Anitha Sandeep.

LECTURE 27

SYNOPSIS

- Turing Machines - basics and formal definition
  Language acceptability by TM, Examples of TM

TARGET

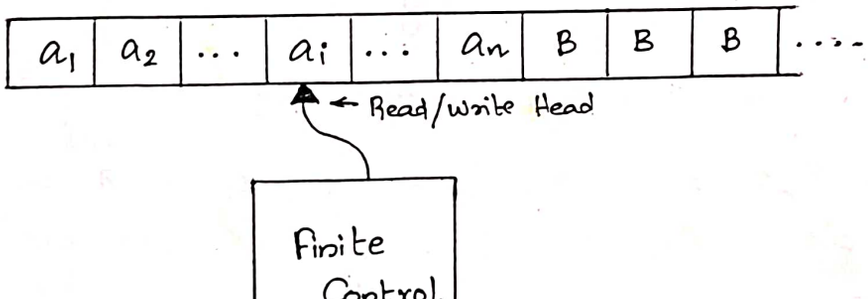
- At the end of this lecture students should be able to explain the concepts behind the machine corresponding to type-0 language.

INTRODUCTION

This lecture concentrates on a type of machine named Turing Machine (TM) that accepts type-0 language. following are the concepts to be covered.

(i) Turing Machine - basics and formal definition.
(ii) Language acceptability by TM.
(iii) Examples of TM

(i) Turing Machine - basics and formal definition



The basic model of TM is shown in the figure. It has a finite control, an input tape that is divided into cells and a read/write head that can move to left or right. Input tape is used to store the input string and the remaining cells contain blank symbol.

In one move the TM, depending upon the symbol scanned by the read/write head and the state of finite control

1. State may or may not change,
2. With or without a write operation the read/write head moves one cell left or one cell right or remains no change.

A Turing Machine is formally defined as

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

Q is the finite set of states.

Σ is the finite set of input symbols.

Γ is the finite set of tape symbols including Σ and Blank symbol

δ is the transition function that maps from

$$Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, N\}$$

q_0 is the start state that belongs to Q.

F is the set of final states. $F \subseteq Q$ and

B is the blank symbol.

Let us discuss some moves of TM.

1) $\delta(q, a) = (p, X, R)$ - from state q by reading i/p a state changes to p, writes X and head moves one cell right

2) $\delta(q, b) = (p, b, L)$ - from state q by reading i/p b state changes to p, without write operation head moves one cell left.

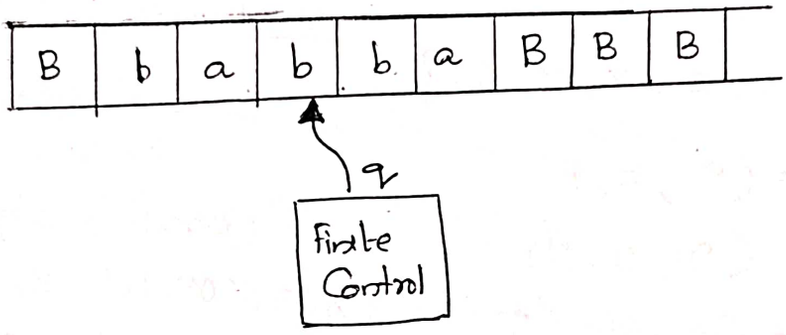
3) $\delta(q, x) = (p, y, N)$ - from state q by reading i/p x state changes to p, writes y and head remains no change.

4) $\delta(q, a) = (q, b, N)$ - from state q by reading i/p a state remains same, writes b and head remains no change.

Instantaneous Description (ID)

ID of Turing Machine is denoted as $\alpha_1 q \alpha_2$. Here q is the current state of the Machine and α_1, α_2 is a string in Γ^*. ie $\alpha_1\alpha_2$ is the string in the i/p tape and the read/write head now points to first symbol in α_2. ID is used to represent the current status of TM or used to represent the status of TM at a particular instant.

eg:



ID : baq bbaB

where ba is α_1 and bbaB is α_2

ii) Language Acceptability by TM

Language accepted by TM M, denoted as L(M) is the set of strings in Σ^* that cause M to enter a final state or halt state.

formally language accepted by M is defined as

$$L(M) = \{ w / w \text{ in } \Sigma^* \text{ and } q_0 w \vdash^* \alpha_1 p \alpha_2 \text{ for some } p \text{ in } F, \text{ and } \alpha_1 \text{ and } \alpha_2 \text{ in } \Gamma^* \}$$

Language that is accepted by TM is said to be recursively enumerable. The term enumerable derives from the fact that it is precisely these languages whose strings can be enumerated by a TM. The class of recursively enumerable languages is very broad and properly includes the CFL's.

(iii) Example

Q. Design a TM that accepts the Language $L=\{0^n1^n / n \geq 1\}$

The required TM is

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

$Q = \{q_0, q_1, q_2, q_3, q_f\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, X, Y, B\}$

$\delta(q_0, 0) = (q_1, X, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, 1) = (q_2, Y, L)$

$\delta(q_2, 0) = (q_2, 0, L)$

$\delta(q_2, X) = (q_0, X, R)$

$\delta(q_1, Y) = (q_1, Y, R)$

$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_0, Y) = (q_3, Y, R)$

$\delta(q_3, Y) = (q_3, Y, R)$

$\delta(q_3, B) = (q_f, B, N)$

verification considering sample string 000111

$q_0000111 \vdash Xq_100111 \vdash X0q_10111$
$\vdash X00q_1111 \vdash X0q_2Y11 \vdash Xq_200Y11$
$\vdash q_2X00Y11 \vdash Xq_000Y11 \vdash XXq_10Y11$
$\vdash XX0q_1Y11 \vdash XX0Yq_111 \vdash XX0q_2YY1$
$\vdash XXq_20YY1 \vdash Xq_2X0YY1 \vdash XXq_00YY1$
$\vdash XXXq_1YY1 \vdash XXXYq_1Y1 \vdash XXXYYq_11$
$\vdash XXXYq_2YY \vdash XXXq_2YYY$
$\vdash XXq_2XYYY \vdash XXXq_0YYY$
$\vdash XXXYq_3YY \vdash XXXYYq_3Y$
$XXXYYYq_3B \vdash XXXYYYq_fB$

q_0 is the start state

B is the Blank symbol and

$F = \{q_f\}$

Logic behind this design is checking off symbols. from state q_0 it reads the left most 0, changes it to X and moves to right by changing state to q_1 until it reaches the left most 1. from q_1 by reading 1, changes it to Y and state changes to q_2 and traverse left until it reaches right most X. from q_2 by reading X state rests to q_0 and

moves one cell to the right, and the above process is repeated. Hence for all 0's if it finds matching 1's it moves to the final state q_f.

