Unit 6

DATE
PAGE

Notes by Soham

Turing Machine :- act as a Transducers

↳ Systems of rules, states & transitions rather than real m/c.

Purpose :-

↳ Deciding Formal lang.

↳ Solving mathematical functions.

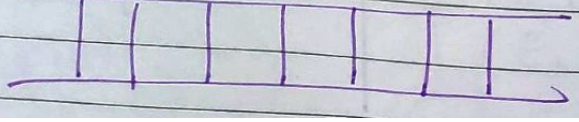
↳ Used to accept recursively enumerable lang. (generated by Type-0 grammar)

Recursively ↓ repeating same set of rules any no. of times.

Enumerable ↓ list of elements.

Model of Turing Machine :-

1) **I/P Tape :-**



FA < PDA < LBA < TM

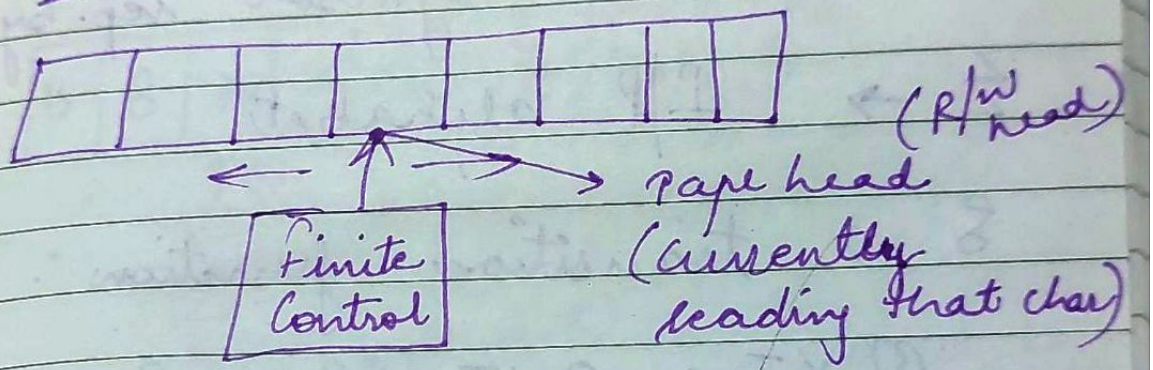
DATE
PAGE

empty char → Δ

Eg:- Let string is abai

a	b	a	Δ	Δ	Δ	Δ

2.) Finite Control



Tape head can be moved to left or right of I/P tape. It will point out the current char which is to be read from I/P tape.

DATE
PAGE

Representation of TM :-

$M = (Q, T, B, \Sigma, \delta, q_0, F)$

$Q \rightarrow$ Finite set of states

$T \rightarrow$ Tape alphabet (symbols on tape)

$B \rightarrow$ Blank symbol rep. by Δ.

$\Sigma \rightarrow$ I/P alphabet

$\delta \rightarrow$ transition function.

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

↑ State — ↑ Tape alphabet — $\{L, R\}$ moving tape ptr to left (L) or right (R).

$q_0 \rightarrow$ initial state

$F \rightarrow$ set of finite states.

In case of non-deterministic :-

$$Q \times \Gamma = 2^{(Q \times \Gamma \times (L \times R))}.$$

DATE
PAGE

Design turing m/c for $\{a^n b^n / n \geq 1\}$

⇒ Let : "aabb"

