THEORY OF COMPUTATION

UNIT - 5

K CHAIRMADURAI ME,.. AP

Department of Computer Science and Engineering

Adhiparasakthi Engineering College , Melmaruvathur

Unit V Undecidability

Syllabus

- Unsolvable Problems and Computable Functions
- Post Correspondence Problem (PCP)
  - -Modified PCP (MPCP)
- Recursive and recursively enumerable languages
  - – Properties
- Universal Turing machine
- Tractable and Intractable problems
- P and NP completeness
  - – Kruskal's algorithm
  - – Travelling Salesman Problem
  - - 3-CNF SAT problems.

Introduction

- The problems solved by the turing machine into two classes.
  - those that have an algorithm
    - turing machine halts whether or not it accepts its input.
  - those that are only solved by turing machines.
    - that may run forever on inputs that do not accept.

Languages in TM

Recursive languages

Recursively Enumerable languages

Non Recursively Enumerable languages

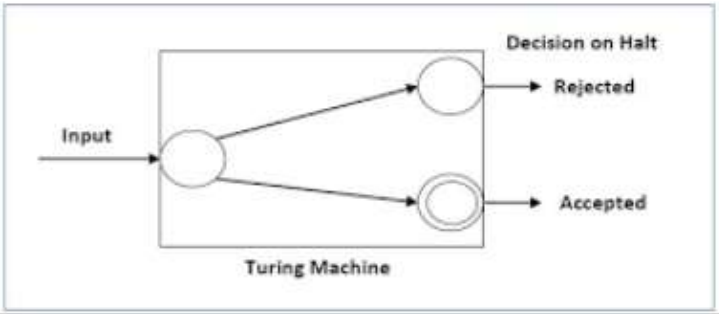
Recursive language

- A language ‘L’ is said to be recursive if there exists a Turing machine which will **accept all the strings** in ‘L’ and **reject all the strings** not in ‘L’.
- The Turing machine will halt every time and give an answer(**accepted or rejected)** for each and every string input.
- A language ‘L’ is decidable if it is a recursive language.
- All decidable languages are recursive languages and vice-versa.

For any string w:

$w \in L \Rightarrow M$ halts in a final state

$w \notin L \Rightarrow M$ halts in a non-final state



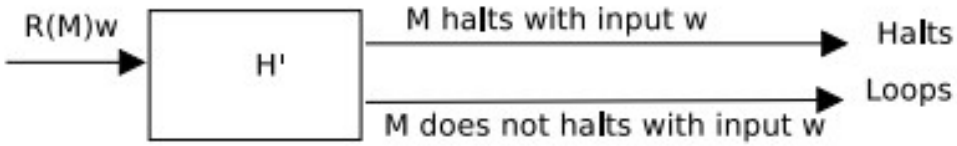
Recursively Enumerable language(RE)

- A language 'L' is said to be a recursively enumerable language if there exists a Turing machine which will accept (and therefore halt) for all the input strings which are in 'L' but **may or may not halt for all input strings which are not in 'L'.**
- By definition , all Recursive languages are also RE languages but not all RE languages are Recursive languages.

For any string w:

$w \in L \Rightarrow M$ halts in a final state

$w \notin L \Rightarrow M$ halts in a non-final state **or** loops forever



Non- Recursively Enumerable

There are languages which **does not have a Turing machine** at all

Decidable Problem

- A problem is said to be Decidable if we can always **construct a corresponding algorithm** that can answer the problem correctly.

- Suppose we are asked to compute all the prime numbers in the range of 1000 to 2000. To find the solution of this problem, we can easily devise an algorithm that can enumerate all the prime numbers in this range.

Decidable Problem

- Decidability in terms of a Turing machine, a problem is said to be a Decidable problem if there exists a **corresponding Turing machine which halts on every input with an answer- yes or no.**
- It is also important to know that these problems are termed as Turing Decidable since a Turing machine always **halts on every input, accepting or rejecting it.**

Undecidable problem

Problems that has **no algorithm** to determine the answer for a given input or which have an algorithm that answers for some input

