Unit 3 TOC Sem 4 Yash Batch

Theory of Computation Unit-3

① Turing Machine (TM)

→ Introduced by Alan Turing in 1936

> used to formalize the concept of algorithm or mechanical procedure.

It is a 7-tuple → A tuple with 7- elements.

$TM = Q, \Sigma, \Gamma, \delta, q_0, q\text{-accept}, q\text{-reject}$

Q → It is a finite set of sets

Σ → It is the input alphabet (not include ␣ (blank)).

Γ → It is the tape alphabet (include blank symbol ␣)

$(\Sigma \subseteq \Gamma)$

δ → Transition function $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

$q_0 \in Q$ is the start state

$q\text{-accept} \in Q$ is the accept state

$q\text{-reject} \in Q$ is the reject state $(\neq q\text{-accept})$

> Note :- In Turing Machine, we can read/write in I/P - O/P tape and in both left/right direction.

Computability Theory → It is a branch of mathematical logic that studies which problems are computable i.e. problems solved by algorithms.

Also known as Recursion Function Theory

Extension of Turing Machines

→ By increasing extensions they do not increase the computational power

i) Multi-tape Turing Machine

→ Has multiple tapes, each with its own head.

→ Transition func^n uses read-writes multiple symbol at once.

ii) Non-deterministic Turing Machine

→ Can have multiple possible moves for a single configration.

→ Accepts input if any path leads to an accept state.

iii) Multi track turing machine

→ A tape divided into tracks.

→ Each cell contains a tuple of symbols.

iv) Turing machine with stay option

→ Head can stay in place (adds to direcⁿ choices)

→ Slight extension but doesn't change power.

v) Oracle Turing Machine

→ Can ask an oracle a question about a language instantly.

vi) Universal Turing Machine

→ It can simulate any other Turing Machine on any input.

Equal in power

# Deterministic Turing Machine	Non-Deterministic Turing Machine
i) for an input, we can move either in left or in right direction.	for an input, we can move in both direction.
ii) Single path	Many path (parallel exploration)
iii) Used in Algorithm design actual machines	Used in Complexity theory.
iv) e.g→ write a specific symbol and move in one direcⁿ on choice	e.g→ in state q_1 reading 1 machine move to q_2 write 0 move right OR

iv) Turing machine with stay option

→ Head can stay in place (adds to direⁿ choices)

→ Slight extension but doesn't change power.

v) Oracle Turing Machine

→ Can ask an oracle a question about a language instantly.

vi) Universal Turing Machine

→ It can simulate any other Turing Machine on any input.

Equal in power

# Deterministic Turing Machine	Non-Deterministic Turing Machine
i) for an input, we can move either in left or in right direction.	for an input, we can move in both direction.
ii) Single path	Many path (parallel exploration)
iii) Used in Algorithm design actual machines	Used in Complexity theory.
iv) e.g→ write a specific symbol and move in one direⁿ no choices.	e.g→ in state q_1 reading 1 machine move to q_2 write 0 move right OR q_3 write 1 move left

Linear bounded automata

It is a restricted Turing Machine where the tape is bounded by the length of the input.

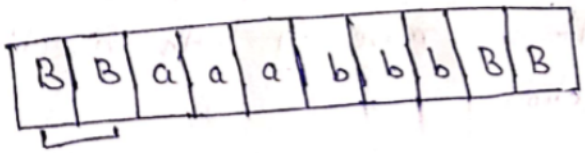
also we can say it is a non-deterministic turing machine that never moves its tape-head outside the tape that contains the input.

