UNIT-VI

Language decidability

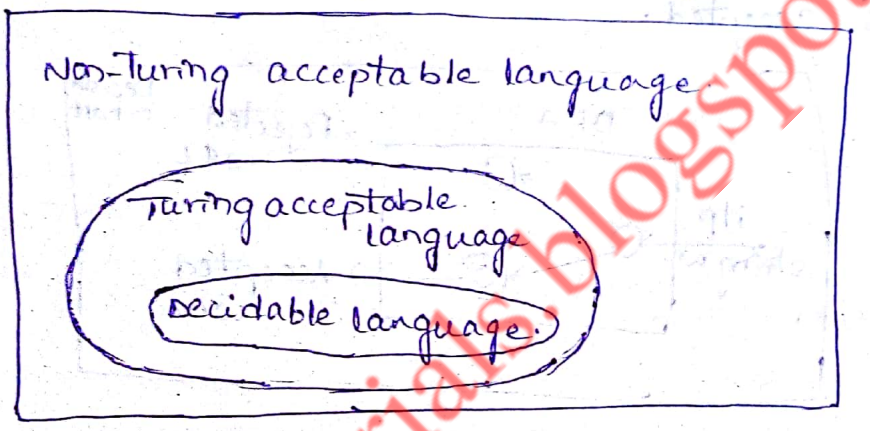
* Introduction
* Examples

Introduction:-

- **Decidable problem:-**

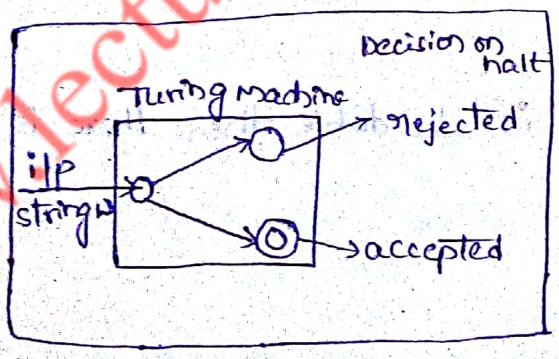
* A language is called Decidable (or) recursive if there is a turing machine which accepts and halts on every i/p string "w".

* Every decidable language is a turing acceptable.



* A decision problem 'p' is decidable if the language 'L' of all "yes" instances to 'p' is decidable.

* for a decidable language, for each i/p string, the turing machine halts either at the accept (or) the reject state



Examples:-

1) find out whether the following problem is decidable (or) not.

Is a number 'm' prime?

Sol:- Prime numbers = {2, 3, 5, 7, 11, 13, 17, 19, ----}

divide the number 'm' by all the numbers b/w

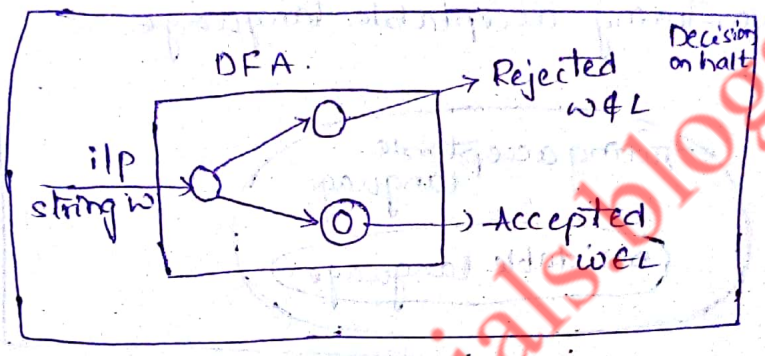
2 and $m/2$ starting from 2.

If any of these numbers produce a remainder 0. then it goes to the rejected state. otherwise it goes to the accepted state. so, here the answer could be made by YES (or) NO.

Hence, it is a decidable problem.

2) Given a Regular language 'L' and string 'w', how can we check if $w \in L$.

sol:- Take the DFA that accepts 'L' and check if 'w' is accepted.



Some more decision problems are

i) Does DFA accept the empty language.

ii) Is $L_1 \cap L_2 = \phi$ for regular sets.

iii) If a language L is decidable then its complement L' is also decidable.

iv) If a language is decidable then there is a turing machine for it.