Reference : Introduction to Automata Theory, Languages and Computation by John .E. Hopcroft , Jeffery .D. Ullman . (Page No 146-151)

UNIVERSITY QUESTIONS

Q1. Formally define a Turing Machine. (4 marks)

Q2. Design a Turing Machine that accepts $L=\{a^n b^n c^n : n \geq 1\}$ (15 marks)

Q3. Explain the language acceptability by a TM. (4 marks)

Q4. Design a Turing Machine that accepts $L=\{a^n b^n c^n / n > 0\}$ (10 marks)

SUMMARY

- Turing Machine - Models the computing capability of a general purpose computer. TM is studied both for the class of languages it defines and the class of integer functions it computes.

EXERCISE QUESTIONS.

Q1. Build a TM that accepts the language $L=\{a^n b^{n+1} / n > 0\}$

Q2. Build a TM that accepts the language $L=\{a^n b^{2n} / n \geq 0\}$

Q3. Construct a TM that accepts strings that belongs to $(a,b)^*$ containing substring bbb.

Q4. Design a TM that accepts strings belonging to $(0,1)^*$ that are odd palindrome.

Q5. Construct the Turing Machine for the following languages:

i) $L = \{w / w \in (a,b)^* \text{ and } |w| \text{ is even}\}$

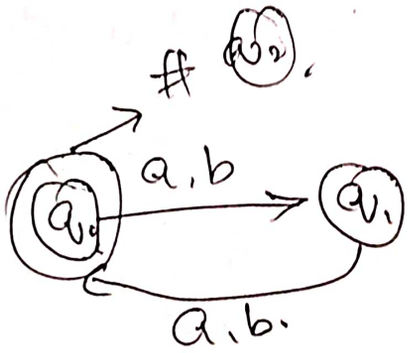
ii) $L = \{w / w \in (a,b)^* \text{ and } |w| \text{ is a multiple of 3}\}$

iii) $L = \{a^n b^m a^{n+m} / n \geq 0, m \geq 1\}$

iv) $L = \{a^n b^n a^n b^n / n \geq 0\}$

v) $L = \{ww^R / w \text{ is in } (0,1)^*\}$

vi) $L = \{w / w \in (0,1)^* \text{ and } w \text{ contains equal no. of 0's and 1s}\}$



LECTURE 28

SYNOPSIS

- Variants of TM
  - Two Way Infinite tape TM, Multi Tape TM, Multi Head TM, Multi Dimensional TM.

TARGET

- At the end of this lecture students should be able to explain the first four variants / types of TM.

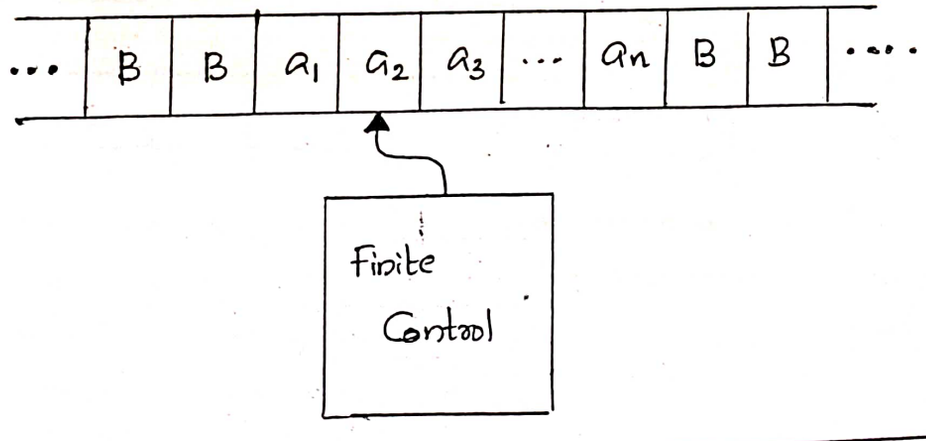
INTRODUCTION.

This lecture concentrates on the types of TM. One reason for the acceptance of TM as a general model of a computation is that the model with which we have been dealing is equivalent to many modified versions that would seem off hand to have increased computing power. In this lecture we cover the following categories:

(i) Two Way Infinite Tape TM.
(ii) Multi Tape TM.
(iii) Multi Head TM.
(iv) Multi Dimensional TM.

(i) Two Way Infinite Tape TM.

As its name implies the tape is infinite to the left as well as to the right as shown in the figure.