e.g. → $a^n b^n c^n$: $n \geq 1$ $\quad$ a^n : n is non-prime

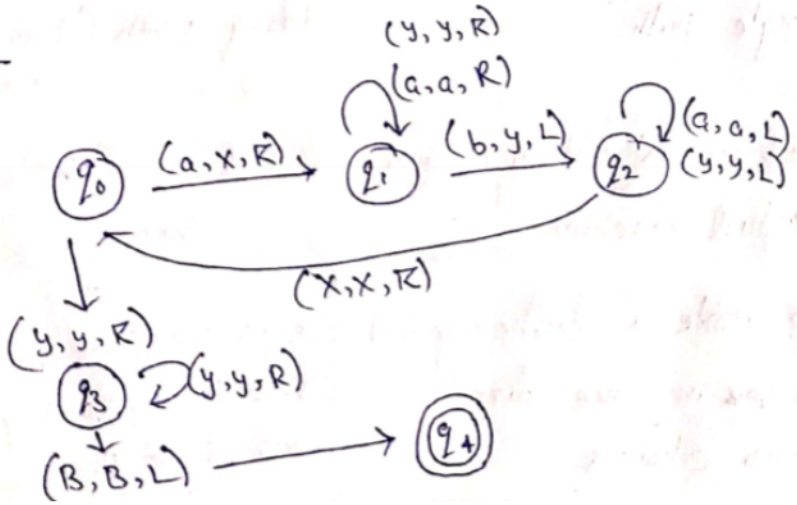
a^n : n is prime $\quad$ ww : $w \{(a,b)^*\}$

Design of a Turing Machine

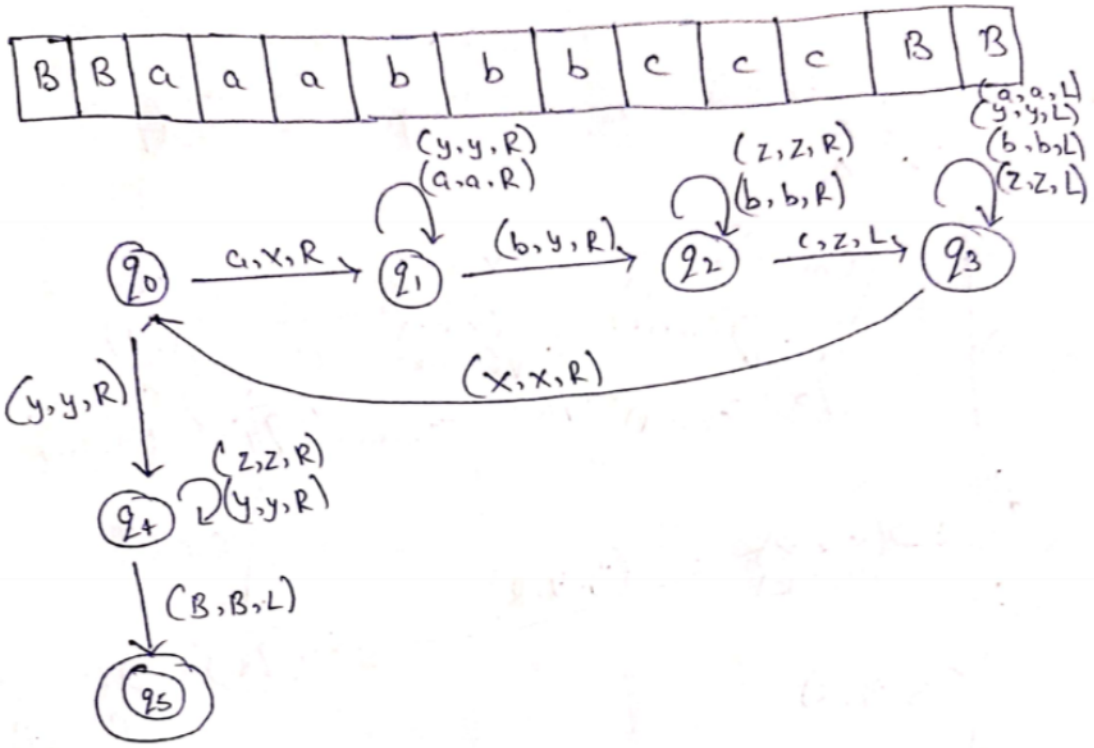
① $a^n b^n \mid n \geq 1$ (INP) $\quad$ [X X X Y Y Y]



