Nadar Saraswathi College of Engineering & Technology

Engineering Excellence for Empowerment

NSCET E-LEARNING PRESENTATION

LISTEN … LEARN… LEAD…

COMPUTER SCIENCE AND ENGINEERING

III YEAR / V SEMESTER

CS8501 – Theory of Computation

P.MAHALAKSHMI,M.E,MISTE

ASSISTANT PROFESSOR

Nadar Saraswathi College of Engineering & Technology,

Vadapudupatti, Annanji (po), Theni – 625531.

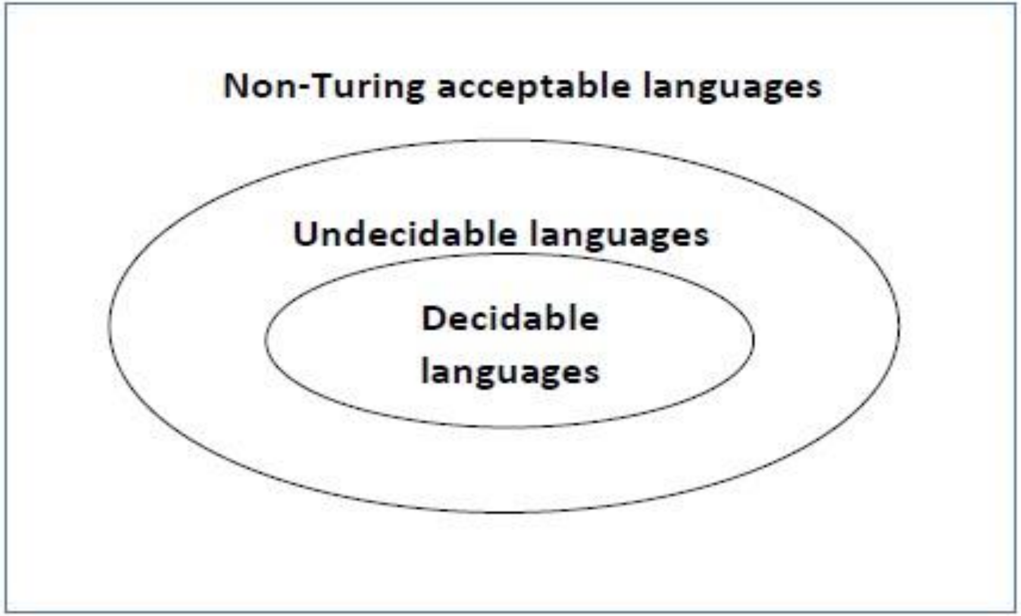
UNIT V

Undecidability

Non Recursive Enumerable Language

Introduction

For an undecidable language, there is no Turing Machine which accepts the language and makes a decision for every input string **w** (TM can make decision for some input string though). 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 sometimes, they may be recursively enumerable languages.



Example

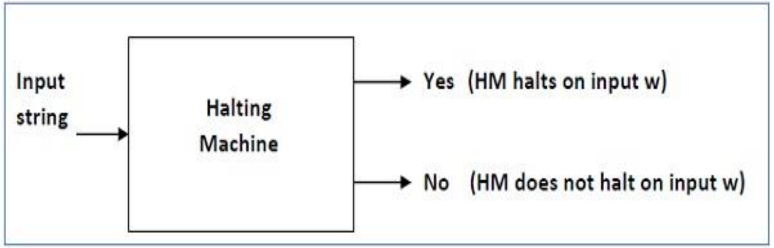
- ✓ The halting problem of Turing machine
- ✓ The mortality problem
- ✓ The mortal matrix problem
- ✓ The Post correspondence problem, etc.

Halting Problem

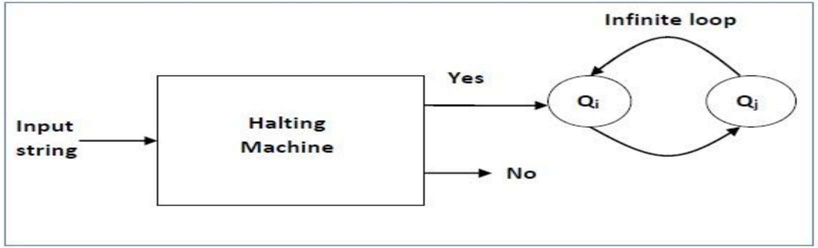
Problem – Does the Turing machine finish computing of the string **w** in a finite number of steps? The answer must be either yes or no.

Input – A Turing machine and an input string **w**.

Proof – At first, we will assume that such a Turing machine exists to solve this problem and then we will show it is contradicting itself. 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, 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.
- ✓ The following is the block diagram of an 'Inverted halting machine' –



- ✓ Further, a machine $\mathbf{(HM)_2}$ which input itself is constructed as follows –
- ✓ If $(HM)_2$ halts on input, loop forever. Else, halt.
- ✓ Here, we have got a contradiction. Hence, the halting problem is **undecidable**.

Topic

Decidable problems for recursively enumerable language

Rice Theorem

Rice theorem states that any non-trivial semantic property of a language which is recognized by a Turing machine is undecidable. A property, P, is the language of all Turing machines that satisfy that property.