A Two Way Infinite Tape TM is formally defined as $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where Q is the set of states, Σ is the set of i/p symbols, Γ is the set of tape symbols, δ is the transition function that maps from $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, q_0 is the start state, B is blank symbol and F is the set of final states.

following are the points need to be noted in respect to the configuration of Two Way Infinite Tape TM:

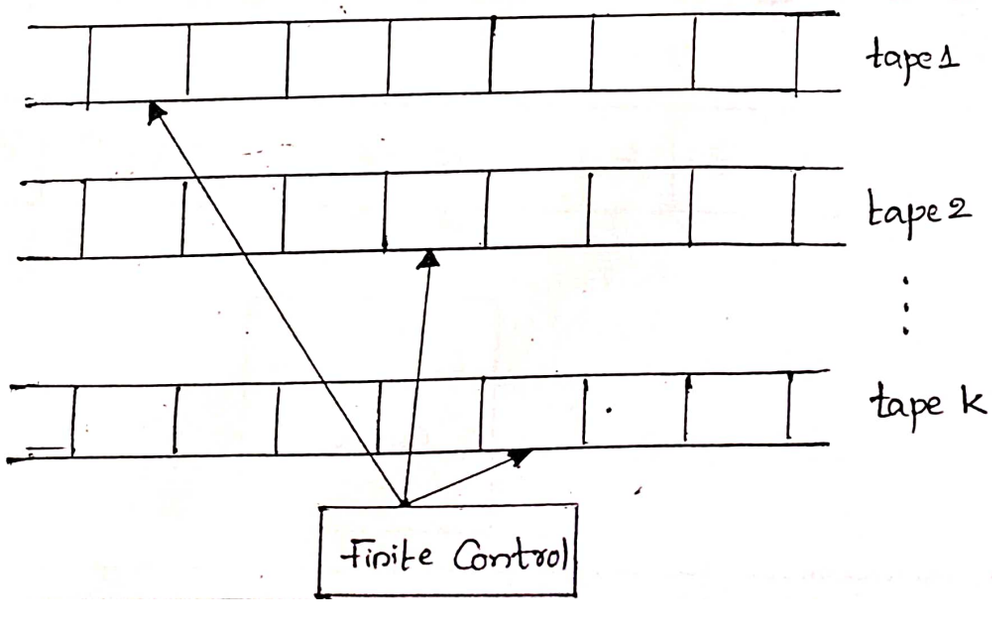
1) After storing the i/p string in the i/p tape, the cells on the right hand side and left hand side of the string are loaded with blank symbols. ie the ID of standard TM will be of the form $\alpha q_i \beta BB...B$. But in the case of Two Way Infinite Tape TM it will be like $BB...B\alpha q_i \beta BB..B$

2) As there is no left end of the tape, there is no possibility of jumping off the left end of the tape.

3) At some point of time if all the cells of the tape are B's (blank) and the state is say q_i, then the configuration is denoted as $q_i B$.

This model does not provide any additional computational capability.

(ii) Multi Tape TM

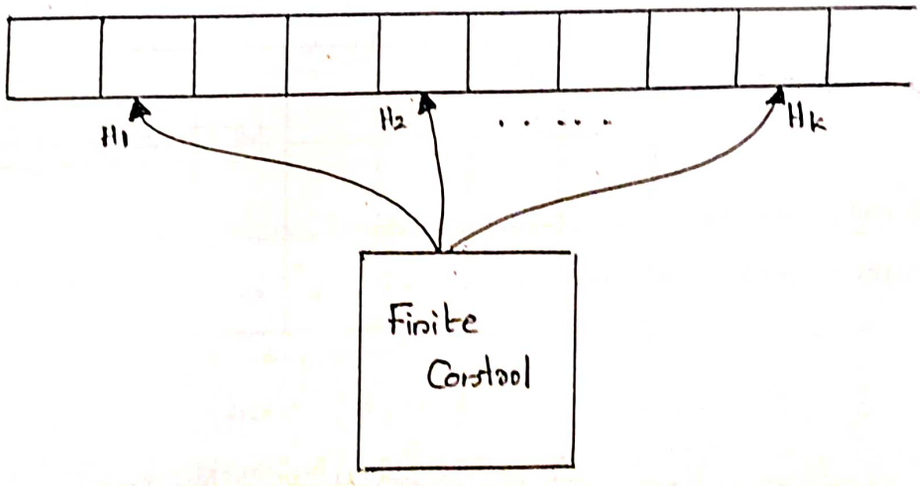


As shown in the figure a Multi Tape TM consists of a finite control with k heads and k tapes. Each tape can be either finite or infinite. On a single move depending on the state of finite control and the symbol scanned by each of the tape heads the machine can:

1. Change state.
2. Print a new symbol on each of the cells scanned by its tape heads.
3. Move each of its tape heads independently, one cell to the right or left or keep it stationary.

A k-Tape TM is formally defined as $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where Q is the set of states, Σ is the set of i/p symbols, Γ is the set of tape symbols, δ is the transition function that maps from $Q \times \Gamma^k \rightarrow Q \times (\Gamma \times \{L, R, N\})^k$, q_0 is the start state, B is the blank symbol and F is the set of final states.

(iii) Multi Head TM.

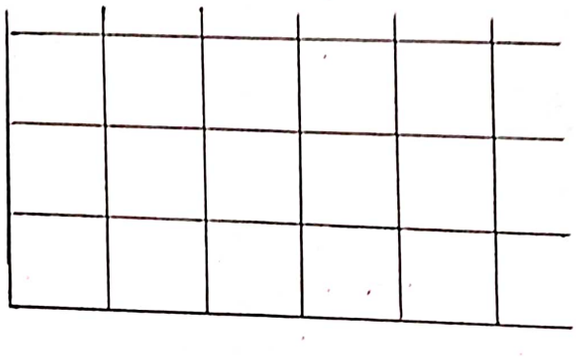


As shown in the figure a k head TM has a fixed no. of k heads. The heads are numbered 1 through k and a move of TM depends on the current state and on the k symbols scanned by each head. In one move, the heads may each move independently left, right or remain stationary.

A k-Head TM is formally defined as $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where Q is the set of states, Σ is the set of i/p symbols, Γ is the set of tape symbols, δ is the transition function that maps from $Q \times T^k \to Q \times (\Gamma \times \{L, R, S\})^k$, q_0 is the start state, B is the blank symbol and F is the set of final states.

(iv) Multi Dimensional TM.

Multi Dimensional TM has the usual finite control, but the tape consists of a k-dimensional array of cells, infinite in all 2k directions for some fixed k. Depending on the state and symbol scanned, the device changes state, prints a new symbol and moves its tape head in one of 2k directions, either positively or negatively along one of the k axes. Initially the i/p is along one axis and the head is at the left end of the i/p. At any instant, only a finite number of rows in any dimension contain non blank symbols and these rows each have only a finite number of nonblank symbols. The following figure represents a 2-D tape.



