D Y PATIL DEEMED TO BE UNIVERSITY
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Theoretical Computer Science

Unit 6: Undecidability

Faculty Name : Ms. Namita Pulgam

Lecture No:

Undecidability and Recursively Enumerable Language

Recursively Enumerable Languages

- The language that is accepted by Turing Machine is called as **Recursively enumerable language**.
- It is also called as **Turing Acceptable language**.
- A language **L € ∑*** is said to be Turing acceptable if there is a Turing Machine which **halts on every w € L** with answer YES,

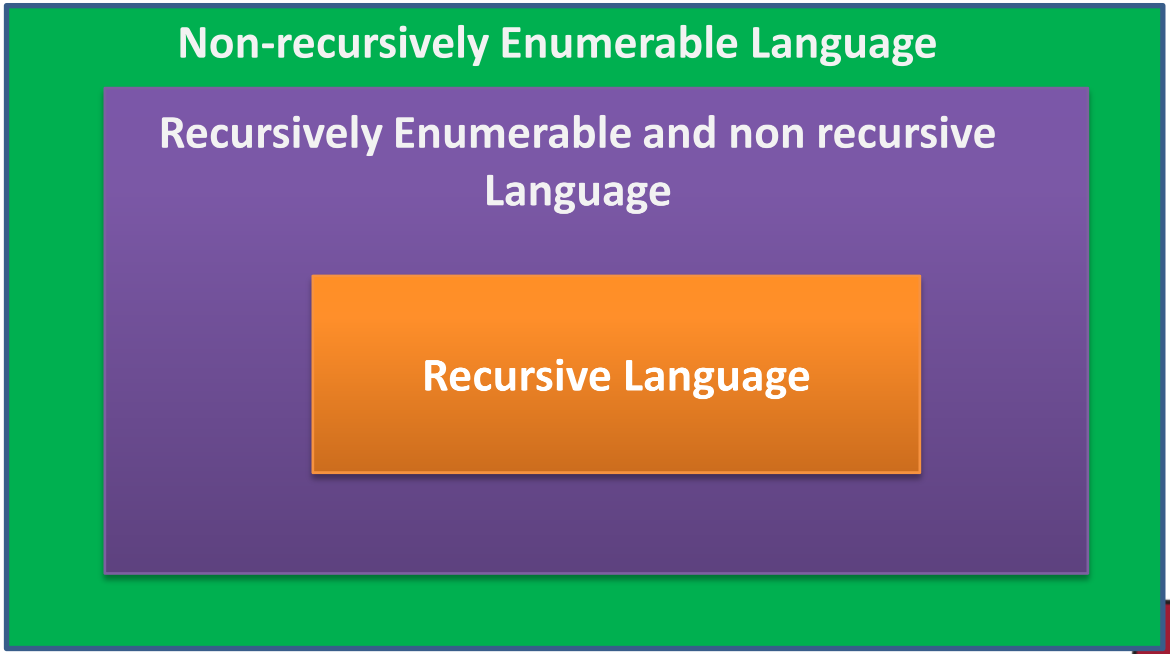
  but if w ~~€~~ L then it may not halt.

Recursive and Recursively Enumerable Languages

- Recursive language is also called as **Turing decidable**
- A language L € $\sum$* is said to be Recursive language or Turing decidable if there is a Turing machine which **always halts on every w € $\sum$*.**
- **If w € L then it halts with answer YES and if w ∉ L then it halts with answer NO.**

Relationship between Recursive and Recursively Enumerable Language

- Each Turing Decidable language is Turing Acceptable.
- Each Turing Acceptable language is need not to be Turing Decidable.
- There are some languages which are neither Turing Acceptable nor Turing Decidable



Difference between Recursive and Recursively Enumerable Language

SNo.	Recursive Language	Recursively Enumerable Language
(1)	A language L is said to be recursive language if there is a Turing machine which always halts on every $w \in Z^*$. If $w \in L$ then it halts with answer YES and if $w \notin L$ then it halts with answer NO.	A language L is said to be recursive enumerable language if there is a Turing machine which halts on every $w \in L$ with answer YES but if $w \notin L$ then it may not halts.
(2)	It is a Turing decidable language.	It is a Turing acceptable language.
(3)	Each Turing decidable language is Turing acceptable.	Each Turing acceptable language need not be Turing decidable.

