

Step 3

Construct PDA


 $(q, 010^4, S) \vdash (q, 010^4, 0CC) \vdash (q, 10^4, CC)$ $\vdash (q, 010^4, 1SC) \vdash (q, 0^4, SC)$ $\vdash (q, 0^4, 0CCC) \vdash (q, 0^3, CCC)$ $\vdash (q, 0^3, 0CC) \vdash (q, 0^2, CC)$ $\vdash (q, 0^2, 0C) \vdash (q, 0, C)$ $\vdash (q, 0, 0) \vdash (q, \epsilon, \epsilon)$ Thus $010^4 \in N(A)$

□□□

CHAPTER**6****Undecidability****University Prescribed Syllabus**

The Church-Turing thesis, Universal Turing Machine, Halting Problem, Introduction to Unsolvable Problems.

Syllabus Topic : The Church-Turing Thesis**6.1 Church Turing Thesis**

 Write Short note on Church Turing Thesis.

- Many mathematicians were finding if there is a procedure, algorithm for solving a problem that can be implemented on Turing machine. Alonzo Church also worked on same problem and he solved some problems. Chomsky unrestricted grammar recursive recursion function theory by λ calculus.
- If a computational problem P is solvable by human being then it is solved by computer and if it is solvable by computer then it is solved by Turing machine.
- Mr. Church formalised simple hypothesis over it. If a computational problem P is solvable by human being then it is directly solved by Turing machine.

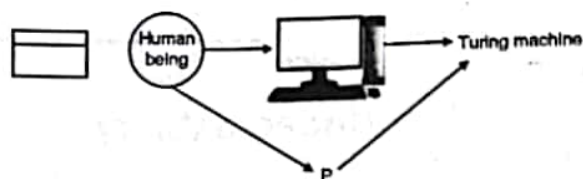


Fig. 6.1.1

Above formalism leads to,

1. **Turing theory (Weak form)** : A turing machine can compute anything that can be composed by digital computer.
2. **Turing theory (strong form)** : A turing machine can compute anything that can be computed.

Define string,

$D = \{p : P \text{ is a polynomial with an integral root} \}$

Consider turing machine,

$M = \{ \text{The input is a polynomial over variable } x_1, x_2, \dots, x_n \}$

- Evaluate P on n tuple of integers.
- If p ever evaluates to 0, accept
- M recognized D but does not decide D.

The Church Turing thesis explain similarities between recursive function and computable one,

1. Recursive function is a mathematical concept.
2. Computable function is more of constructivism.

Now it is universally accepted by computer scientists that turing machine is a mathematical model of an algorithm.

Syllabus Topic : Universal Turing Machine

6.2 Universal Turing Machine (UTM)

Write Short note on UTM.

- The limitations of Turing machine execute only one program at a time hence it must be constructed for every new computation to be performed for a every input output relation.

- This is why Turing machine introduce as a solution could be used to work like a simulator a computer with an arbitrary program.
- This Turing machine is capable to simulate the action of any Turing machine on any input, is called **Universal Turing Machine**.

6.2.1 Attributes of UTM

- ✓ Reprogrammable machine
- ✓ Simulates any other Turing Machine

UTM gives following information in its tape,

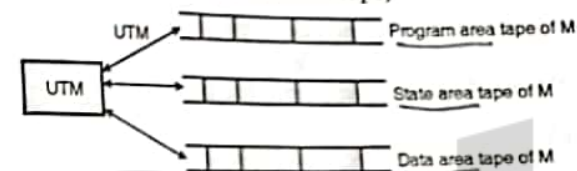


Fig. 6.2.1

1. The description of in terms of its machine.
2. The initial configuration of M i.e. starting state and the start symbol, (state area of tape).
3. The processing data (input string) to be fed to M (Data are tape of M) UTM algorithm.
 1. Scan the state area of the tape and read the symbol (O are tape of M) from the same area that M reads to start with and also the starting or initial state of M.
 2. Move side tape to the program area contains the description of M and find the row in machines headed by the state symbol read in step 1.
 3. Find the column headed by the input symbol read in step 1 and read the triplet (new symbol (a), new state (a), direction to move) stored as the entry (current state, current input symbol), which is the entry at the intersection of row and column f and in step 2.
 4. Move the tape to reach the appropriate cell in the data area, replace the symbol by new symbol, move the head in required direction, read the next symbol and finally reach the state area and replace the state scanned symbol then goto step 1.