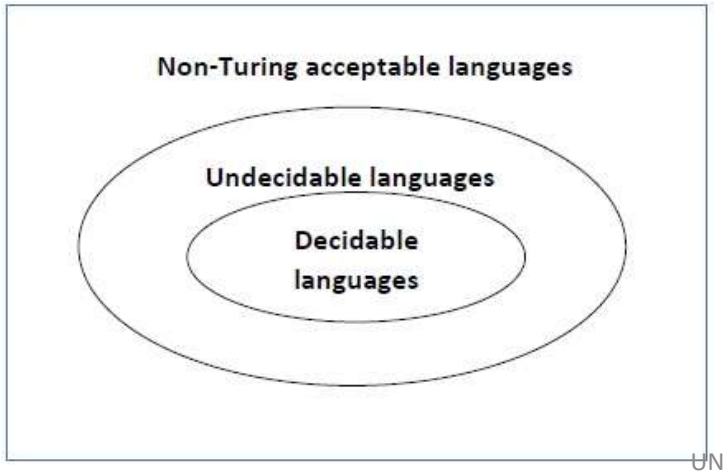
- Semi-Decidable problems are those for which a Turing machine halts on the input **accepted** by it but it can either **halt or loop forever** on the input which is rejected by the Turing Machine. Such problems are termed as Turing Recognisable problems.

Undecidable problem

- The problems for which we **can't construct an algorithm** that can answer the problem correctly in finite time are termed as Undecidable Problems. These problems may be partially decidable but they will never be decidable. That is there will always be a condition that will lead the Turing Machine into an **infinite loop** without providing an answer at all.
  - Whether a CFG generates all the strings or not?
  - Ambiguity of CFG?
  - Membership problem of a Turing Machine?
  - Finiteness of a Turing Machine?
  - Emptiness of a Turing Machine?

Languages in TM

1. Recursive ---------- Decidable
2. Recursively Enumerable (RE)
3. Non-RE Undecidable



Recursive - Decidable

- A language ‘L’ is said to be recursive if there exists a Turing machine which will
  - accept all the strings in ‘L’ and
  - reject all the strings not in ‘L’.

- The Turing machine **will halt** every time and give an answer**(accepted or rejected)** for each and every string input.

Recursively Enumerable-Undecidable

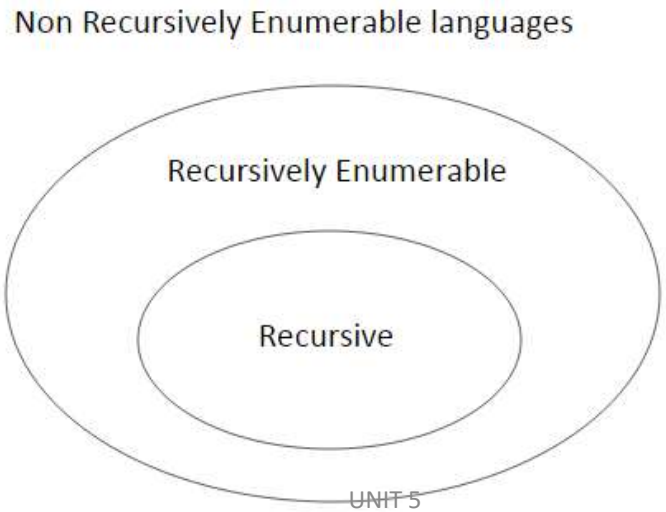
Recursively enumerable language(RE) –

A language 'L' is said to be a **recursively enumerable language** if there exists a Turing machine which will

- accept (and therefore halt) for all the input strings which are in 'L'
- but may or may not halt for all input strings which are not in 'L'.

Non-RE -Undecidable

Non- Recursively enumerable language(RE) – A language ‘L’ is said to be a non recursively enumerable language if there doesn't exists a Turing machine



Recursive, RE, Non-RE

- Recursive --------Regular, Context Free
- RE -----Lu----------Universal Language
- Non-RE -----Ld ------- Diagonalization language

A Language that is Not Recursively Enumerable

Our long-range goal is to prove undecidable the language consisting of pairs (M, w) such that:

1. M is a Turing machine (suitably coded, in binary) with input alphabet $\{0, 1\}$,
2. w is a string of 0's and 1's, and
3. M accepts input w.

Codes for Turing Machines

the Turing machines with the integers, and we can talk about "the ith Turing machine, M_i." To represent a TM $M = (Q, \{0,1\}, \Gamma, \delta, q_1, B, F)$ as a binary string, we must first assign integers to the states, tape symbols, and directions L and R.