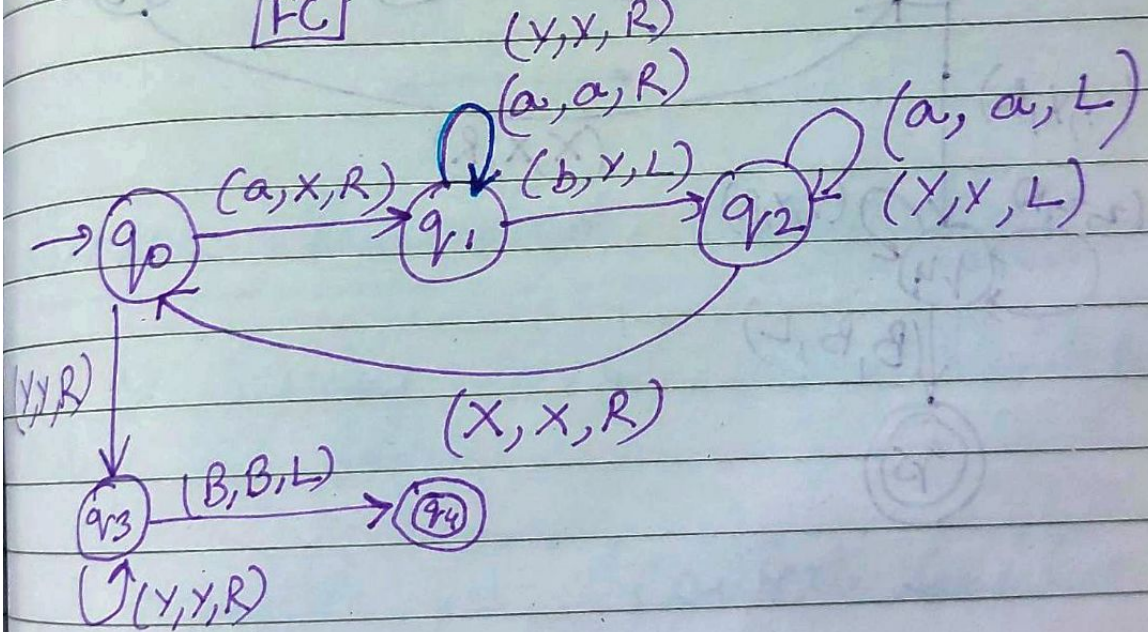
B | ~~a~~ | ~~a~~ | ~~B~~ | ~~B~~ | B

Infinite tape is given.