- UTM has, linear tape as any other turing machine hence it cannot be directly put finite machine of M on this tape, because finite machine is a two dimensional matrix, it has to be converted into one directorial form.
- In UTM we encode M as a string of symbols line,

(1) Alphabet encoding

Symbol	a	b	c	d
	↓	↓	↓	↓
Encoding	1	11	111	1111

(2) State encoding

Store	q ₁	q ₂	q ₃	q ₄
	↓	↓	↓	↓
Encoding	1	11	111	1111

(3) Head move encoding

Move	L	R
	↓	↓
Encoding	1	11

We can perform transition encoding by using above symbols
e.g. transitions.

	$\delta(q_1, a) = (q_1, b, L)$
ncoding	↓
	1#1#11#11#1
	↑ ↑ ↑ ↑
	Separator

Same as in turing on active encoding.

Transition $\delta(q_1, a) = (q_2, b, L)$ $\delta(q_2, b) = (q_3, C, R)$

Encoding	↓	↓	↓
	1#1#11##11#1##	11#11#111#111#11	
		↑	
		Separator	

turing machine encodes transition in binary format of the simulator machine M.

1#1#11#11#1##11#11#111#111#11#11##....

A turing machine is described with a binary string of 0's and 1's therefore the set of Turing machines forms a language. Each string of this language is the binary encode of a Turing machine.

E.g. Langue to TURING MACHINE

L₂ { #1#1#11#1, Turing machine 1
##1##1##1#1111 Turing machine 2
..... }

6.2.2 Composite Turing Machine and Iterated Turing Machine

Write Short note on Composite and Iterated Turing Machine.

- Two or more turing machine can be obtained to solve a collection of a simpler problems, so that the output of one Turing machine forms the input to the next Turing machine and so on. This is called as **composition**.
- This is used to break the complicated job into number of jobs implementing each separately and then combing them together to get answer for the job required to be done.

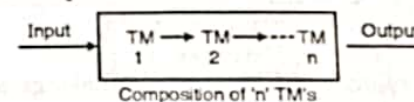


Fig. 6.2.2

- Another any having a combination in turing machine is by applying its own output as input repetitively. This is called **iteration or recursion**.

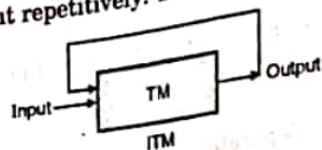


Fig. 6.2.3

Syllabus Topic : Introduction to Unsolvable Problems

6.3 Introduction to Unsolvability Problems

6.3.1 Recursive and Recursive Enumerable Languages

- We are well known about recursive function, how we will see about and R.E. languages.
- Language accepted by turing machine M is set of all strings of input symbols which are accepted by turing machine M . for each string of input symbols there are three possibilities with M .

1. Turing machine will halt on final state (Accept)
2. Turing machine will halt on non final state
3. Turing machine will enter an infinite loop.

1. Recursive enumerable language

Write a short note on Recursive Enumerable Language.

- A language L is recursive enumerable if there exists some Turing machine M to accept L .
- If there string of input symbols which belongs to language L will be accepted by Turing machine M .

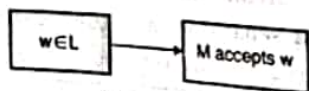


Fig. 6.3.1

- If string of input symbols which does not belongs to long L , the Turing machine will be either take half in non final state or enters in an infinite loop.

- All the languages which are accepted by Turing machine are recursive enumerable.

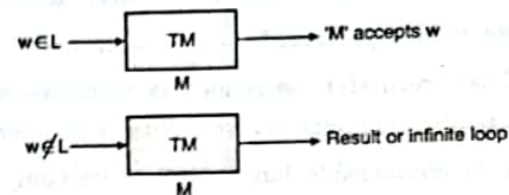


Fig. 6.3.2

2. Recursive language

A language L is recursive language which is decidable if some Turing machine M decides it. In this case M satisfies following conditions,

1. M accepts w if $w \in L$
2. M rejects w if $w \notin L$

In recursive language, Turing machine will decide the language is in particular format or not. For e.g. if there is CFL, so there are rules for recognizing it. Turing machine will decide that language is CFL or not using all the symbols of CFLs.

So this language is recursive.