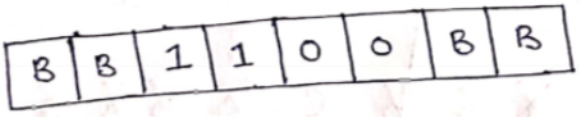
Blank Space



② $a^n b^n c^n \mid n \geq 1$

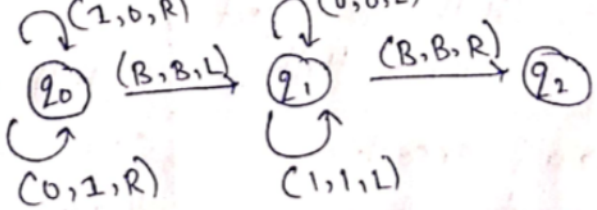


Turing Machine for 1's Complement

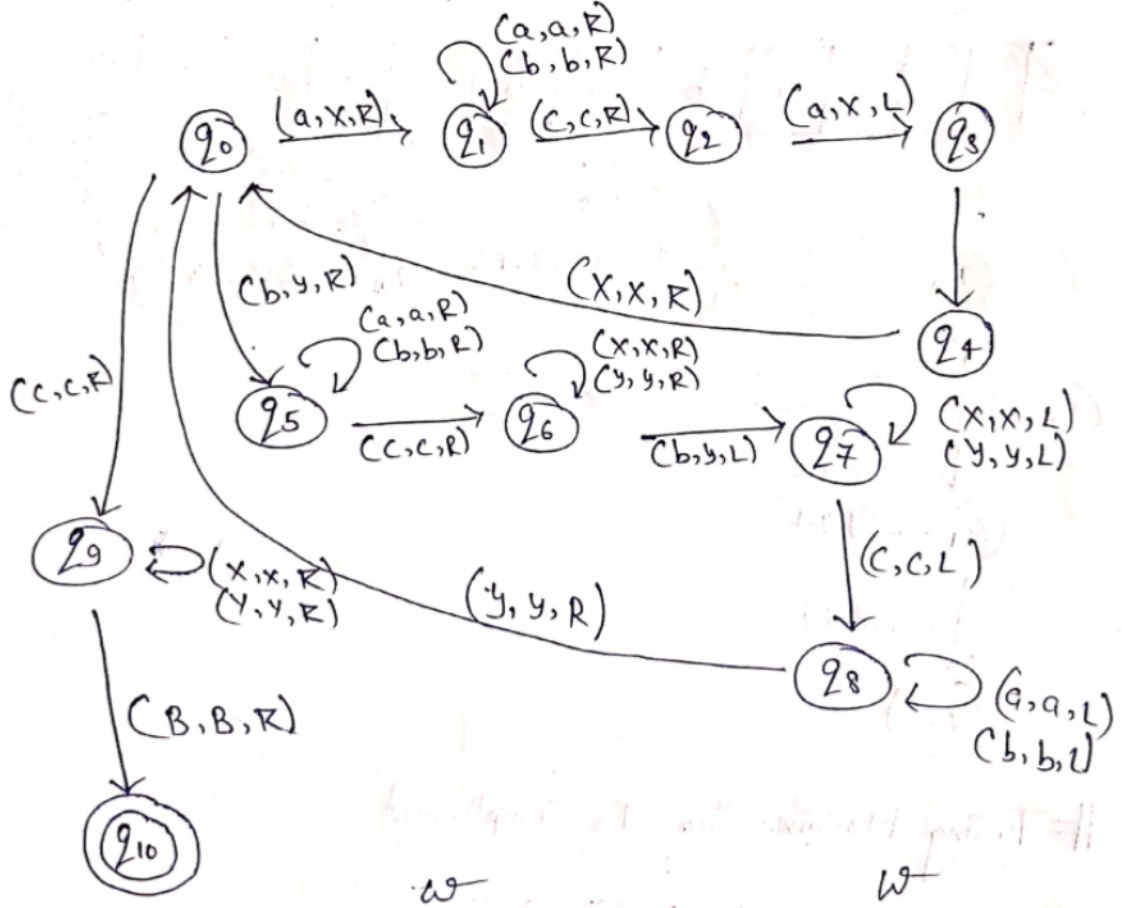


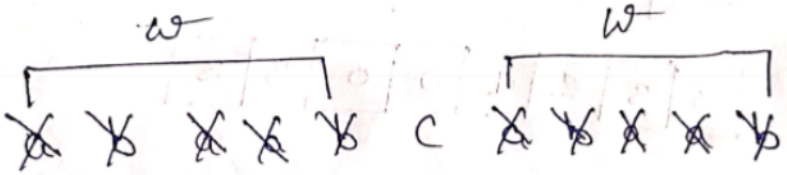
(α β γ) → newstate
γ → direction change
α → change

States	0	1	B
q_0	$1Rq_0$	$0Rq_0$	BLq_1
q_1	$0Lq_1$	$1Lq_1$	BRq_2
q_2	—	—	—



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





# Recursive Enumerable lang	Recursive lang
i) accepted by TM	accepted by Halting TM
ii) three states → Halt and accept Halt and reject Never Halt	Two states → Halt and accept Halt and reject
iii) Closed under all except Set diff, complement.	Closed under all except Homomorphism.

Church Turing Thesis

i) A func^n can be computed algorithmically (i.e. by a step-by-step process) can be computed by a turing machine.

→ No mathematical model is more powerful than Turing machine.

→ There is no problem which can be solved by digital comp but not by Turing machine.

Decidable language → a language is decidable if it is a recursive lang.

Partially decidable language → a language is partially decidable if it is a recursively enumerable lang.

Undecidable language → It may be sometimes be partially decidable but not decidable.
If a lang is not even partially decidable, then there exists no TM for that lang.

- Semi-decidable
  - ↳ always halt if ans is 'yes'.
  - ↳ may or may not halt if ans is 'No'.

# Decidability Problems	Undecidability Problems
→ A problem is decidable if we can construct a Turing machine which will halt in finite amount of time for every i/p and give answer as 'yes' or 'no'.	→ If there is no Turing machine which will always halt in finite amount of time to give answer as 'yes' or 'no'.