- We shall assume the states are $q_1, q_2, \ldots, q_r$ for some r. The start state will always be q_1, and q_2 will be the only accepting state. Note that, since we may assume the TM halts whenever it enters an accepting state, there is never any need for more than one accepting state.

q1 = 0
q2 = 00
q3 = 000

- We shall assume the tape symbols are $X_1, X_2, \ldots, X_s$ for some s. X_1 always will be the symbol 0, X_2 will be 1, and X_3 will be B, the blank. However, other tape symbols can be assigned to the remaining integers arbitrarily.

x1= 0 = 0
x2 = 1= 00
x3 = B= 000

- We shall refer to direction L as D_1 and direction R as D_2.

L=D1=0
R=D2=00

Codes for Turing Machines

Once we have established an integer to represent each state, symbol, and direction, we can encode the transition function δ. Suppose one transition rule is $\delta(q_i, X_j) = (q_k, X_l, D_m)$, for some integers i, j, k, l, and m. We shall code this rule by the string $0^i10^j10^k10^l10^m$. Notice that, since all of i, j, k, l, and m are at least one, there are no occurrences of two or more consecutive 1's within the code for a single transition.

A code for the entire TM M consists of all the codes for the transitions, in some order, separated by pairs of 1's:

$$C_1 11 C_2 11 \cdots C_{n-1} 11 C_n$$

where each of the C's is the code for one transition of M.

Codes for Turing Machines

q1 = 0
q2 = 00
q3 = 000

Example 9.1: Let the TM in question be

$$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$$

where δ consists of the rules:

x1= 0 = 0
x2 = 1= 00
x3 = B= 000

L=D1=0
R=D2=00

$$\delta(q_1, 1) = (q_3, 0, R)$$
$$\delta(q_3, 0) = (q_1, 1, R)$$
$$\delta(q_3, 1) = (q_2, 0, R)$$
$$\delta(q_3, B) = (q_3, 1, L)$$

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

For example, the first rule can be written as $\delta(q_1, X_2) = (q_3, X_1, D_2)$, since $1 = X_2$, $0 = X_1$, and $R = D_2$. Thus, its code is $0^1 10^2 10^3 10^1 10^2$, as was indicated above. A code for M is:

Codes for Turing Machines

The codes for each of these rules, respectively, are:

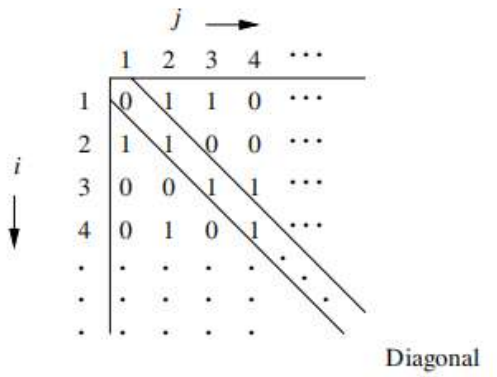
0100100010100
0001010100100
0001001001010 0
0001000100010010

0100100010100110001010100100110001001001010011000100010001001 0

The Diagonalization Language (L_d)

- The language L_d, the *diagonalization language*, is the set of strings w_i such that w_i is not in $L(M_i)$.

That is, L_d consists of all strings w such that the TM M whose code is w does not accept when given w as input.



The table that represents acceptance of strings by Turing machines

The process of compleenting the diagonal to construct the characteristic vector of a language that appears in any row is called diagonalization.

Then the complement of the diagonal cannot be the characteristic vectorof any Turing machine.

L_d is not a Recursively Enumerable Language

Theorem 9.2 : L_d is not a recursively enumerable language. That is, there is no Turing machine that accepts L_d.

PROOF: Suppose L_d were $L(M)$ for some TM M. Since L_d is a language over alphabet $\{0, 1\}$, M would be in the list of Turing machines we have constructed, since it includes all TM's with input alphabet $\{0, 1\}$. Thus, there is at least one code for M, say i; that is, $M = M_i$.

Now, ask if w_i is in L_d.

- If w_i is in L_d, then M_i accepts w_i. But then, by definition of L_d, w_i is not in L_d, because L_d contains only those w_j such that M_j does *not* accept w_j.
- Similarly, if w_i is not in L_d, then M_i does not accept w_i, Thus, by definition of L_d, w_i *is* in L_d.

Since w_i can neither be in L_d nor fail to be in L_d, we conclude that there is a contradiction of our assumption that M exists. That is, L_d is not a recursively enumerable language. □