =$ False, then

$K(x) < |x| - b$ (x is compressible by b)

Proof: M (i : binary integer)

{

generate all binary strings s in lexicographic order.

check if $f(s) =$ false

print s

}

→ will get i^{th} binary string for which $f(x) =$ false & print x.

Ex. s: { 0, 00, 01, 11, ... }

if $i = 3$

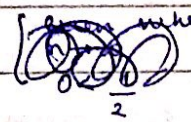
print: 01 ↳ set for which $f(x) =$ false

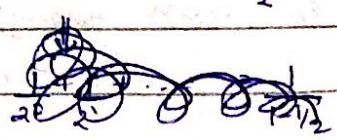
From this, we'll try to find some description of s.

We also know that,

$$\lim_{n \to \infty} \frac{\#\text{ binary strings of False upto length } n}{\text{Total no. of binary strings upto length } n} = 0 \quad \text{(from (2))} \qquad \text{— (4)}$$

$$< \frac{1}{2^n}$$





trying to approximate value of fraction

(have app. like this because we want to convert it into binary string)

Description of s (d(s)) :

$$|d(s)| \overset{=}{\not\leq} |KM>| + |<i>|$$

$|KM>|$ ↓ length of encoding of M

$|<i>|$ ↓ length of encoding of i

$d(s) = <M, i>$

[not talking about $K(x)$ in $K(x)$: min^m descrip^n. Here : just a descrip^n.

DATE : / /
PAGE NO. :

sufficiently
- we have long binary strings

If I can bound ④, I can bound i

recall: $\{x = 10.13^* \mid f(x) = \text{false}\}$

If put in lexicographic order

$\{0, 00, 001, 0001, \ldots\}$
$\quad 1 \quad 2 \quad 3 \quad 4$

↓ we want the upper limit for i

$$\frac{\#\text{ binary strings } \ni \text{ False upto len } n}{\#\text{ total binary strings upto len } n} < \frac{1}{2^x} \quad \text{— ⑤}$$

$\{\underset{\phi}{}\ \underset{1}{0},\ \underset{2}{1},\ \underset{3}{00},\ \underset{4}{01},\ \underset{5}{10},\ \underset{6}{11}, \ldots\}$

↑
$i \Rightarrow$ if want to get this pos^n, we've calculated all strings upto length $|10| = 2$

$\therefore i < 6$

if i is pos^n of binary string ~~upto~~ of length n, then, have to calculate all strings upto length n.

~~(2 + 2 + ... + 2)~~ $(2^1 + 2^2 + \ldots + 2^n)$

$$i < \frac{2^{n+1} - 1}{2^x}$$

$$\Rightarrow |d(s)| \leq |KM>| + \left\{ \frac{2^{n+1} - 1}{2^x} \right\}$$

$$K(s) \leq d(s)$$

$$\Rightarrow K(s) \leq |KM>| + \frac{2^{n+1} - 1}{2^x} \quad \text{— ⑥}$$

we want to fix x ~~st~~ in ⑤ st ⑥ reduced to

$$K(s) \leq |s| - b$$

$= \frac{2^{n+1}-1}{2^x} < 2^{n+1-x}$ $\Rightarrow 2^{n+1-x}$

this is the integer value we want to convert into binary

$K(s) \leq |\langle M \rangle| + (n+1-x)$

$\leq |s| - b$ → want to reduce into this

Fix x s.t

$|\langle M \rangle| + (n+1-x) \leq |s| - b$

$x > |\langle M \rangle| + 1 + b + n - |s|$ ↳ x will change as n changes

We have for sufficiently large binary strings

↓

for this, we've to choose $n > n_0$ s.t above value of x holds.

We can choose x in such a way (for $n > n_0$).

18-02-19

Revise last class:

(2) $\forall b > 0 \quad \forall x \in \{0,1\}^* \wedge |x| > n_0 \wedge f(x) = \text{false}$. Then x is b-compressible.

