NAME: Mareena Fernandes ROLL NO: 8669 CLASS: SE IT

AT TUTORIAL NO: 10

Q.1] Variants of Turing Machine.

Ans: In a standard turing Machine the tape is semi-infinite. It is bounded on left and unbounded on the right.

* Some of the extensions are given below:-

- Tape is infinite length on both sides.
- Multiple heads in a single tape.
- Multiple tape with each tape having its own independent head.
- K-dimension tape.
- Non-deterministic turing machine.

* Two way infinite turing Machine:-

- In a standard turing machine the leftmost blanks are fixed and are shown in instantaneous description whereas, right hand side blanks are not included.
- In two way infinite turing machine both the left and right side blanks are infinite and both are not shown in instantaneous description.

* A turing machine with multiple heads.

- A turing machine with single tape can have multiple heads.
- Let us consider a tape having multiple heads

H_1 and H_2.

Each Head is capable of performing write | read | move operation.

The behaviour of this turing machine can be defined as given below.

[(State symbol under H_1, symbol under H_2) and (New state $(S_1, M_1)(S_2, M_2)$)]

where

S_1 = Symbol under H_1

S_2 = Symbol under H_2

M_1 = movement (L, R, N) of Head 1.

M_2 = movement (L, R, N) of Head 2

* Multitape Turing Machine:

- Has Multiple ~~ta~~ tapes with each tape having its own independent head.

```
B  a  b  a  a  b  b  a  b  B  B  B
         ↑

B  b  a  a  b  b  a  a  b  B  B  B
               ↑
```

- The transition can be defined as $\delta(q_1, a_1, a_2) = (q_2, (S_1, M_1), (S_2, M_2))$

q_1 = current state.

q_2 = event state

q_1 = symbol under head on tape 1

q_2 = symbol under head on tape 2

S_1 = symbol written in the current cell tape 1.

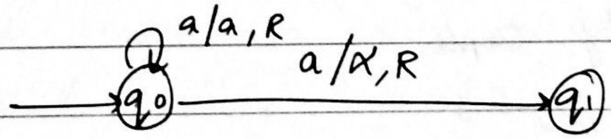
S_2 = symbol written in the current cell tape 2.

M_1 = Movement (L, R, N) of Head 1 on tape 1.
M_2 = Movement (L, R, N) of Head 2 on tape 2

* Non-Dertministic Thuring Machine:

- Non-Deterministic Turing Machine is a powerful feature.
- NDTM does not make Turing Machine powerful.
- For every NDTM, there is an equivalent Turing Machine.
- It is easy to design NDTM for certain type of problems.
- Transition behaviour:

  $(q_0, a) = \{(q_0, a, R)\ (q_1, \alpha, R)\}$



Q.2] Applications of Automata (FSM, PDA, TM) in detail with example.

Ans. * <u>FINITE AUTOMATA</u>

- A finite automation (FA) is a simple idealized machine used to recognise patterns within input taken from some charachter set.
- It doesn't have the capability to store long sequence of input alphabets.
- Finite Automata can be constructed for type 3 grammar.
- Input alphabets are accepted by reaching "final states".

- NFA can be converted into equivalent DFA
- It consists of 5 tuples $M=[Q, \Sigma, \delta, q_0, F]$
- Finite Automata can be constructed for regular language.

* PUSH DOWN AUTOMATA

- A push down automation (PDA) is a type of automation that employs a stack.
- It has stack to store the input alphabets.
- PDA can be considered for type 2 grammar.
- Input alphabets are accepted by reaching:

  ① Empty stack.

  ② Final state.
- NPDA has more capability than DPDA
- It consists of 7 ~~ty~~ tuples:

  $L = [Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$
- ~~For~~ Push down automata can be considered for context free grammar.

* TURING MACHINE

- A turing machine is type of automation that emloys an infinite tape.
- It has infinite tape to store the input alphabets.
- Turing Machine can be considered for type-0 grammar.
- Inputs alphabets are accepted by reaching.

  ① Blank tape

  ② Final state

  ③ Performing No operation like left or right traversing

- Turing Machine can be programmed turing machine is the most powerful computational model
- It consists of 7 tuples:

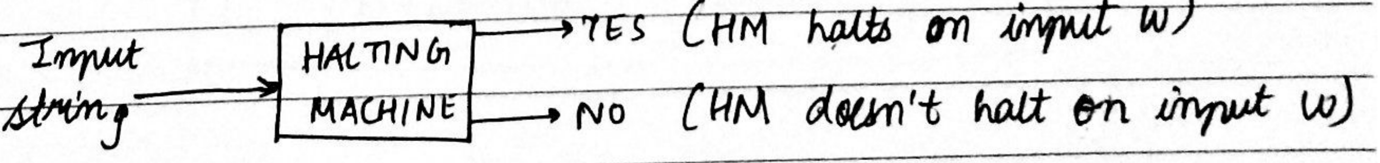
  $L = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$
- Turing machine can be constructed for regular, non-regular, context free and context sensitive language.

Q3] Halting Problem.

Ans: Input - A turing machine and an input string w.