B	B	~~X~~	~~a~~	~~X~~	~~Y~~	~~Y~~	~~Y~~	B	B

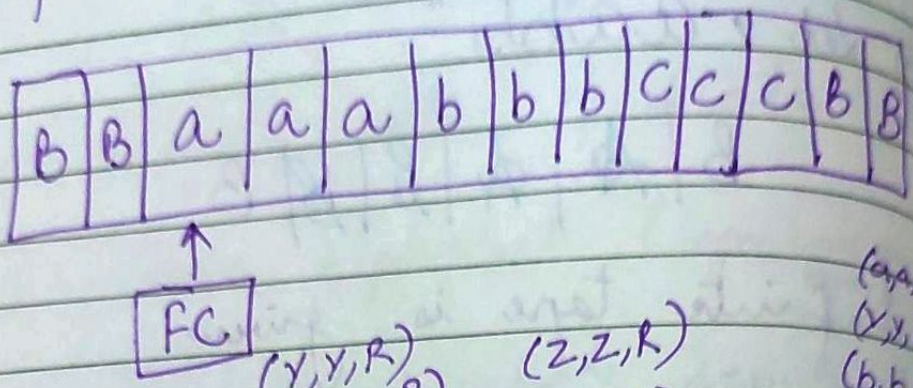
R/W

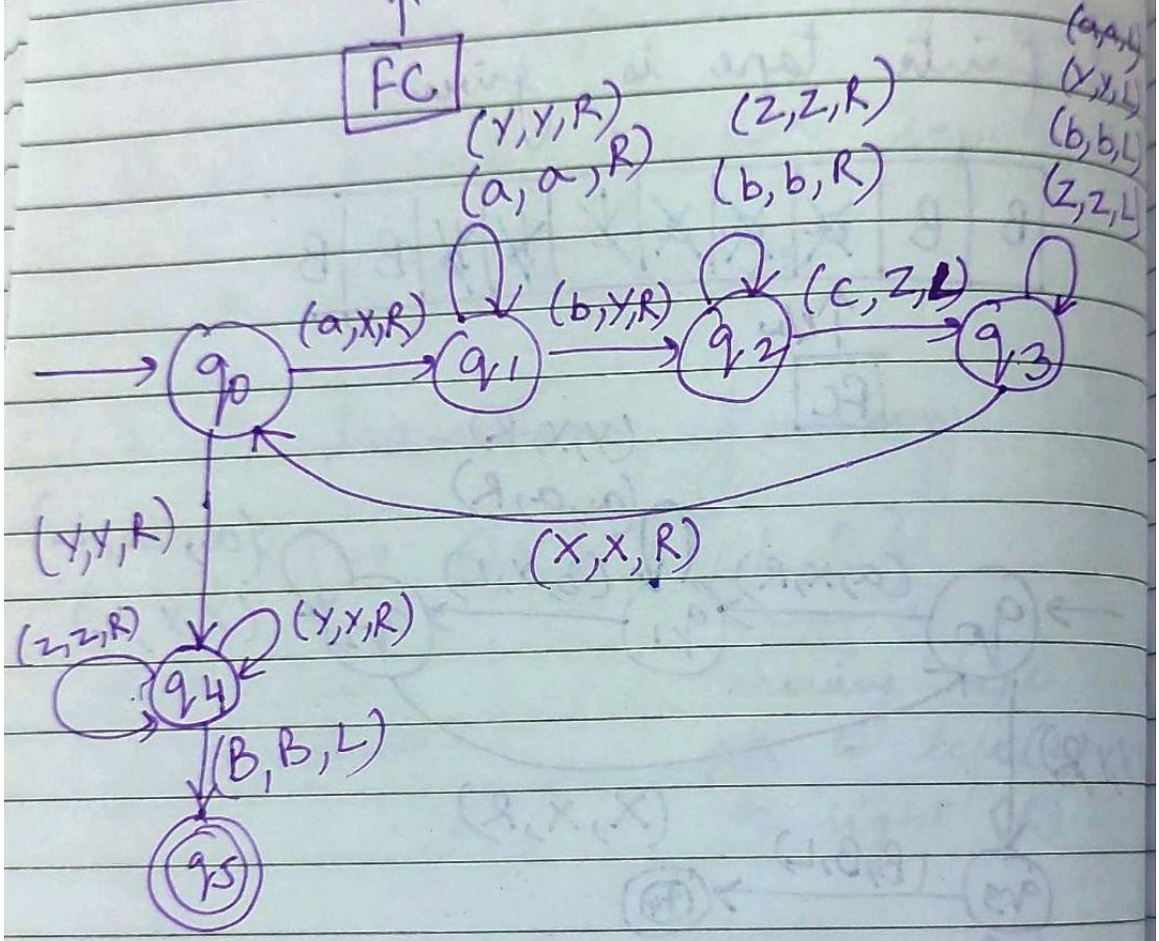
FC

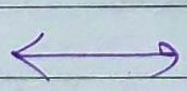


DATE
PAGE

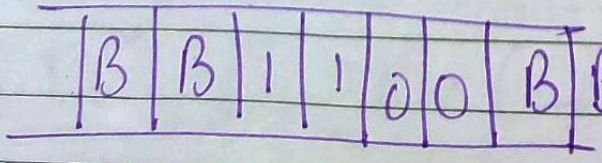
Q. Design Turing machine for $\{a^n b^n c^n \mid n \geq 1\}$





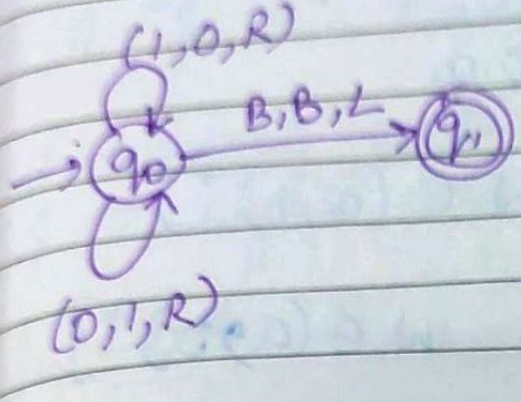


Q. Turing Machine for 1's complement.



Eg:- Ø 1 Ø 1 B
1 0 1 0 ↑

(Also called Transducer)
i.e converting I/P to O/P.



Linear Bounded Automation

↳ Turing m/c with limited size I/P tape.

↳ used in context sensitive lang.

↳ LBA is linear bound automata. It is not categorised into DFA & NDFA.