→ It has an algorithm to determine the answer for a given I/p.	→ It has no algo.
e.g → equivalence of RL.	e.g → equal or not (CFL). Ambiguous of CFL.

Halting Problem (IMP)

Alan Turing proved that the Halting Problem is undecidable.

↳ There is no general algorithm that can decide whether the program halts or not.

OR

No Turing Machine can solve the Halting Problem.

Why is it Undecidable?

Turing used proof by Contradiction

i) Assume we have a machine H that solves the Halting Problem: H(M, w) returns:
- YES if machine M halts on input w
- NO if it runs forever

ii) Now, create a new machine ∀ that uses H like this:
- On input M, do:
  Run H(M, M)
  if H says M halts, loop forever
  if H says M doesn't halt, then halt

iii) Now, ask: what happens when we run ∀(∀)?
- if H says ∀(∀) halts
  ↪ then ∀(∀) loops forever.
- if H says ∀(∀) loops forever
  ↪ then ∀(∀) halts.

This is a Contradiction
So, such a machine H cannot exist.)

Post Correspondence Problem (PCP)

The PCP helps in describing the undecidability of the string.

The PCP consists of 2 lists of strings that are of equal length over the input.

The 2 lists are

$A = w_1, w_2, w_3, \ldots, w_n$

$B = x_1, x_2, x_3, \ldots, x_n$

then, there exists a non-empty set of integers such that

$$w_1, w_2, w_3 \ldots = x_1, x_2, x_3 \ldots$$

To solve PCP, we will try all combinations of $i_1, i_2, i_3 \ldots$ to find $w_i = x_i$.

Then, we can say that PCP has a soln otherwise it doesnot have a soln.

e.g→ X_1 X_2 X_3

$M = ab, bab, bbaaa$

$N = a, ba, bab$
y_1 y_2 y_3

for $X_2 X_1 X_3 \Delta y_2 y_1 y_3$

have a PCP soln?