Formal Definition

If P is a non-trivial property, and the language holding the property, L_p , is recognized by Turing machine M, then $L_p = \{<M> \mid L(M) \in P\}$ is undecidable.

Description and Properties

Property of languages, P, is simply a set of languages. If any language belongs to P ($L \in P$), it is said that L satisfies the property P.

A property is called to be trivial if either it is not satisfied by any recursively enumerable languages, or if it is satisfied by all recursively enumerable languages.

A non-trivial property is satisfied by some recursively enumerable languages and are not satisfied by others. Formally speaking, in a non-trivial property, where $L \in P$, both the following properties hold:

Property 1 – There exists Turing Machines, M1 and M2 that recognize the same language, i.e. either ($<M1>, <M2> \in L$) or ($<M1>,<M2> \notin L$)

Property 2 – There exists Turing Machines M1 and M2, where M1 recognizes the language while M2 does not, i.e. $<M1> \in L$ and $<M2> \notin L$

Proof

Suppose, a property P is non-trivial and $\varphi \in P$.

Since, P is non-trivial, at least one language satisfies P, i.e., $L(M_0) \in P$, $\ni$ Turing Machine M_0.

Let, w be an input in a particular instant and N is a Turing Machine which follows –

On input x

Run M on w

If M does not accept (or doesn't halt), then do not accept x (or do not halt)

If M accepts w then run M_0 on x. If M_0 accepts x, then accept x.

A function that maps an instance ATM = {<M,w>| M accepts input w} to a N such that

If M accepts w and N accepts the same language as M_0, Then $L(M) = L(M_0) \in p$

If M does not accept w and N accepts φ, Then $L(N) = \varphi \notin p$

Since A_{TM} is undecidable and it can be reduced to Lp, Lp is also undecidable.

Topic

Post Correspondence Problem

Introduction

The Post Correspondence Problem (PCP), introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet ∑ is stated as follows − Given the following two lists, **M** and **N** of non-empty strings over ∑ −

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

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

We can say that there is a Post Correspondence Solution, if for some $i_1, i_2, \ldots\ldots\ldots\ldots i_k$, where $1 \leq i_j \leq n$, the condition $x_{i1} \ldots\ldots x_{ik} = y_{i1} \ldots\ldots y_{ik}$ satisfies.

Example 1

Find whether the lists

M = (abb, aa, aaa) and N = (bba, aaa, aa)

have a Post Correspondence Solution?

Solution

x_1	x_2	x_3	
M	Abb	aa	aaa
N	Bba	aaa	aa

Here,

$x_2x_1x_3$ **= 'aaabbaaa'**

and $y_2y_1y_3$ **= 'aaabbaaa'**

We can see that

$x_2x_1x_3 = y_2y_1y_3$

Hence, theExample 2

Find whether the lists **M = (ab, bab, bbaaa)** and **N = (a, ba, bab)** have a Post Correspondence Solution?

solution is **i = 2, j = 1, and k = 3.**

Solution

x1	x2	x3	
M	ab	bab	bbaaa
N	a	ba	bab

In this case, there is no solution because –

$| x_2x_1x_3 | \neq | y_2y_1y_3 |$ (Lengths are not same)

Hence, it can be said that this Post Correspondence Problem is **undecidable**.

Topic

Class P and NP Problems

P-Class

- The class P consists of those problems that are solvable in polynomial time, i.e. these problems can be solved in time $O(n^k)$ in worst-case, where **k** is constant.
- These problems are called **tractable**, while others are called **intractable or super polynomial**.
- Formally, an algorithm is polynomial time algorithm, if there exists a polynomial $p(n)$ such that the algorithm can solve any instance of size **n** in a time $O(p(n))$.
- Problem requiring $\Omega(n^{50})$ time to solve are essentially intractable for large n. Most known polynomial time algorithm run in time $O(n^k)$ for fairly low value of k.
- The advantages in considering the class of polynomial-time algorithms is that all reasonable **deterministic single processor model of computation** can be simulated on each other with at most a polynomial slow-d

NP-Class

- The class NP consists of those problems that are verifiable in polynomial time. NP is the class of decision problems for which it is easy to check the correctness of a claimed answer, with the aid of a little extra information. Hence, we aren't asking for a way to find a solution, but only to verify that an alleged solution really is correct.
- Every problem in this class can be solved in exponential time using exhaustive search.

P versus NP

- ✓ Every decision problem that is solvable by a deterministic polynomial time algorithm is also solvable by a polynomial time non-deterministic algorithm.
- ✓ All problems in P can be solved with polynomial time algorithms, whereas all problems in *NP - P* are intractable.
- ✓ It is not known whether ***P = NP***. However, many problems are known in NP with the property that if they belong to P, then it can be proved that P = NP.
- ✓ If ***P ≠ NP***, there are problems in NP that are neither in P nor in NP-Complete.
- ✓ The problem belongs to class **P** if it's easy to find a solution for the problem. The problem belongs to **NP**, if it's easy to check a solution that may have been very tedious to find

NP Hard and NP-Complete Classes

A problem is in the class NPC if it is in NP and is as **hard** as any problem in NP. A problem is **NP-hard** if all problems in NP are polynomial time reducible to it, even though it may not be in NP itself.