LBA = TM + I/P size tape

DATE
PAGE

Examples:-

1. $L = \{a^n b^n c^n : n \geq 1\}$
2. $L = \{a^n : n \text{ is prime}\}$
3. $L = \{a^n : n \text{ is non-prime}\}$
4. $L = \{a^{n!} : n \geq a\}$
5. $L = \{ww : w \in (a,b)^+\}$
6. $L = \{www^R : w \in (a,b)^+\}$
7. $L = \{w^n : w \in \{a,b\}^+ \; n \geq 1\}$
8. $L = \{a^n : n = m^2, m \geq 1\}$

Model :-

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

$Q \rightarrow$ ~~FFT~~ states

$\Sigma \rightarrow$ I/P

$\Gamma \rightarrow$ Tape alphabet

$\delta \rightarrow$ transition

q_0 → initial state

F → set of final states

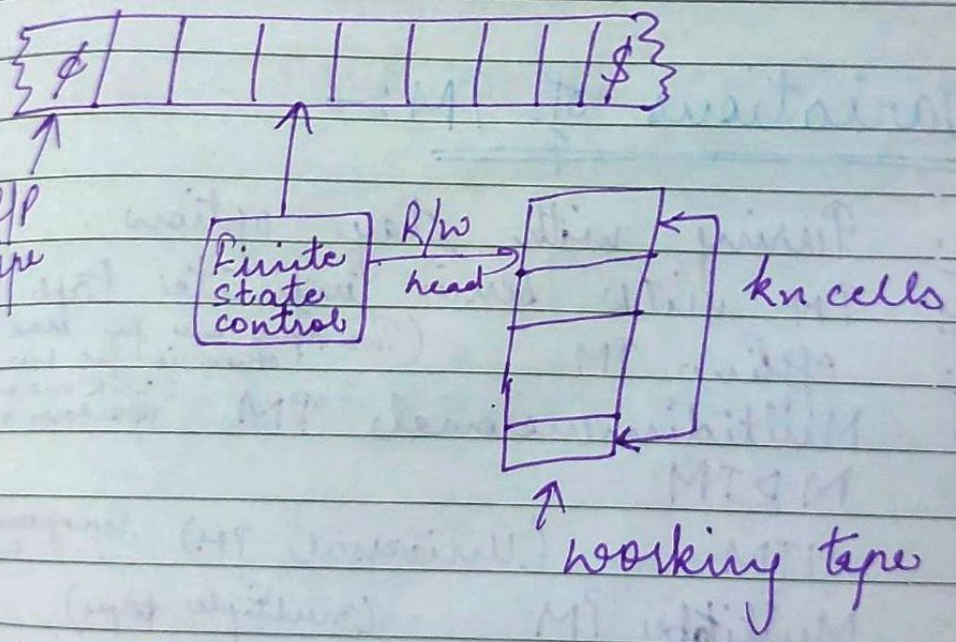
b → blank

¢, $ → markers.

¢ → Left end marker.

$ → Right end "

In LBA, there are two tapes.



↳ On I/P tape, the head never prints & never moves to the left.

↳ On working tape, the head can modify the contents in any way without restriction.

ID = (q, w, k)

where

$q \in Q$

$w \in \Gamma$

k → some integer b/w 1 & n.

value of k changes to :-

⇒ k-1 if R/w head moves to the left.

⇒ k+1 if head moves to the right.

Variations of TM :-

1. Turing with stay option
2. TM with semi infinite tape
3. Offline TM (Two tapes one for head & other is for R/W)
4. Multidimensional TM (L & R move with up & down ")
5. NDTM
6. UTM (Universal TM) reprogrammable 3-tapes
7. Multitape TM (multiple tapes)
8. Multihead TM
9. TM with 3 states
10. Jumping TM
11. Non-erasing TM
12. Always writing TM.

A) TM without writing capacity
(will become FA)

B. TM with I/P size Tape
(will become LBA)

C. TM with tape used as a stack * (NDTM)
(will be PDA)

D. TM with finite ~~food~~ tape.
(will become FA)

$FA < PDA < LBA < TM$

$\longleftrightarrow$

NDTM :-

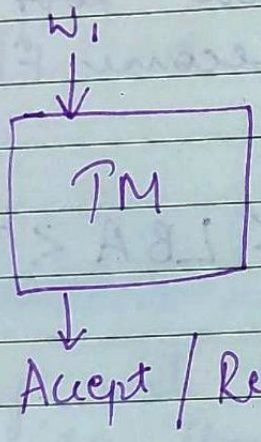
$\delta = Q \times \Sigma \to P\{\Gamma \times (R/L) \times Q\}$

Recursively Enumerable Lang:- (may or may not halt)

⇒ L is R.E if there is a TM.