Let $q \in Q$ and $C_k \in \Gamma - \{B\}$

The configuration at a particular instant is denoted by

$$(q, (H_1, H_2), (i_1, j_1, C_{i_1 j_1}), (i_2, j_2, C_{i_2 j_2}) \cdots (i_k, j_k, C_{i_k j_k}) \cdots))$$

* Remaining part is contd in next lecture.

LECTURE 29

SYNOPSIS

- Variants of TM (contd)
  - Non Deterministic TM, Off line TM, Universal TM. Equivalence of Single Tape and Multitape TM.

TARGET

- At the end of this lecture students should be able to explain the variants/modifications of TM and should able to prove the equivalence of Single Tape and Multitape TM.

INTRODUCTION.

This lecture is a continuation of last lecture and covers the following topics:

(i) Non Deterministic TM (NDTM).
(ii) Offline TM.
(iii) Universal TM.
(iv) Equivalence of Single Tape and Multitape TM.

(i) Non Deterministic TM.

A Non Deterministic TM is a device with a finite control and a single one way infinite tape. for a given state and tape symbol scanned by the read/write head the machine has a finite number of choices for the next move. Each choice consists of a new state, a tape symbol to write and a direction of head movement. The NDTM accepts its input if any sequence of choices of moves leads to an accepting state. Non determinism allows parallel computation.

* finite no of choices for the next move

A Nondeterministic TM is formally defined as $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where Q is the set of states, Σ is the set of i/p symbols, Γ is the set of tape symbols, δ is the transition function that maps from $Q \times \Gamma \longrightarrow$ power set of $(Q \times \Gamma \times \{L, R, N\})$, q_0 is the start state, B is the blank symbol and F is set of final states.

(ii) Offline TM

An Offline TM is a multitape TM whose i/p tape is read only. Usually we surround the i/p by endmarkers, ¢ on the left and $ on the right. The TM is not allowed to move the i/p tape head off the region between ¢ and $. It should be obvious that the offline TM is just a special case of multitape TM and therefore is no more powerful than any of the models we have considered.

(iii) Universal TM.

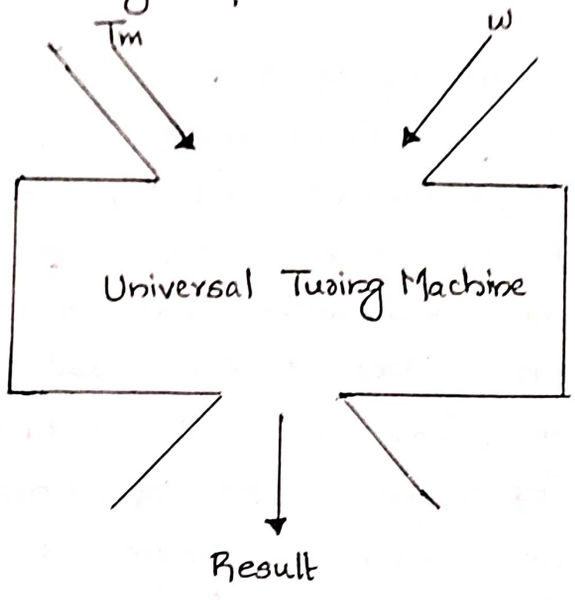
There is a certain generic TM that can be programmed about the same way that a general purpose computer can, to solve any problem that can be solved by turing machine. The program that makes this generic machine behave like a specific machine Tm will have to be a description of TM. In other words we shall be thinking of the formalism of turing machines as a programming language in which we can write programs. Programs written in this language can then be interpreted by universal machine that is to say another programs in same language.

To begin we must present a general way of specifying turing machines so that their descriptions can be used as input to other turing machines.

Universal TM 'U' takes two arguments, a description of a machine Tm, "Tm" and a description of an input string w, "w". We want U to have the following property:

U halts on input "Tm" "w" if and only if Tm halts on input w.

U("m" "w") = "m(w)" is the functional notation of universal Turing Machine.



Universal Turing machine interprets all Turing machine

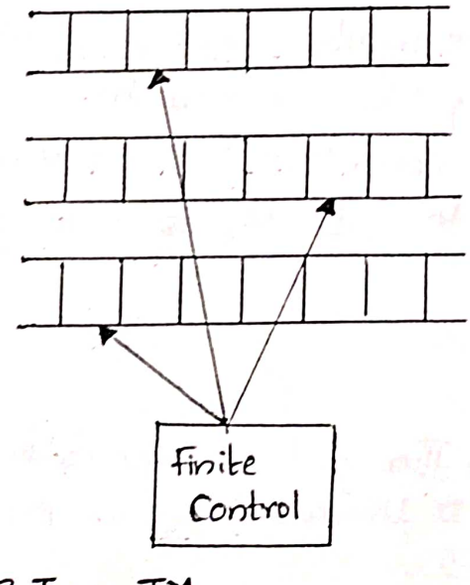
Tm : Desc of Turing Machine

w: Input String

U halts for Tm, w if Tm has final state halt for w

(iv) Equivalence of Single Tape and Muti Tape TMs.

Theorem: If a language L is accepted by a MultiTape TM it is accepted by a Single Tape TM.



3 Tape TM

Head 1		X					
Tape 1	A_1	A_2	..	..	..		Am
Head 2					X		
Tape 2	B_1	B_2	..	..	..	..	Bm
Head 3	X						
Tape 3	C_1	C_2	..	..	...	..	Cm

Simulation of 3 Tape by One Tape

Proof: Let L be accepted by M_1, a TM with k tapes. We can construct M_2, a single tape TM with 2k tracks, two tracks for each of M_1's tapes. One track records the contents of the corresponding tape of M_1 and the other is blank, except for the marker in the cell that holds the symbol scanned by the corresponding head of M_1. The finite control of M_2 stores the state of M_1, along with a count of number of head markers to the right of M_2's tape head.