Undeciadability

- There are some problems which no computer can solve.
- We can have approximate solutions for the problems which cannot be solved by computational means.
- A class of problems is said to be **decidable** if there exists some definite algorithm which always terminates with the correct answer.
- **We can also say that the problem is decidable if there exists a Turing machine which gives correct answer for every statement in the domain of the problem.**

Undeciadability

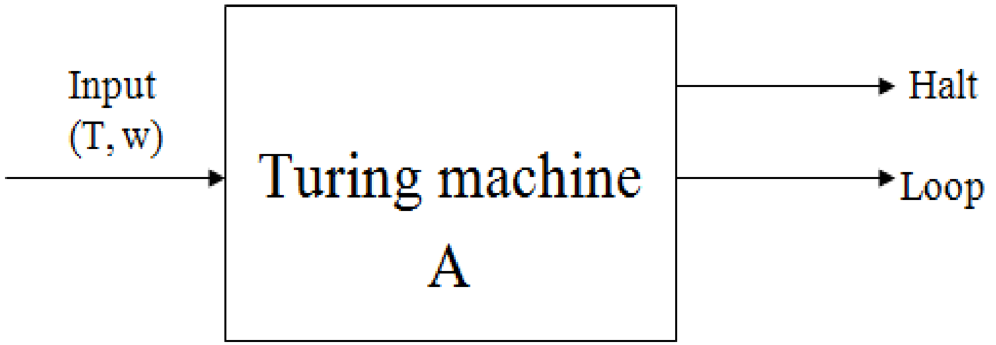
- The class of problems for which there is no Turing Machine that gives correct answer for every input instance said to be **undecidable problems.**
- These problems are also called as unsolvable problems.
- Some examples of undecidable problems are:
  - Halting Problem
  - Post Correspondence problem

Halting Problem

- **The problem of determining whether a given Turing machine M with the input ‘w’ will ever halt or not is called as Halting problem.**
- The Halting problem is undecidable.
- Because, in reality, there is no Turing machine which takes any other Turing machine as input and decides whether it halts or not.

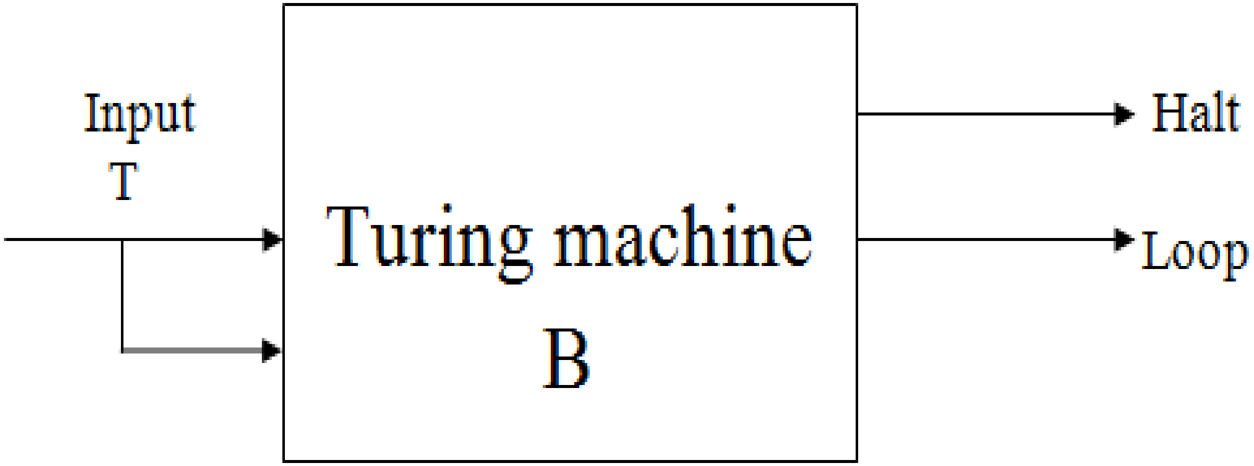
Halting Problem Proof

- **Proof by Contradiction**
- We will assume that there exists a Turing machine which solves the Halting problem.
- Consider a **Turing machine A**, which takes configuration of any **other Turing machine T** and the **input string ‘w’** as input. So, the Turing machine A can determine whether the Turing machine T will ever halt or not for a given input string ‘w’.