⇒ It has 3 states:-
- ↳ Halt & Accept
- ↳ Halt & Reject
- ↳ Never Halt

⇒ Closed under all except complem



Recursive Lang:- (TM will always Halt)

⇒ L is recursive if there is Halting / Total T.M.

⇒ Two states:- Halt & Accept
Halt & Reject

⇒ Closed under all except Homomorphism & substitution.

Decidability & Undecidability :-

Decidable :- (RL) L is decidable if it is recursive lang. All decidable lang. are recursive lang. & vica-versa.

Partially Decidable lang :- (REL)

A lang. 'L' is partially decidable if 'L' is a recursively enumerable language.

Undecidable lang :- (No TM)

↳ when it is not decidable.

↳ It can be partially decidable

↳ If a lang. is not partially decidable, then there exists no TM for that lang.

Time Complexity of Turing m/c :-

Refers to measure of the number of times the tape moves when the m/c is initialized for some I/P symbols.

$$T(n) = O(n \log n)$$

DATE
PAGE

Space complexity of Turing m/c

is the number of cells of the tape written.

$$S(n) = O(n)$$

Power of Linear Bounded Automata

Length = function (length of initial I/P string, constant c)

Here,

m/m info $\leq c \times$ I/P info.

Cellular Automation :-

↳ is a collection of cells arranged in a grid of specified shape, such that each cell change state as a function of time, according to a defin set of rules driven by the states of neighboring cells.

DATE
PAGE

Char. of a CA :-

↳ Cells in a CA reside on a grid which has specified shape & exist in a finite number of dimensions.

↳ Each cell on the grid has a state. While there are numerous finite possibilities of the state, the simplest state form is usually ON or OFF.

↳ The cells adjacent to one cell constitute its neighborhood. Cells in neighborhood affect each other & each cell on CA grid has a neighborhood.

(Undecidability) ↗

The Halting Problem :-

⇒ means if a program is given, will it halt? (means it will either accept or reject).

⇒ Given a TM, will it halt when run on some particular given I/P string ?

↳ There will be no generalized algo for this.

⇒ Given some program written in some lang. (Java / C etc) will it even get into an infinite loop or will it always terminate ?

⇒ This is an undecidable problem.

Q. Why ?

⇒ In general, we can't always know.

⇒ The best we can do is run the program & see whether it halts.

⇒ For many programs, we can

DATE
PAGE

see that it will always halt or sometimes loop.

⇒ But for programs in general, the question is undecidable. So no TM can be designed on this problem.

⇒ Can we design a m/c which if given a program can find out or decide if that program will always halt or not halt on a particular I/P ?

Let us assume that we can:

H(P, I) → H → m/c or program
P → program
I → I/P.
- Halt
- Not Halt

This allows us to write another program:

X → Program
C → m/c

```
C(X)
if { H(X, X) == Halt }
    loop forever;
else
    return;
```

If we run 'C' on itself :-

C(C)

- H(C,C) == Halt → Not Halt (contradiction)
- H(C,C) = Not Halt → Halt (contradiction)

There is contradiction on both sides so H is wrong. So our assumption of m/c 'H' is wrong. So 'H' will not exist.

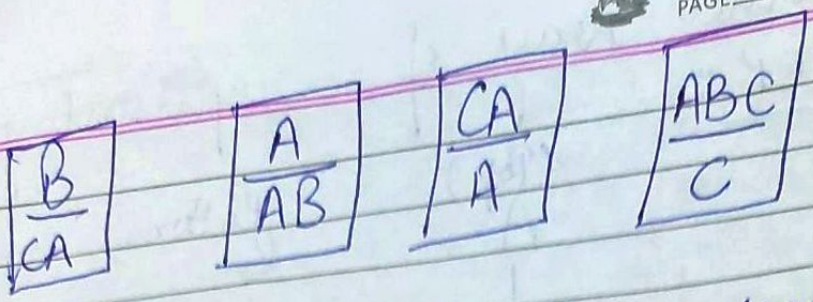
⟷

The Post Correspondance Problem :- (PCP)

↳ Introduced by Emil Post.