Each move of M_1 is simulated by a sweep from left to right and then from right to left by the tape head of M_2. Initially M_2's head is at the leftmost cell containing a head marker. To simulate a move of M_1, M_2 sweeps right visiting each of the cells with head markers and recording the symbol scanned by each head of M_1. When M_2 crosses a head marker it must update the count of head markers to its right. When no more head markers are to the right, M_2 has seen the symbols scanned by each of M_1's heads, so M_2 has enough information to determine the move of M_1. Now M_2 makes a pass left, until it reaches the left-most head marker. The count of markers to the right enables M_2 to tell when it has gone far enough. As M_2 passes each head marker on the leftward pass it updates the tape symbol of M_1 scanned by the head marker and it moves the head marker one symbol left or right to simulate the move of M_1. Finally M_2 changes the state of M_1 recorded in M_2's control to complete the simulation of one move of M_1. If the new state of M_1 is accepting then M_2 accepts.

Reference: Introduction to Automata Theory, Languages & Computation by John. E. Hopcroft, Jeffery D Ullman (Page no 159–166)

UNIVERSITY QUESTIONS.

Q1. Explain Offline Turing Machine (4 marks)

Q2. Formally define a deterministic one tape TM (5 marks)

Q3. Prove that if a language L is accepted by a multitape TM, it is accepted by a single tape TM (15 marks)

Q4. What are Offline Turing Machines? (4 marks)

Q5. Explain Multitape Turing Machine. (5 marks)

Q6. What are the different types of Turing machines? Explain each. Also give suitable examples for each. (20 marks)

Q7. Define Universal TM and Universal language. (6 marks)

SUMMARY

- Variants of TM – Two Way Infinite Tape, Multitape, Multihead, Multidimensional, Non Deterministic, Offline, Universal

LECTURE 30

SYNOPSIS

- Recursive Language

TARGET

- At the end of this lecture students should able to explain Recursive Language and its properties.

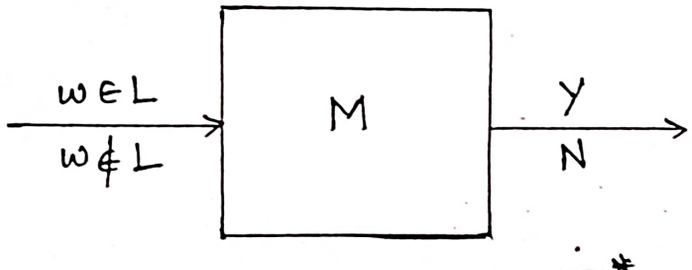
INTRODUCTION.

This lecture focus on a class of language called Turing decidable / Recursive language and its properties.

Recursive Language.

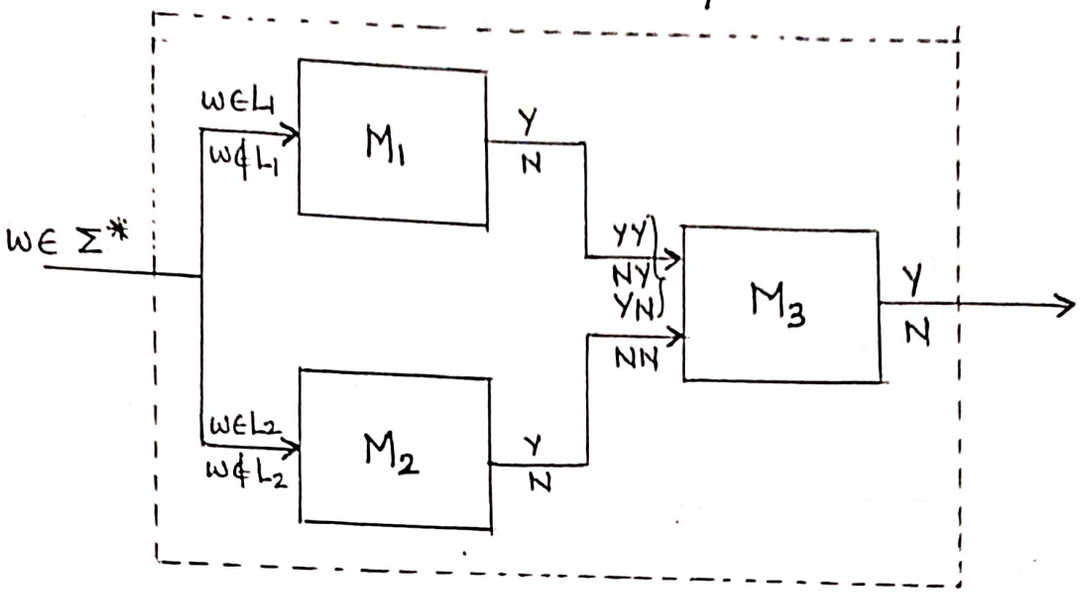
Definition: A language $L \subseteq \Sigma^*$ representing a problem over Σ, is said to be Recursive Language, if there exist a Turing Machine M which always halts when given any input $w \in \Sigma^*$, whether $w \in L$ or $w \notin L$. If $w \in L$ then M halts will output Y indicating that the string w is in L, and if $w \notin L$ then M halts will output N indicating that w does not belong to L.