$X_2 X_1 X_3 = babbbbaaa$
$y_2 y_1 y_3 = baabab$
] not equal

There is no soln ∴ The PCP is undecidable.

Rice Theorem (IMP)

Trivial Property → Property that holds for every machine or holds for none is trivial.

Rice Theorem doesn't apply to such a properties

Non-Trivial Property → Property that is neither true nor false for every computable function.

[e.g→ lang of machine = Σ^*]

> any non-trivial property about the language recognized by a Turing Machine is undecidable.

Note → This property depends on what language the TM accepts, not on how the machine is built or how fast it runs.

Reducibility

↳ A reduction is a way of converting one problem to another problem, so that the soln to the second prob. can be used to solve the first problem.

e.g → finding area of rect reduces to measuring it's width and height

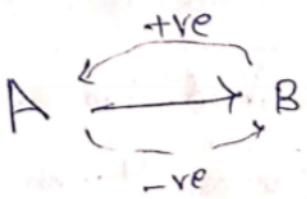
if A reduces to B, you can use a soln to B to solve A

$A \rightarrow B$

$A \leq B$

$A \leq_m B$

] A is reducible to B.



- if A is un-decidable then B is also undecidable.
- if B is decidable then A is also decidable.

Q Explain Recursion Theorem

This theorem says that any computable func^n can include its own code as input.

Let T be a T.M that computes a func^n f

Then, there exists a T.M (R) such that

$$\phi_R = \phi_f(R)$$

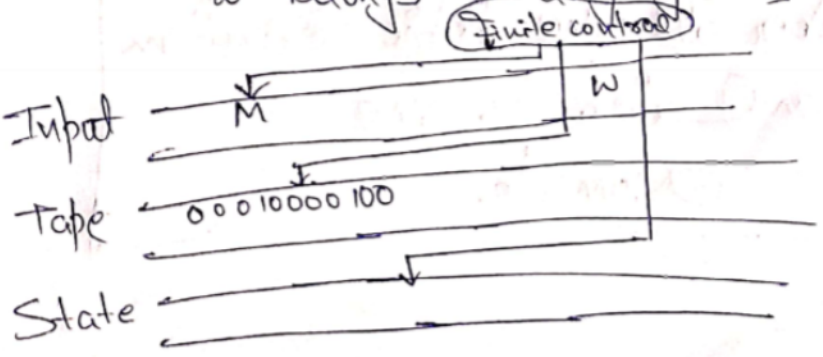
↳ func^n computed by R (ϕ_R)

Q Universal Turing Machine → (accept many T.M.)

The universal language L_0 is a set of binary strings which can be modeled by a turing machine.

UTM represented :- (M, w)

[M → T.M string in $(0+1)^*$ such that w belongs to language L]



(PYQ)

Q Diff^{n} b/w Pushdown Automata and Turing Machine

Q Short notes on:- Universal Turing Machine, Halting Problem, Post Corr^{n} Problem, [Recursion Theorem.], Church Hypothesis.

Q Diff^{n} b/w :- (DTM and NDTM)

Q Prove that Halting Problem is undecidable.

Q Define Recursively Enumerable language and its Properties

Q Variants of Turing Machine

Q Construct a Turing Machine M to accept the set L of all string over [0, 1] ending with 010.

Q Remember all the topic definition and also its one example.

Pushdown Automata and Turing Machine :

* PDA *

1) Read single alphabet at a time, perform operation on stack and header will move to right by one cell.

2) Header always ~~read~~ move from left to right and top to bottom.

3) Memory is available in the form of stack.

4) Can not change contents of i/p tape

* TM *

1) Read single alphabet at a time, replace that symbol by same / with some other symbol and header will move to right by one cell.

2) Header ~~always~~ can move from left to right & right to left and top to bottom.

3) Lack of memory

4) Can replace the contents of i/p tape by replacing corresponding alphabets.