↳ Here we have 'Dominos' - (Denominator & Numerator)

Eg:- Let us take 4 dominos as :-



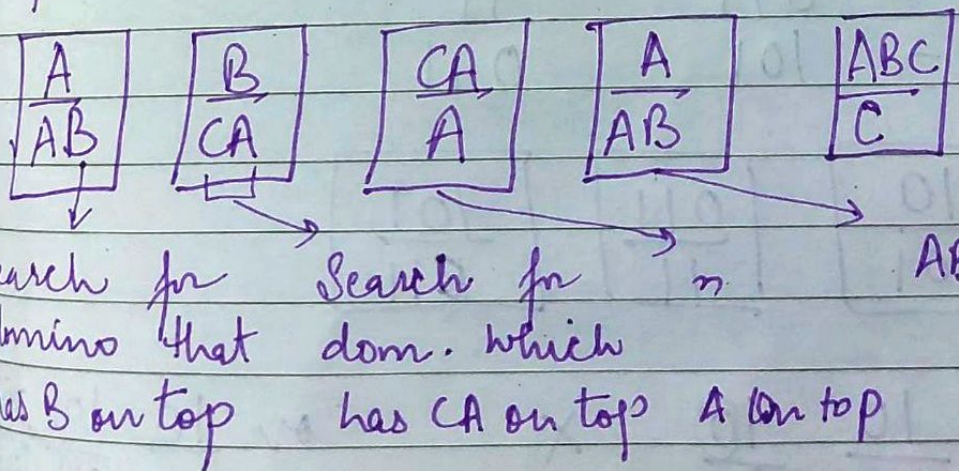
→ We need to find a sequence of dominos such that the top & bottom strings are the same.

→ So we need to arrange the symbols such that top & bottom have same symbols.

→ Any dominos can be used any no. of time.

Sol

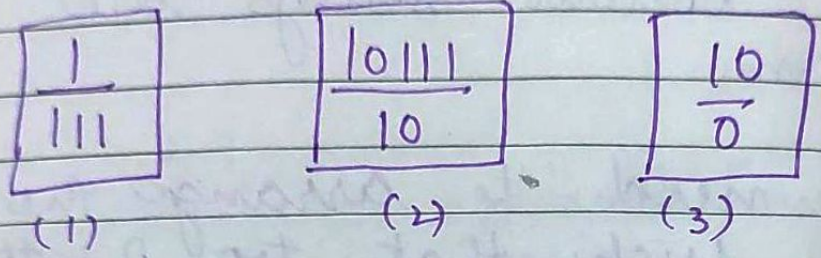
Always start with dominos where 1st symbol of den. & num. are same.



⇒ Top = A B C A A A B C
Bottom = A B C A A A B C
So this is same.

Another way of representing PCP :-

(Domino no.)	(Top) A	(Bottom) B
1	1	111
2	10111	10
3	10	0



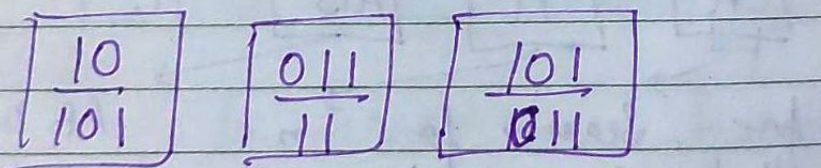
(1) (2) (3)

	(2)	(1)	(1)	(3)
A :	10111	1	1	10
B :	10	111	111	0

So A = B

Eg :-

	A	B
1 -	10	101
2 -	011	11
3 -	101	011



⇒ $\frac{10}{101}\ \frac{10}{101}$ ✗ or $\frac{10}{101}\ \frac{101}{011}\ \frac{10}{101}$ ✗

or $\frac{10}{101}\ \frac{011}{11}$ ✗

DATE
PAGE

1010 1101 101 x
1010 1101 011

So this is undecidable.