Recursive Language is also known as Turing Decidable Language. The following figure shows the concept behind Recursive Language.



where $L \subseteq \Sigma^*$ and $w \in \Sigma^*$

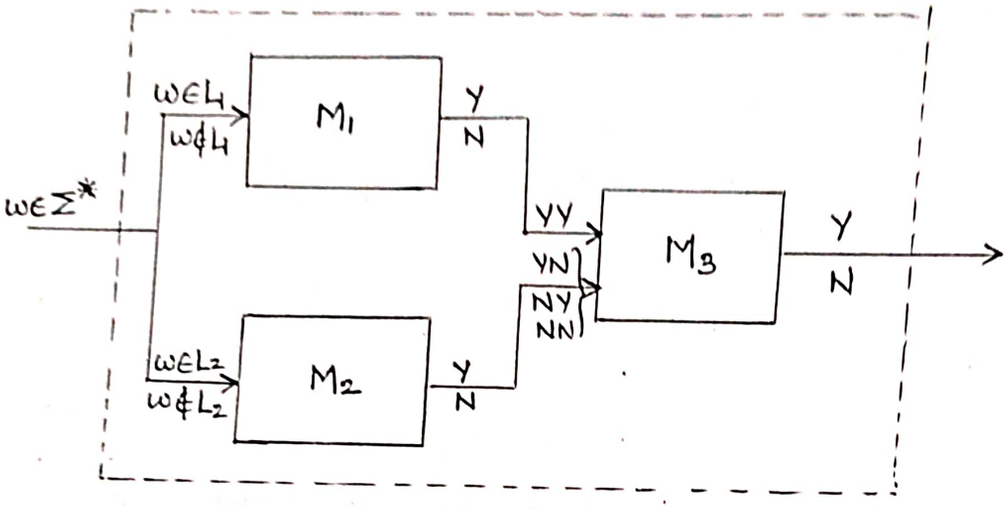
Theorem 1: If L_1 and L_2 are Recursive Languages then $L_1 \cup L_2$ is also Recursive.



Let M_1 be a TM for deciding the language L_1 and M_2 be a TM for deciding the language L_2. Now construct a new TM M_3 having $\{Y, N\}$ as the set of symbols. M_1, M_2 and M_3 are arranged as shown in figure. Input to M_3 will be either YY, YN, NY or NN. The machine M_3 halts by producing output Y if the i/p contains atleast one Y otherwise M_3 halts by producing output N. ie Machine M_3 returns Y as output if atleast one of the outputs of M_1 or of M_2 is a Y.

ie if $w \in L_1$ or $w \in L_2$ or $w \in$ both L_1 & L_2 the machine M_3 halts by producing Y, else M_3 halts by producing N. Hence $L_1 \cup L_2$ is also recursive since there exist a machine that halts by producing o/p Y for strings $w \in L_1 \cup L_2$ and halts by producing o/p N for strings $w \notin L_1 \cup L_2$

Theorem 2: If L_1 and L_2 are recursive then $L_1 \cap L_2$ is also recursive.

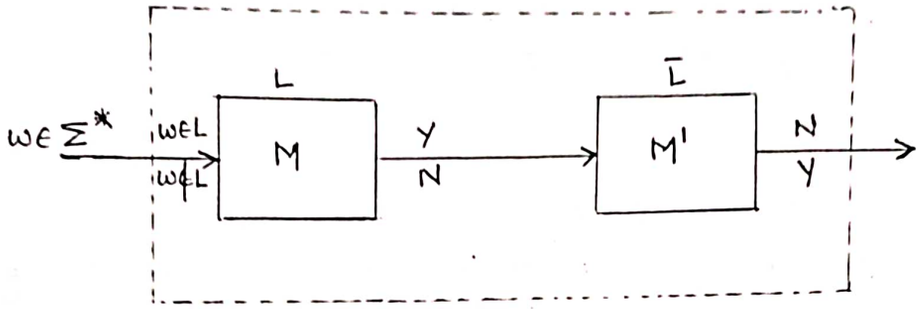


Let M_1 be a TM for deciding L_1 and M_2 be a TM for deciding L_2. Now construct a TM M_3 having {Y, N} as the set of symbols. M_1, M_2 and M_3 are arranged as shown in figure. Input to M_3 will be either YY, YN, NY or NN. The machine M_3 halts by producing o/p Y if the i/p contains only Y, otherwise M_3 halts by producing o/p N. ie M_3 returns Y as output if both M_1 and M_2 produces o/p Y.

ie If w∈ w belongs to both L_1 and L_2 then M_3 halts by producing o/p Y, else M_3 halts by producing o/p N.

Hence $L_1 \cap L_2$ is also recursive since there exist a machine M_3 that halts by producing o/p Y for i/p strings $w \in L_1 \cap L_2$ and halts by producing o/p N for i/p strings $w \notin L_1 \cap L_2$.

Theorem 3: If L is recursive language, then $\bar{L}$ is also recursive



Let M be a TM for deciding L. Now construct a machine M' having {Y, N} as the set of symbols. M and M' are arranged as shown in figure. Input to M' will be either Y or N. The machine M' halts by producing o/p N if the i/p is Y and M' halts by producing o/p Y if the i/p is N.

ie $w \in \Sigma^*$ is applied as i/p to the above arrangement

If $w \in L$ then machine M halts by producing o/p Y and M' halts by producing o/p N.

If $w \notin L$ then machine M halts by producing o/p N and M' halts by producing o/p Y.

Hence $\bar{L}$ ie $\Sigma^* - L$ is also recursive since there exist a machine that halts by producing o/p Y for strings $w \in \bar{L}$ and halts by producing o/p N for strings $w \notin \bar{L}$.

Reference: Introduction to Automata Theory, Languages & Computation by John. E. Hopcroft, Jeffery. D. Ullman (page No: 178 – 181)

UNIVERSITY QUESTIONS.

Q1. Show that the union of two recursive languages is recursive (4 mar

SUMMARY

- Recursive Language – A language $L \subseteq \Sigma^*$ is said to be recursive, if there exist a TM that halts by producing o/p Y for i/ps $w \in L$ and halts by producing o/p N for i/ps $w \notin L$. Recursive language is also known as Turing Decidable language.
- Recursive Language is closed under union, intersection and complementation.

LECTURE 31

SYNOPSIS

- Recursively Enumerable Language.

TARGET

- At the end of this lecture students should be able to explain Recursively Enumerable Language and its properties.

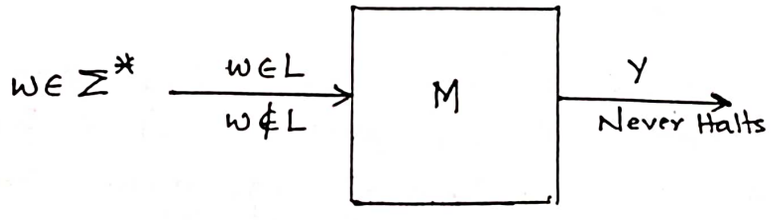
INTRODUCTION.