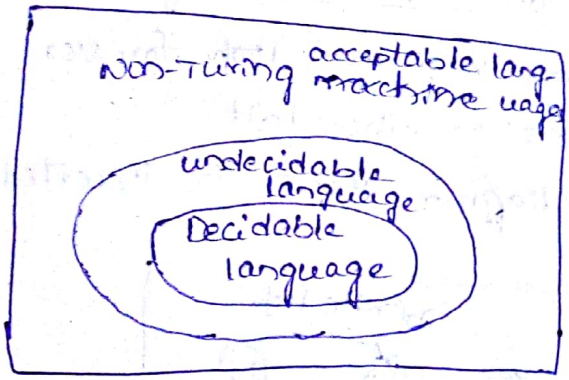
Undecidable problems:-

Introduction:-

* for an undecidable language there is no "TM" which accepts the language and makes a decision for every ilp string 'w'.

* A decision problem 'p' is called undecidable if the language 'L' of all 'YES' instances to 'p' is not decidable.

undecidable languages are not recursive languages but, some times they may be recursive enumerable languages.



Examples :—

i) The halting problem of turing machine.

ii) The mortality problem.

iii) The mortal Matrix problem.

iv) The post Correspondence problem [pcp]

i) The halting problem :

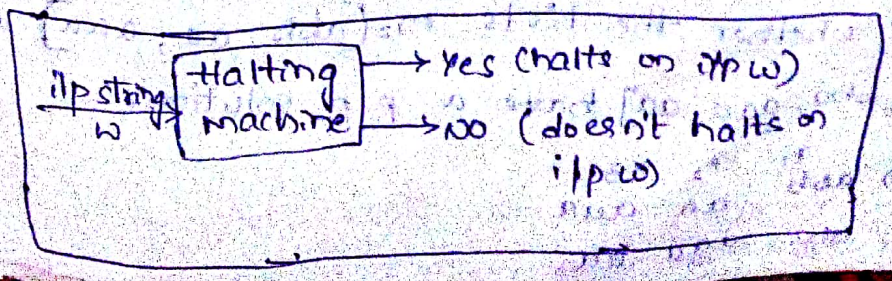
The halting problem i/p : a turing machine and the i/p string w.

problem : Does the turing machine finish computing of the string 'w' in a finite no. of steps ? The answer must be either yes (or) NO.

Proof :— At first, we will assume that a turing machine exists to solve, the problem. we will show and then it is contradicting it self.

we will call this 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 then the o/p comes as YES. otherwise, as NO

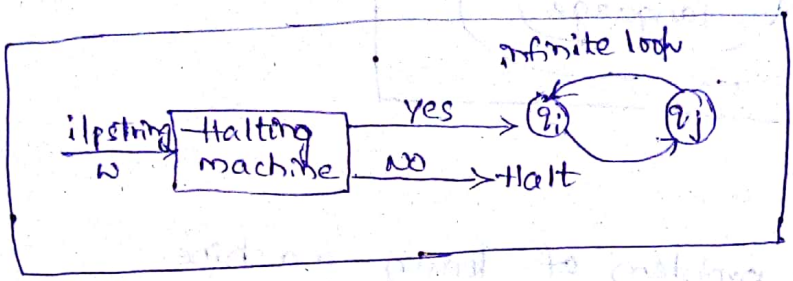


Now, we will design an inverted halting machine as.

i) If HM returns YES then loop forever.

ii) If HM returns NO then halt.

The following block diagram shows the inverted halting machine.



further, a machine "HM" which ilp itself is constructed as follows.

i) If HM halts on ilp loop forever.

ii) Else Halt

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

**Post correspondence problem (pcp):—

- It was introduced by "Emil post" in 1946 is as undecidable decision problem.

The pcp problem over an ilp alphabet 'Σ' is stated as follows.

Given the following two lists, M, and N. of non-empty strings over Σ

$$M=(x_1, x_2, x_3 \cdots x_n)$$

$$N=(y_1, y_2, y_3 \cdots y_n)$$

we can say that there is a pcp solution if for some $i_1, i_2, i_3 \cdots i_K$ where $1 \le i_K \le n$. The condition.

$$x_{i_1} x_{i_2} x_{i_3} \cdots x_{iK} = y_{i_1} y_{i_2} y_{i_3} \cdots y_{iK}$$ satisfies

Ex:- 1) find whether the lists $M=[abb, aa, aaa]$ and $N=[bba, aaa, aa]$ have a pcp solution.