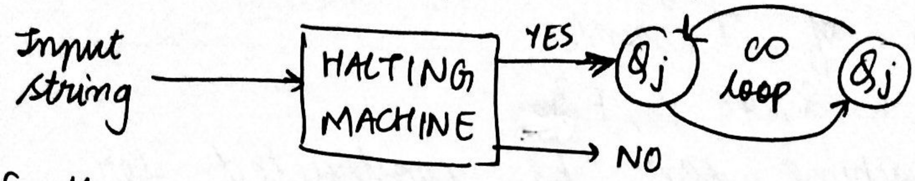
Problem - Does the Turing machine finish computing of the string w in a finite number of steps? The answer ~~is~~ must be either yes or no. This is called Halting Problem.

Proof - At first, we will assume that such a Turing machine exits to solve this problem and then we will show this contracting itself. We will ~~show~~ call this Turing machine as a Halting machine that produces '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'. The following is the block diagram of a Halting Machine.



Now we will design an inverted halting machine (HM) as -

- If H returns, YES, then loop forever.
- If H returns, NO, then halt.



- Further a machine (HM)$_2$ which input itself is constructed as follows:

If (HM)$_2$ halts on input, loop forever
Else , halt.

Here, we have got a contradiction the halting problem undecidable.

Q.4] Post Carrespondence Problem.

Ans: The Post Carrespondence Problem (PCP) introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet Σ is state as follows: Given the following two lists M and N of non-empty strings over Σ.

$M. (x_1, x_2, x_3 .... x_n)$

$N. (y_1, y_2, y_3 .... y_n)$

We can say that there is a post correspondance solution if for some $i_1, i_2, i_3 .... i_k$ $1 \leq i \leq n$ the condition $x_{i_1} = y_{i_1} ... y_{i_k}$ statistics.

Q.5] Recursive and Recursively Enumerable Language.

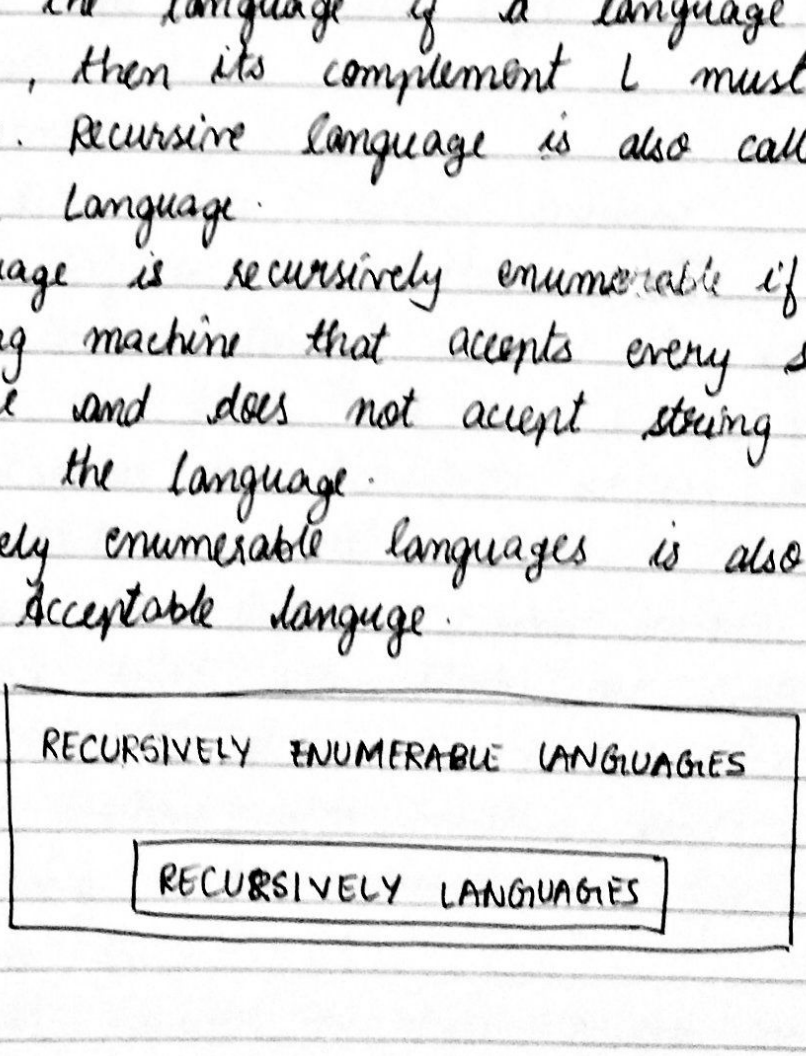
Ans: There are three possible outcome of executing a Turing machine over a given input the Turing machine may

- Halt and accept the input
- Halt and reject the input
- Never halt

A language is recursive if there exists a Turing machine that accept every string of the language and rejects every string that is not in the language if a language L is recursive, then its complement L must also be recursive. Recursive language is also called Turing Decidable Language.

A language is recursively enumerable if then exists a turing machine that accepts every string of the language and does not accept string that are not in the language.

Recursively enumerable languages is also called Turing Acceptable languge.


RECURSIVELY ENUMERABLE LANGUAGES
RECURSIVELY LANGUAGES