This lecture gives emphasis on a class of language called Turing Acceptable / Recursively Enumerable language and its properties.

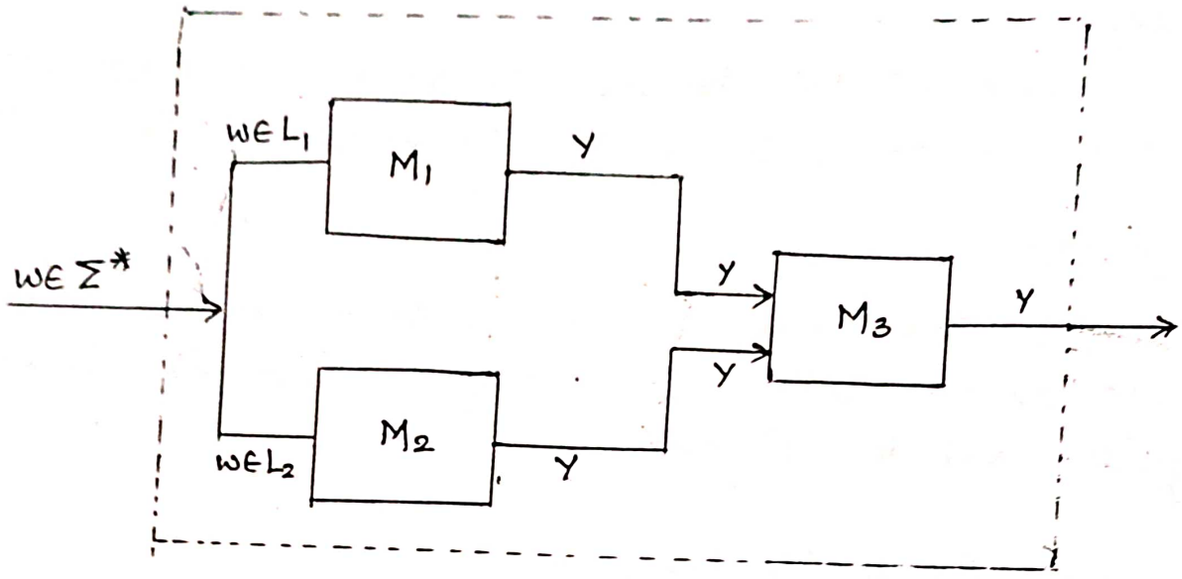
Recursively Enumerable Language.

Definition: A language $L \subseteq \Sigma^*$ is said to be Recursively Enumerable Language, if there exist a Turing Machine M that halts by producing o/p Y for input strings $w \in L$ and never halts for input strings $w \notin L$. ie If $w \notin L$ then M continues in an infinite loop.

Recursive Enumerable Language is also known as Turing Acceptable Language.



Theorem 1: If L_1 and L_2 are Recursively Enumerable Languages then $L_1 \cup L_2$ is also Recursively Enumerable.

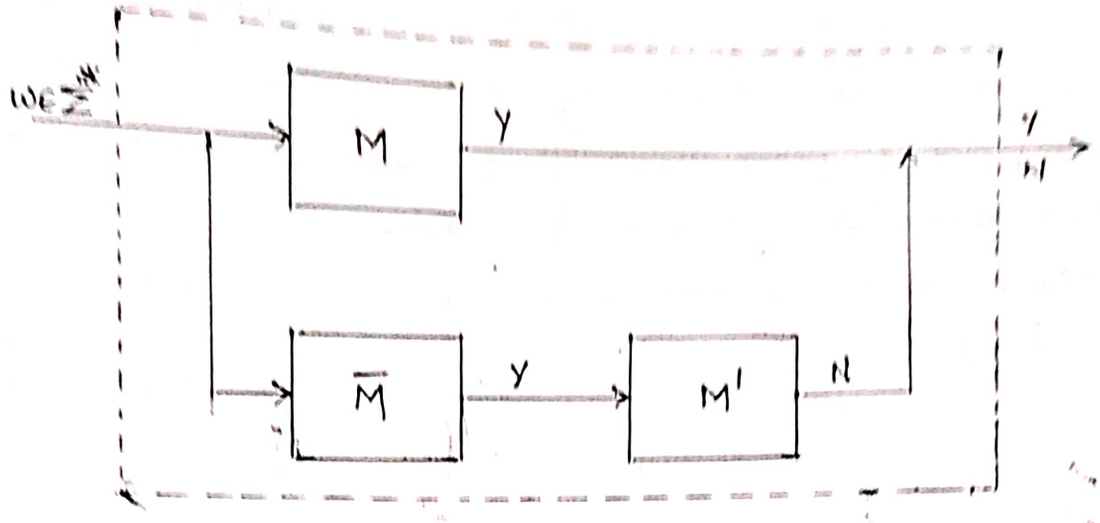


Let M_1 be a TM for accepting L_1 and M_2 be a TM for accepting L_2. Now construct a TM M_3 having $\{Y\}$ as the set of i/p symbol. M_1, M_2 and M_3 are arranged as shown in figure.

If at any stage, there is an o/p from any one of M_1 or M_2, then on the first o/p from either M_1 or M_2, the machine M_3 is activated and the o/p from M_1 or M_2, whichever is available is written on the tape of M_3. If the i/p tape of M_3 contains a symbol Y then the machine M_3 halts by producing o/p Y. Else it continues in an infinite loop, indicating that $L_1 \cup L_2$ is recursively enumerable.

Hence $L_1 \cup L_2$ is recursively enumerable since there exist a machine that halts by producing o/p Y for i/ps $w \in L_1$ or $w \in L_2$ or $w \in$ both L_1 & L_2. Otherwise the machine continues in an infinite loop.

Theorem 2: For a given language L, if both the Languages L and $\bar{L} = \Sigma^* - L$ are Turing Acceptable then L is Turing Decidable.