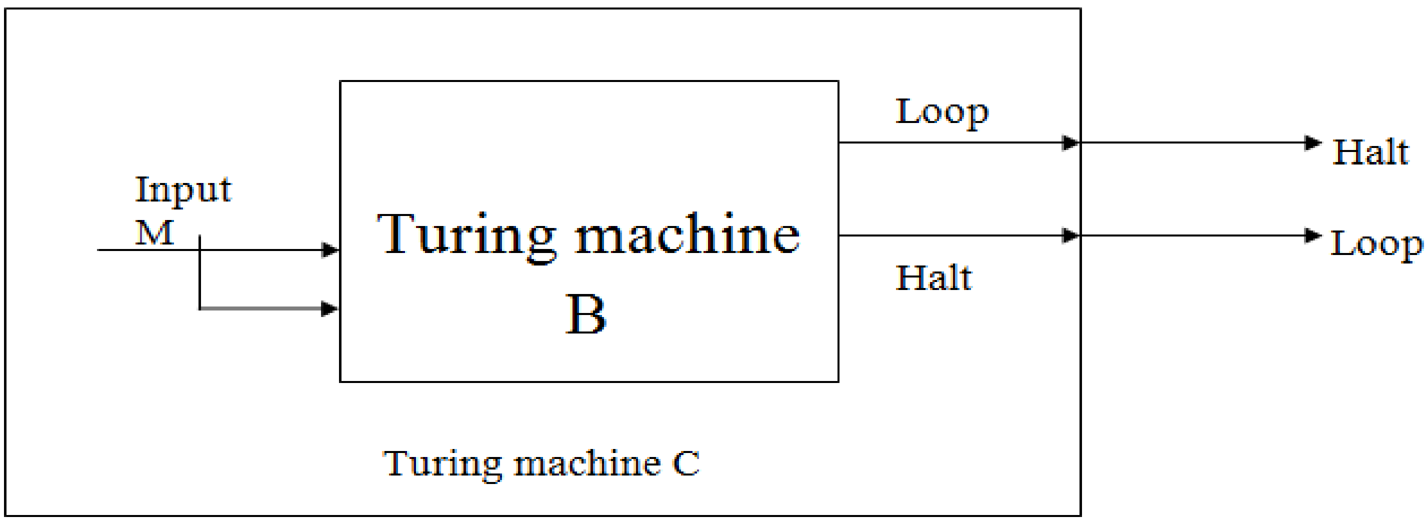
Halting Problem Proof continued..

- Now, we will construct another **Turing machine B** with both the input as T. Now B will decide whether the Turing machine T will ever halt or not.



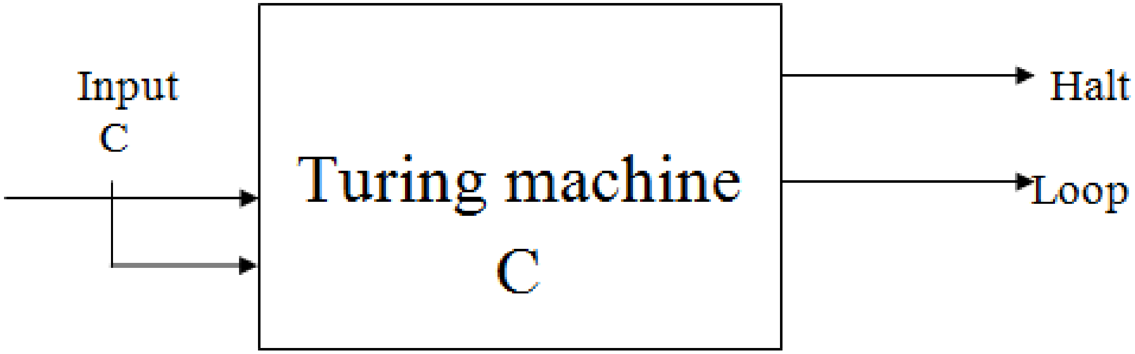
Halting Problem Proof continued..