Sol:-

	x_1	x_2	x_3
M	abb	aa	aaa
N	bba	aaa	aa

Here $x_2 x_1 x_3 = aaabbbaaa$

$y_2 y_1 y_3 = aaabbbaaa$

we can see that $x_2 x_1 x_3 = y_2 y_1 y_3$.

Hence, the solution is $i=2, j=1, K=3$.

2) find whether the list $M = [ab, bab, bbaaa]$ and $N = [a, ba, bab]$ have a pcp solution.

Sol:-

	x_1	x_2	x_3
M	ab	bab	bbaaa
N	a	ba	bab

In this case, there is no solution. because

$|x_2 x_1 x_3| \neq |y_2 y_1 y_3|$ length's are not same.

Hence, it can be said that this pcp is a undecidable problem.

Modified post correspondence problem:-

Given two lists $M = x_1, x_2, x_3 \cdots x_n$ and $N = y_1, y_2, y_3 \cdots y_n$

Given a set of pairs of strings $\{(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)\}$

then the solution is an instance such that,

$$x_1 x_{i_1} x_{i_2} \cdots x_{i_n} = y_1 y_{i_1} y_{i_2} \cdots y_{i_n}$$

that means the pair (x_1, y_1) is forced to be at the begining of the strings.

Ex:-

	x_1	x_2	x_3
M	11	100	111
N	111	001	11

Sol:- ∴Then the solution is $x_1 x_2 x_3 = y_1 y_2 y_3$.

$x_1 x_2 x_3 = 11100111$

$y_1 y_2 y_3 = 11100111$

That means it is essential to have x_1 & y_1 at the beginning of list.

P and Np Classes :—

* P-problems
* Np-problems
* P vs Np

- **P-problems:—**

* P is the class of problems that can be solved by deterministic algorithm in a polynomial type time $p(n^k)$ where 'n' is the size of i/p string.
* P-problem consist of a language accepted by deterministic Turing machine that runs in polynomial amount of time.

Ex:— 1) shortest path problem
2) Equivalence of NFA and DFA.
3) shortest cycle in a graph.
4) sorting algorithms

- **Np-problems:—**

* Np-problem is a class of problems that can be solved by Non-deterministic algorithms in a polynomial time $p(n^k)$ where 'n' is the size of i/p string.
* Np-problems consists of a language accepted by Non-deterministic turing machine that runs in a polynomial amount of time.

Ex:— 1) Travelling sales man problem.
2) subgraph isomorphism

Np-problem classified into two types:

i) Np-hard problem.
ii) Np- complete problem.

- **Np-hard problem:—**

If there is a language x such that every language y in Np can be polynomially reduceable to x. and we cannot prove that x is in Np. then x is said to be Np-hard problem.

Ex: Turing machine halting problem.

- **Np- Complete problem:—**

If there is a language x such that every language

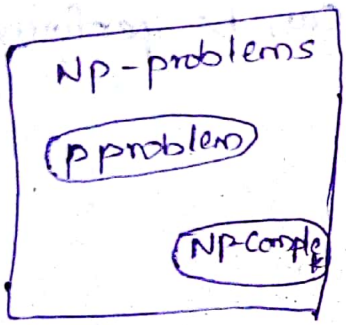
y in NP can be polynomially reducable to x and we can prove that x is in NP then x is said to be NP-complete problems.

Ex:- 1) Travelling salesman problem.
2) Subgraph isomorphism.

P VS NP :-

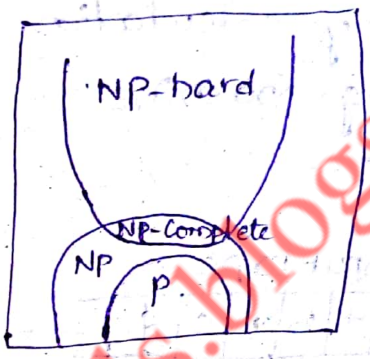
1. Ladner's theorem:

(a) $P \neq NP$



2. Euler's theorem:-

(a) $P \neq NP$



(b) $P = NP$