Let M and $\bar{M}$ be the TMs that accept languages L and $\bar{L}$ respectively. The machines M and $\bar{M}$ are arranged as shown in figure. In addition to M and $\bar{M}$, a machine M' is arranged in series with $\bar{M}$ that inverts the o/p of $\bar{M}$.

The overall machine UM with the above configuration and description will be able to decide the language L. Whenever an i/p string $w \in \Sigma^*$ is received by UM, its control unit writes w on the tape of both M and $\bar{M}$ and activates both M and $\bar{M}$. Whenever an o/p if at all, comes out of M or $\bar{M}$ then the overall machine UM gives o/p and halts.

If $w \in L$ then halts with o/p Y. In this case UM halts by producing o/p Y. If $w \in \bar{L}$ then $\bar{M}$ halts by producing o/p y which is complemented by M' and UM halts by producing o/p N. Hence L is turing decidable since there exist a machine UM that halts by producing o/p Y for i/p $w \in L$ and halts by producing o/p N for i/p $w \notin L$.

Reference: Introduction to Automata Theory, Languages & computation by John. E. Hopcroft, Jeffery. D. Ullman. (Page No: 178-181)

UNIVERSITY QUESTIONS.

Q1. Write a short note on recursively enumerable languages. (5 marks)

Q2. Show that the union of two recursively enumerable languages is recursively enumerable. (4 marks)

Q3. Show that if a language L and its complement are both recursively enumerable, then L is recursive (4 marks)

Q4. Show that the universal language is recursively enumerable (10 marks)

SUMMARY

- Recursively Enumerable Language – A language $L \subseteq \Sigma^*$ is said to be Recursively Enumerable, if there exist a TM that halts by producing o/p Y for i/p $w \in L$ and never halts for i/p $w \notin L$.
- Union of two Recursively Enumerable language is also recursively enumerable.
- If L and $\bar{L}$ are Recursively Enumerable then L is recursive.

LECTURE 32

SYNOPSIS

- Decidable and Undecidable problems, Halting problem, Reducibility

TARGET

- At the end of this lecture students should able to answer questions regarding decision problems.

INTRODUCTION.

This lecture gives importance to Decidable and Undecidable problems, Halting problem, Reducibility.

Decidable and Undecidable problems.

Problems of the class for which there exist a solution either yes or no are called as decision problems. Decision problems are classified in to two: Decidable and Undecidable problems.

Decidable problems are problems for which there exist an algorithm that produces an output either yes or no. If there exist no such algorithm for the problem then it is called as undecidable problem.

A problem P is said to be Decidable / Solvable if the language $L \subseteq \Sigma^*$ representing the problem is Turing Decidable or Recursive, Otherwise the problem is Undecidable.

Consider the following problem based on Fermat's Conjecture. Is there no solution in positive integers to the equation $x^i + y^i = z^i$ if $i \geq 3$? Note that x, y, z & i are not parameters but bound variables in the statement of the problem. There is one Turing Machine that accepts any input and one that rejects any input. One of these answers fermat's Conjecture correctly even though we do not know which one. Hence fermat's Conjecture is Decidable.

Yes or No is given then Decision proputi is proved.

Algorithm → Yes or No { Decidable problem }

Consider another problem related to universal turing machine: "Does Turing Machine M accept input w?". Here both M and w are parameters of the problem. In formalizing the problem as a language we shall restrict w to be over alphabet {0,1} and M to have tape alphabet {0,1,B}. As the restricted problem is undecidable, the more general problem is surely undecidable.

Halting Problem

Given the description of turing machine Tm and an input w, when started in the initial configuration q_0w, perform a computation that eventually halts? The domain of this problem is to be taken as the set of all turing machines and all w; ie we are looking for a single turing machine that, given the description of an arbitrary Tm and w, will predict whether or not the computation of Tm applied to w will halt.

We cannot find the answer by simulating the action of Tm on w, say by performing it on universal turing machine, because there is no limit on the length of computation. If Tm enters an infinite loop then no matter how long we wait, we can never be sure that Tm is in fact in a loop. It may be simple case of very long computation. What we need is an algorithm that can determine the correct answer for any Tm & w by performing some analysis on the machines description and the i/p. But it is clear that no such algorithm exists. In short halting problem is to determine for an arbitrary given turing machine Tm and input w, whether Tm will eventually halt on i/p w.

Reducibility.

A reduction is a way of converging one problem in to another problem in such a way that a solution to the second problem can be used to solve first problems. Reduction doesn't mean to make smaller, it means to transform or convert a problem X to another problem Y that is atleast as hard as X. Usually Y is atleast as hard as X and so we express a reduction from X to Y as $X \leq Y$

Use of Reducibility to show that a problem is unsolvable.

Reducibility is the primary method for proving that a problem is computationally unsolvable. Here is how it works:

a) Reduce an unsolvable problem X to a more difficult or general problem Y.

b) Then show that if you could solve the more general problem Y, you could use the solution to Y to solve the more specific problem X, because Y is atleast as hard as X.

c) Conclude that since X is known to be unsolvable, and the solution to Y would also solve problem X then Y must be unsolvable too.

Reducibility and Decidability/Solvability.

Reducibility also plays an important role in classifying problems as decidable or undecidable and later on in complexity too.

- When X is reducible to Y, Solving X can't be harder than solving Y, because the solution to Y can also be used to solve X.
- In terms of the decidability/computability question, then
  (a) If X is reducible to Y and Y is decidable, then X is also decidable.

(b) Equivalently, if X is undecidable / unsolvable and X reduces to Y then Y is also undecidable. This is the key to proving that problem are undecidable / unsolvable.

Reference : 1. Introduction to Automata Theory, Languages and Computation by John. E. Hopcroft, Jeffery. D. Ullman (Page No: 177-179, 181)

2. An Introduction to Automata Theory & formal Languages by Adesh. K. Pandey (Page No: 249-253)

UNIVERSITY QUESTIONS.

Q1. What is meant by halting problem? (4 marks)

Q2. Show that halting problem is undecidable. (10 marks)