- Now we will construct another **Turing machine C which takes output of B as input** and does opposite job of machine B.
- That means if the **Turing machine B halts then C will loop** and if **Turing machine B loops then C will halt.**



Halting Problem Proof continued..

- Let us give **Turing machine C itself as input to C.**
- Now for this configuration the **Turing machine C will halt with input C if and only if machine C loops** and **C will loop with its input if and only if C halts.**
- In both of the cases the result is wrong. This is the contradiction.
- Hence we can conclude that machine C does not exists and so the machines A and B do not have existence. **Thus, the Halting problem is unsolvable.**



Post Correspondence Problem (PCP)

- Consider an alphabet $\sum$. Let A and B are two sequences of non-empty strings with same length over $\sum$.
- The sequences A and B are represented as:

$$A = (a_1, a_2, a_3, \ldots\ldots\ldots, a_n)$$

$$B = (b_1, b_2, b_3, \ldots\ldots\ldots b_n)$$

Where each **a_i and b_i is a nonempty string in A and B** respectively.

Post Correspondence Problem (PCP)

- The post correspondence problem is to determine if there exists a sequence of one or more integers such that

$$a_i\ a_j\ a_k\ \ldots\ldots a_m = b_i\ b_j\ b_k \ldots\ldots b_m$$

- Where each of these integers i, j, k ….m is greater than or equal to ‘1’ and less than or equal to n (‘n’ is the length of A and B).

- **The sequence (i, j, k,….m) is called as solution to the post correspondence problem.**

- The **PCP is unsolvable** since there is no algorithm which can determine such sequence for the given lists.

PCP Example 1:

Does the PCP with two lists

A = {a, abaaa, ab}

B = {aaa, ab, b} have a solution?

Solution:

- We have to find such sequence using which if we list out the elements of A and B then it will generate same strings.
- **Consider the sequence (2, 1, 1, 3)**

$$A_2 \; A_1 \; A_1 \; A_3 = \text{abaaaaaab}$$

$$B_2 \; B_1 \; B_1 \; B_3 = \text{abaaaaaab}$$

Thus, $A_2 \; A_1 \; A_1 \; A_3 = B_2 \; B_1 \; B_1 \; B_3$

Thus the PCP has the solution. The solution is sequence (2, 1, 1, 3)

PCP Example 2:

Determine the solution for the following instance of the PCP

A = {01, 110010, 1, 11}

B = {0,0, 1111,01}

Solution:

- We have to find such sequence using which if we list out the elements of A and B then it will generate same strings.
- **Consider the sequence (1, 3, 2, 4, 4, 3)**

  $A_1\ A_3\ A_2\ A_4\ A_4\ A_3 = 01111001011111$

  $B_1\ B_3\ B_2\ B_4\ B_4\ B_3 = 01111001011111$

- Thus, $A_1\ A_3\ A_2\ A_4\ A_4\ A_3 = B_1\ B_3\ B_2\ B_4\ B_4\ B_3$
- **Thus the PCP has the solution. The solution is sequence (1, 3, 2, 4, 4, 3)**

PCP Example 3:

Does the PCP with two lists

A = {10, 011, 101}

B = {101, 11, 011} have a solution? Justify your answer.

Solution:

- It can be observed from the two lists that the sequences A2 (011) and B2 (11) start with different symbol. Similarly sequences A3 (101) and B3 (011) differ in first place.
- This PCP instance has no solution.

PCP Example 4:

