Theory of Computation

(A study of the computational devices/ machines)

– their power/ limitation with mathematical proof

Problem: There is a finite set of input and there is a finite set of desired output

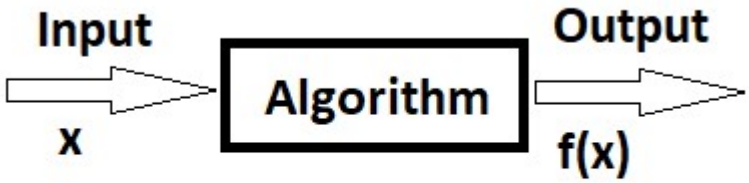
Computation: A "Program" that runs on a machine and carryout a computation

Program: "Algorithm" expressed through a programming language

Algorithm:

- is a recipe of carrying out input & output transformation?
- "finite ordered description of a process to carry out the transformation of input to output"
- it tells "how to compute a function"

Function: is a mapping of D (Domain) to some R (Range)



f: D -> R

- in this case 'Domain' is all input and 'Range' is all possible output

Example: is_prime.Number -> { 'yes', 'no'}

$$is\_prime(n) = \begin{cases} 'yes', & if\ 'n'\ is\ prime \\ 'no', & if\ 'n'\ is\ not\ prime \end{cases}$$

So what we have understand:

- Every ‘Algorithm’ computes a/ some ‘Function(s)’
- ‘Function’ definition does not illustrate the way to achieve the right answer
- So we need an “Algorithm” to have the right results

Some situation may lead:

- for function ‘f’ there is an algorithm ‘A’ to compute ‘f’
- Although having ‘Function’ no ‘Algorithm’ defines how to achieve that state

Question:

- For what ‘Function’ we have ‘Algorithm’?
- to identify the class of ‘Functions’ which admit ‘Algorithm’ to compute them

Set Membership Problem:

Suppose S is a set
Given any ‘a’ to identify a ϵ S

- Assume we have a function ‘f’ then the $graph(f) = \{ (a,b) \mid f(a) = b\}$

- Suppose we found there is no algorithm to solve the ‘Set Membership Problem’ for the set graph(f), then there is no algorithm to computer function ‘f’

Formal Language:

A simple way (language) to present the states of the machine.

Course Outline:

- **Regular Language**
  A Regular Language is a language that can be expressed with a regular expression or a deterministic or non-deterministic finite automata or state machine.
  - Finite Automata
    - Simple machine, no memory
    - It uses in Model Checking, design large electronics circuit
    - Protocol design, Text Processing Application, Important component of the Compiler
  - Nondeterministic Finite Automata
  - Regular Expression
    - String pattern used in many systems
    - Unix command parser: a.*b
    - Document Type Definition (DTD) like XML:
      person(name, addr, child*)
  - Properties of Regular Languages
  - Closure properties of Regular Languages

- **Context Free Language**
  - Context Free Grammar (CFG)
    - Typically, the tree structure to parse/ analyze the text stream
    - Syntax of the Programming Language according to some recursive rules
  - Push down automata
    - The machine model of CFG
  - Decision and closure properties of CFL

- **Recursive and recursively enumerable Language**
  - Fundamental limitation our ability to compute
    - Undecidable - No program/ algorithm
    - Intractable - Solution takes exponential time with respect to input size but no proof
  - Turing Machine
    - What we can compute

- **Intractable Problem**
  - Problems that (appears to) requires exponential amount of time w.r.t. its input size

Text Book:

- Introduction to Automata Theory, Languages, and Computation by Johne Hopcroft, Rajee Motwani, Jeffret D. Ullman
- Introduction to the Theory of Computation 3rd Edition by Michael Sipser