(1) f is a property of binary strings which holds for almost all binary strings

Proof: $f(x \in \{0,1\}^*) \rightarrow \{T, F\}$

$\lim_{n \to \infty} \frac{\text{No. of binary strings upto len } n \text{ when } f \text{ is false}}{\text{" " " " } n} \rightarrow 0$

$= 2^{n+1}-1$

suppose $n = 1$, $f(0) = \text{false}$

Expression $= \frac{|\{0\}|}{|\{\epsilon, 0, 1\}|} = \frac{1}{3}$

↳ this fraction goes down as n ↑es.

DATE : / /
PAGE NO. :

At some value of n, we can bound this fraction (upper bound)

For $n \geq n_0$, expression $< \frac{1}{2^x}$ (let us assume)

No of b.s upto len n when f = false $\leq \frac{1}{2^x}(2^{n+1}-1)$

② Proof :

Design M :

M (i : integer)

{

~~find ith binary string when~~

Generate binary strings in lexicographic order, find ith binary string s for which f is false and print s.

}

This TM puts s in tape

$\therefore \quad d(s) = \langle M, i \rangle$

$\therefore \quad \left| \{ x \in \{0,1\}^* \mid |x| = n \wedge f(x) = \text{false} \} \right| \leq \frac{1}{2^x}(2^{n+1}-1)$

↳ i here is decimal integer. have to convert it to to binary

$|d(s)| = |\langle M \rangle| + i$

$\therefore \quad \frac{1}{2^x}(2^{n+1}-1) < 2^{n+1-x}$

↳ binary no : $n+1-x$

$|d(s)| \leq |\langle M \rangle| + n + 1 - x$

$K(s) \leq |\langle M \rangle| + n + 1 - x \quad \text{—— (1)}$

We've to prove that s is b compressible, ie

$|K(s)| \leq |s| - b \quad \text{—— (2)}$

$$|\langle M \rangle| + n + 1 - x = |s| - b$$

$$x = |\langle M \rangle| + b + 1 \quad \rightarrow \quad \forall\, n \geq n_0$$

for $|x| < n_0$ ⇒ set is now finite

⇓

In this set, we've incompressible as well as compressible strings

(if $f(x)$ is considered to be false)

↓

for $|x| \geq n_0$: almost all strings compressible

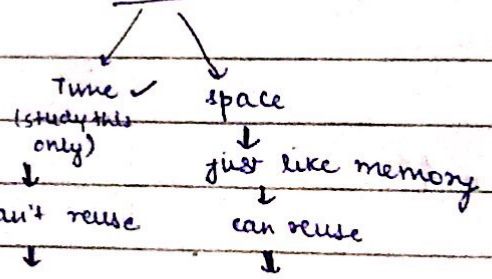
Decidable Problems

(study set of decidable langs)

↳ Every TM will halt.

↳ will either move to reject state or accept state

→ This is a big set. We've to split it into smaller sets based on some resources



Time (How much time a TM take to accept the language)

↓

Time(n) = { ~ take O(n) time ~ . . }

O(n) time

Time (n^2) = { ~ take $O(n^2)$ time ~ . . . }

Time (n^k) = { . . . . $O(n^k)$ time ~ ~ }

~~$x^2 = 0$~~

2 more classes :

P = Union of all the langs which are accepted by deterministic TM in polynomial time ($T(n), T(n^2), \dots T(n^k)$)

NP = " " " " " nondeterministic ($T(n), T(n^2), \dots T(n^k)$)

$$NP = \bigcup_{k>0} NTIME(n^k)$$

$$P = \bigcup_{k>0} TIME(n^k)$$

→ Not yet proven

→ ~~Proved~~ $P \stackrel{?}{=} NP$ (conjecture)

We know that we can convert ND → D.

Then, why this ques^n. → because ND → D can't be done in polynomial time.

↳ Diagonalization won't work to find whether P = NP or P ≠ NP.

```
gcd (x, y) {
    (x, x % y)
    exchange (x, y)
    until x = 0
}
```

How to prove correctness?

Loop invariant

↓

find something in your loop which never changes. Loop must also halt.

↳ Here, gcd will never change throughout the loop.

↳ Counting sort : pseudopolynomial $O(nk)$ $k = O(n^k)$

↳ Prime (n)

```
i = 1
n/i
```

n → O(n) : also pseudo-polynomial

if $n = \log_2 n$ → $O(2^{\log_2 n})$