Does the PCP with two lists

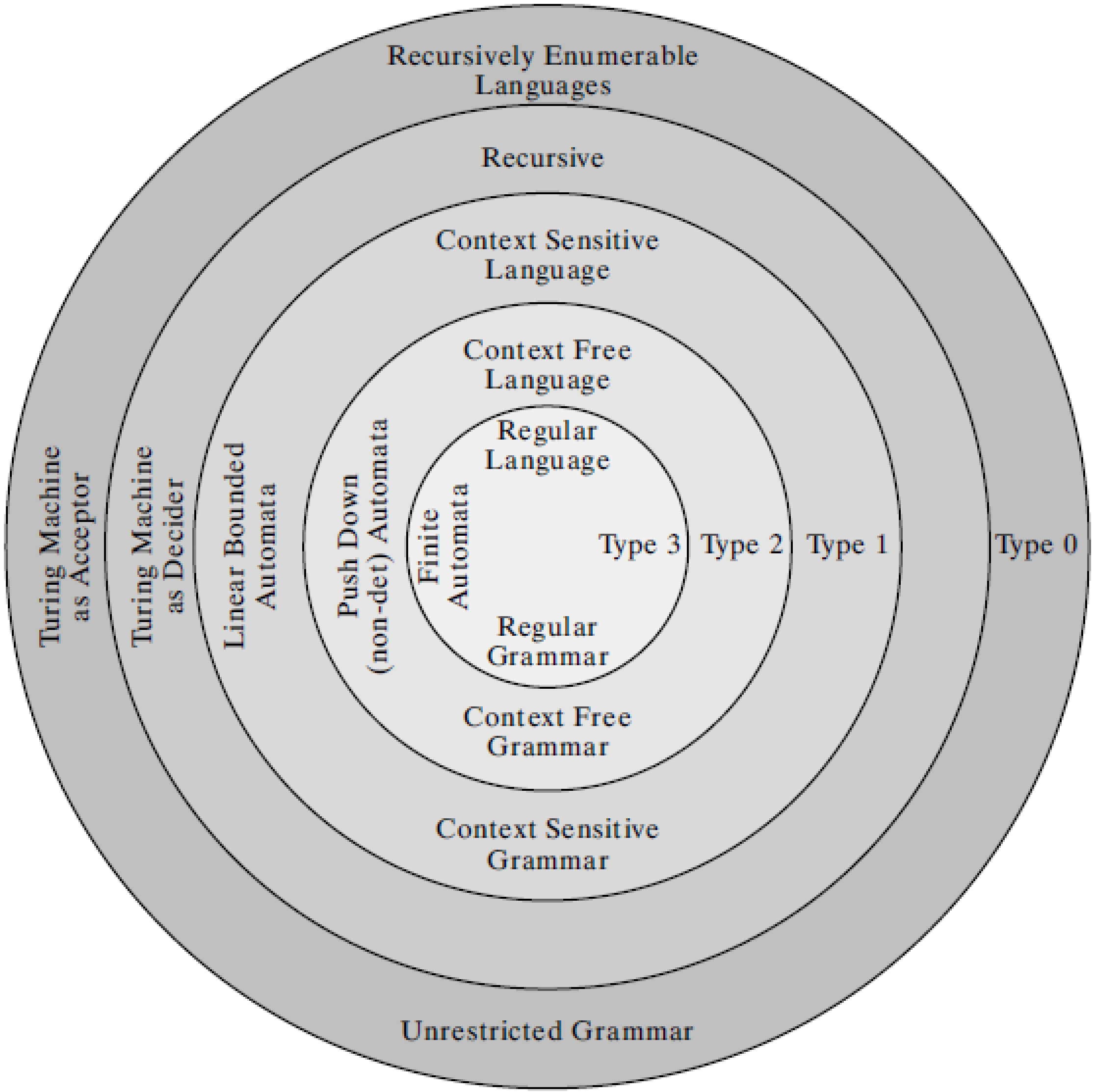
A = {b, babbb, ba}

B = {bbb, ba, a} have a solution?

Solution:

- We have to find such sequence using which if we list out the elements of A and B then it will generate same strings.
- **Consider the sequence (2, 1, 1, 3)**
- $A_2\ A_1\ A_1\ A_3$ = babbbbbba
- $B_2\ B_1\ B_1\ B_3$ = babbbbbba
- Thus, $A_2\ A_1\ A_1\ A_3 = B_2\ B_1\ B_1\ B_3$
- **Thus the PCP has the solution. The solution is sequence (2, 1, 1, 3)**

CHOMSKY HIERARCHY REVISED



Rice Theorem

- The rice's theorem states **any non-trivial property of recursively enumerable languages is undecidable.**
- The property which is true for all elements of set or which is false for all elements of set is called as **trivial property.**
- The property which is true for some recursively enumerable languages and which is false for other recursively enumerable languages is called as **non-trivial property.**

Thank You