Every recursive language is also recursively enumerable (r.e)

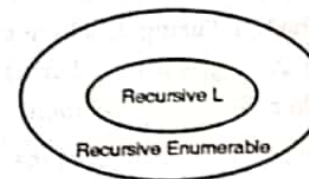


Fig. 6.3.3

If language is recursive then it is also recursively enumerable, but if the language is recursively enumerable language that we can not say it is recursive language.

Recursive and Recursive Enumerable languages are closed under intersection complementation and union. It's property states that,

1. If L is recursive, it's complement L' is also recursive.
2. The union of two recursive languages is recursive and union of two recursive enumerable language is also recursive enumerable.
3. If L is recursive enumerable language and its complement L' is also recursive enumerable then L is recursive language.
4. If L_1 and L_2 are two recursive enumerable languages then $L_1 \cup L_2$ and $L_1 \cap L_2$ is also recursive enumerable.

Hence recursive languages is decidable, computable and solvable language

e.g. RL and CFL and Recursive Enumerable language known as Trurry recognizable or partially decidable or semi decidable language because if $w \in L$, the Turing Machine either rejects or goes in loop.

6.3.2 Unsolvability

☞ Explain the concept of unsolvability.

☞ What do you mean by decidability and undecidability? Explain it with Examples.

The Turing Machine plays an important role in computation of functions and relating classes. Even though Turing Machine computability is great, or solvable there is limitation of Turing Machine. Turing Machine faces decidable and undecidable or unsolvable problems of language.

Decidable problem finds solution, is definite means either yes or No, but, undecidable problem gives undecidable solution means sometime yes and sometime No.

Decidability and Undecidability is analogous to solvability and unsolvability e.g. Does earth move around the sun? Does winter come after rainy season? Solution of these problems are solvable or decidable and we can see another e.g. Will tomorrow be a rainy day? You cannot give confirm

answer for this question. May or may not. It totally depends on nature. So these type of problems are undecidable or unsolvable.

Decidable problems in language like,

- Does FA accept regular language?
- If L_1 and L_2 are two regular languages, are they closed under union, concatenation or kleene closure?
- If L_1 and L_2 are two CFL, then $L_1 \cup L_2$ is CFL?
- Similarly we can see undecidable problems in language like,
 1. For given CFG G is ambiguous or not?
 2. For two given CFL L_1 and L_2 whether $L_1 \cap L_2$ is CFL or not?

Undecidable problems

The undecidable problem in computation can be understood by modifying or taking specific case and operating algorithms.

Theorem 1 :

There exists a language over Σ that is not recursively enumerable.

Proof :

Let, $L = T(M)$

Where, $L \rightarrow$ Recursive Enumerable language

$T(M) \rightarrow$ Turing Machine for M

$\Sigma \rightarrow$ Finite set of input

$\Sigma^* \rightarrow$ countable set of input (One to one correspondence between Σ^* and M)

As turing machine is defined with 7 tuple,

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

Where, M is encoded as a string and Q, Σ, Γ etc are finite set. So the set I of all Turing Machine is countable.

- Let I be set of all languages over Σ and $I \subset \Sigma^*$ which is infinite



i.e. I is uncountable or not in one to correspondence with N .

$$I = \{L_1, L_2, \dots\} \text{ which is finite set}$$

$$\text{and } \Sigma^* = \{w_1, w_2, \dots\}$$

such that any random language state L_1 can be represented as infinite binary sequence,

$$\text{as } x_{11}, x_{12}, \dots, x_{1n}$$

$$\text{where, } x_{ij} = \begin{cases} 1 & w_j \in L_i \\ 0 & \text{otherwise} \end{cases}$$

Using this representation L_1 as an infinite binary sequence.

$$L_1 : x_{11} \ x_{12} \ x_{13} \ \dots \ x_{1j} \ \dots$$

$$L_2 : x_{21} \ x_{22} \ x_{23} \ \dots \ x_{2j} \ \dots$$

:

$$L_i : x_{i1} \ x_{i2} \ x_{i3} \ \dots \ x_{ij} \ \dots$$

Representation of I .

Let $L \subset \Sigma^*$ defined by y_1, y_2, y_3, \dots

Where $y_i = 1 - x_{ii}$

$$\text{if } x_{ii} = 0$$

$$y_i = 1$$

$$\text{if } x_{ii} = 1$$

$$y_i = 0$$

As per our assumption $L \subset \Sigma^*$ represented by infinite binary sequence should be L_k for random k natural number but $L \neq L_k$ since $w_k \in L$.

$$\text{iff } w_k \notin L_k$$

This contradicts our assumption that I is uncountable because I is countable.

