1

F.A & A.T

TUTORIAL - 7

JUNAID. GIRKAR
60004190057
SE COMPS A-3

Q1 Short note on Halting problem.

ANS
- The halting problem, commonly applied to Turing-complete programs and models, is the problem of finding out whether, with the given input, a program will halt at some time or continue to run indefinately.
- The halting problem is often used in an abstract capacity to explain why it may be impossible to decide whether a program will ever run indefinately or not.
- Halting analysis for a program of any significant size requires large-dimensional numbers that would occupy massive memory spaces.
- Now, the halting problem occurs when a string is accepted by a Turing machine.

Halting problem is the language H defined by,

$$H = \left\{ M; a : M \text{ is a valid Turing machine description, and } M \text{ halts on input } a \right\}$$

- The Halting problem is undecidable over Turing machines

e.g: i/p = A Turing machine & an input string w.
Problem = check the Turing machine finish computing of the string w in a finite number of steps.

The block diagram of a Halting machine is

Input string → HALTING MACHINE → Yes (HM halts on i/p w)

→ No (HM doesn't halt on i/p w)

i) First we assume that such a turing machine exists to solve this problem and then we will show it is contradicting itself.

ii) We call Turing machine as a Halting machine that produces a 'yes' or 'no' in a finite amount of time. If the halting machine finishes in a finite amount of time, the output comes as 'yes', otherwise as 'no'.

Q2 Universal turing machine

ANS 1) A computer, like an Automation, is necessarily hard wired to execute a single algorithm. For a general-purpose stored program computer algorithm is basically "fetch an instruction from the current location ; execute the instruction ; go to a new location"

2) We can pull the same trick with a Turing machine. We will write a "program" directly on the tape, along with the "input" to that program. Then we will design a hard-wired Universal Turing machine to emulate the Turing machine described on the tape.

3) The program is stored as a sequence of 5 tuples:
[old state, symbol read, symbol to write, direction, new state]

A Turing machine M is designed to solve a particular problem P, can be specified as :-
1) The initial state q_0 of the TMM
2) The transition function δ of M can be specified as given

If the current state of M is q_i and the symbol under the head is a_i then the machine moves to the state q_j while changing a_i to a_j. The move of tape head may be :-
1. To - left
2. To - right
3. Neutral

$$\left\{ (q_i, a_i, q_j, a_j, m_f) : \begin{array}{l} q_i, q_j \in Q; \\ a_i, a_j \in T; \\ m_f \in \left\{ \begin{array}{l} \text{To - left} \\ \text{To - right} \\ \text{Neutral} \end{array} \right\} \end{array} \right\}$$

UTM should be able to simulate every turing machine. Simulation involves
1. Encoding behaviour of a TM as a program.
2. Execution of the above program by UTM.

Q3 Enumerable language

ANS A Turing enumerable language can be enumerated by some Turing Machine. To enumerate a language means to list the elements one at a time

A Turing machine enumerates a language :

1) A turing machine M can be made to list elements of a languge L
2) A multiple tape Turing machine (K-tapes), Tape 1 can be reserved exclusively for output.
3) let M be a K-Tape Turing machine with $K \geq 1$ and $L \subseteq \Sigma^*$. M

CONDITIONS :

i) The tape heads on first tape moves only in the forward direction replacing blank symbols with valid strings $w_i \in L$

ii) For every $w_i \in L$

$w_1 \# w_2 \# w_3 \cdots \# w_n \# w \#$

$w_1, w_2, \cdots w_n \longrightarrow$ distinct strings in L

a) if L (finite) — nothing printed

b) if L (finite) — M can either Halt / continue to loop forever

c) if L (infinite) — M continues to move forever.

M carries out the following sequence of operations.

1. w_{i+1} is computed from w_i where $w_{i+1} = w_i . \Sigma$ for each alphabet in Σ

2. If w_{i+1} takes the machine M to an accepting state than w_{i+1} is written to tape 1.
3. Step 1 and step 2 are carried out indefinitely if L is infinite.

Q4 Church hypothesis

ANS
1. The turing machine is general model of computation
2. Any algorithmic procedure can be solved by computer can also be solved by T.M
3. Problems computed by a computer or a Turing machine are also known as partial recursive functions.
4. Some enhancements to the TM made the church-Turing thesis acceptable. These enhancements are :-
   a) Multi-Tape
   b) Multi-head
   c) Infinite tapes
   d) Non-determinism.

Since the introduction of TM, no one has suggested an algorithm that can be solved by a computer but cannot be solved by a TM.

Q5 Construct TM for checking well formedness of parenthesis.

ANS Defination : $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

where,

Q = set of states

Σ : input alphabets.

Γ : Tape alphabet

δ : Transition function

q_0 : initial state

B : Blank

F : Finite state

LOGIC :

Replacing first ')' to X

Replacing the 'C' to X

TRANSITION TABLE

$Q \backslash \Gamma$	C	X	)	B
q_0	(q_0, C, R)	(q_0, X, R)	$(q_1,)/X, L)$	(q_f, B, N)
q_1	$(q_0, C/X, R)$	(q_1, X, L)		

Description :

$Q : \{q_0, q_1, q_f\}$

$\Sigma : \{C,)\}$

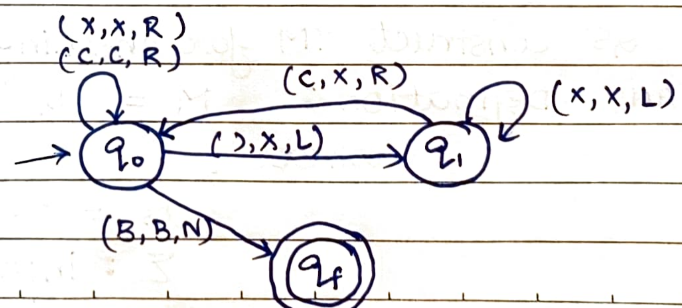
$\Gamma : \{X, B\}$

δ : Transition function $Q \times \{\Sigma \cup \Gamma\} \rightarrow Q \times \{\Sigma \cup \Gamma\}$

$q_0 : \{q_0\}$

B : B

F : $\{q_f\}$



Q6 Construct TM to recognise equal no of a's and b's.

ANS STEP 1: Defination $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

where
Q : set of states
Σ : Input alphabets
Γ : Tape alphabet
δ : Transition function $\delta : Q \times \{\Sigma \cup \Gamma\}$
q_0 : Initial state
B : Blank symbol
F : Final state

STEP 2: Logic : Convert first a and first b to X
i.e. deleting a and b pairwise.

STEP 3: Description

$Q : \{q_0, q_1, q_2, q_3, q_f\}$
$\Sigma : \{a, b\}$
$\Gamma : \{X, B\}$
$\delta : Q \times \{\Sigma \cup \Gamma\} \rightarrow Q \times \{\Sigma \cup \Gamma\} \times (L \cup R)$
$q_0 : \{q_0\}$
$B : B$
$F : \{q_f\}$

STEP 4: Transition Table

Q \ Γ	a	b	X	B
q_0	$(q_1, a/x, R)$	$(q_2, b/x, R)$	(q_0, x, R)	(q_f, B, N)
q_1	(q_1, a, R)	$(q_3, b/x, L)$	(q_1, x, R)	
q_2	$(q_3, a/x, L)$	(q_2, b, R)	(q_2, x, R)	
q_3	(q_3, a, L)	(q_3, b, L)	(q_3, x, L)	(q_0, B, R)

STEP 5: Transition diagram.