As I is countable L should have some member not corresponding to any Turing Machine in I .



This proves the existence of language over Σ is not recursively enumerable.

We now prove a Turing machine language, ATM .

$$ATM = \{(M, w) \mid M \text{ is a Turing machine and accepted } w\}$$

Theorem 2 :

ATM Machine is undecidable.

Proof :

We can prove that Alternating Turing Machine is undecidable by contradiction.

1. Let's we have to prove that Alternating Turing Machine is recursively enumerable or not construct a Turing Machine U which has input (M, w) . Simulate M on w . If M enters in accepting state. Turing Machine U accepts input (M, w) . So that we can say Alternating Turing Machine is recursively enumerable.
2. Now assume H be a Turing machines which decides Alternating Turing Machine and construct Turing Machine D that uses H as subroutine.

$D =$ 1) Input (M)

2) Construct the string $(M, (M))$

3) Run H on input $(M, (M))$

a) If H accepts, then reject

b) If H rejects, then accept.

Now, $D((M))$ described as follows,

$$D((M)) = \begin{cases} \text{accept if } M \text{ does not accept } (M) \\ \text{reject if } M \text{ accepts } (M) \end{cases}$$

By referring action of D on the input (CD) . According to the construction of D

$$D((D)) = \begin{cases} \text{accept } (D) \text{ does not accept by } D \\ \text{reject } (D) \text{ accept by } D \end{cases}$$



Construction of D is stated contradiction hence Alternating Turing Machine is undecidable.

In this theorem Turing machine U is used in the sense of universal Turing machine since it is simulating any other Turing Machine.

Syllabus Topic: Halting Problem

6.4 Halting Problem of Turing Machine

Write a short note on Halting Problem.

Prove $A_{TM} = \{(M, w) \mid \text{The Turing machine } M \text{ halts on input } w\}$ is undecidable.

- In this section we introduce one more technique to prove the undecidability of halting problem in Turing machine. Basically it asks a question "is it possible to tell whether a given machine will halt for some given input?".
- There are two possibilities of Turing Machine configuration.
 1. After finite number of states Turing Machine M will halt.
 2. The Turing Machine M will never reach to a halt state, despite its long runs.
- The halting problem is undecidable.

Suppose Turing Machine, H which decides whether computation of Turing Machine M will ever halt or given description d_M of M and the string w of M or not.

Then for every input (w, d_M) to H.

1. If M halts for input w, H is in accept state
2. If M does not halt for input w is in reject state.

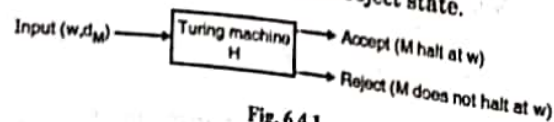


Fig. 6.4.1



- Now consider Turing Machine D, which takes d_M as the input and proceeds as follows. Copy the input d_M duplicate d_M on its tape give input to fit with one modification, whenever H is in accept halt, D will loop forever. D calls to run on M.

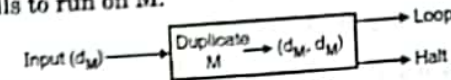


Fig. 6.4.2

For input $w = d_M$

1. Turing Machine B loops if M halts
2. Turing Machine B halts if M does not halt

This can be described as,

$$D(M) = \begin{cases} \text{Accept if } M \text{ does not accept } (M) \\ \text{Reject (If } M \text{ accepts } M) \end{cases}$$

Since D itself behave such a M then replace D by M. Fig. 6.4.3 we get,

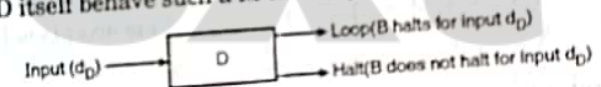


Fig. 6.4.3

Thus, D halts for input d_D if and only if D does not halt for input d_D . This is a contradiction. Hence machine H can decide that any other Turing Machine Halt does not exist.

Theorem : $A_{TM} = \{(M, w) \mid \text{The Turing machine } M \text{ halts on input } w\}$ is undecidable.

Proof : A_{TM} is decidable and get contradiction.

Let, H be the algorithm or Turing Machine such that,

$$T(H) = A_{TM}$$

$$\text{Let, } H(M, w) = \begin{cases} \text{Accept if } M \text{ accepts } w \\ \text{Reject if } M \text{ does not accept } w \end{cases}$$