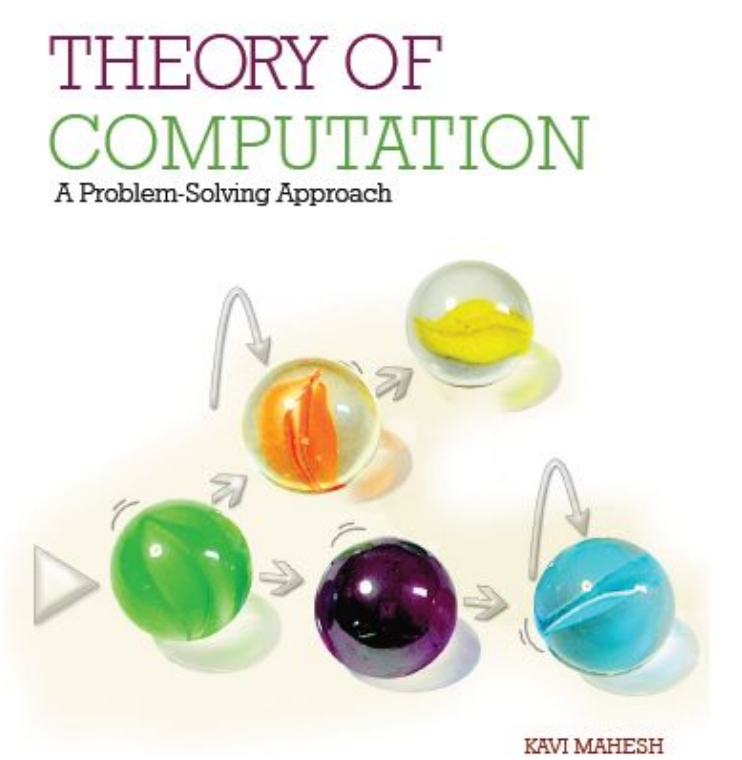


Theory of Computation: A Problem-Solving Approach

Chapter – 1: Computers and the Science of Computing

Dr. Kavi Mahesh

Learning Objectives

- Distinguish between computing machines and other kinds of machines
- Learn what it means to compute
- Learn relationships between computing machines, problems, languages and grammars.
- Learn to define computer science.
- Appreciate the relationships between the science of computing and a core set of subject areas in computer science.
- Appreciate the historical development of the science of computing.
- Get an introduction to computability, intractability and intelligence.
- Understand the reasons for studying the science of computing.

What is (not) a Computer?

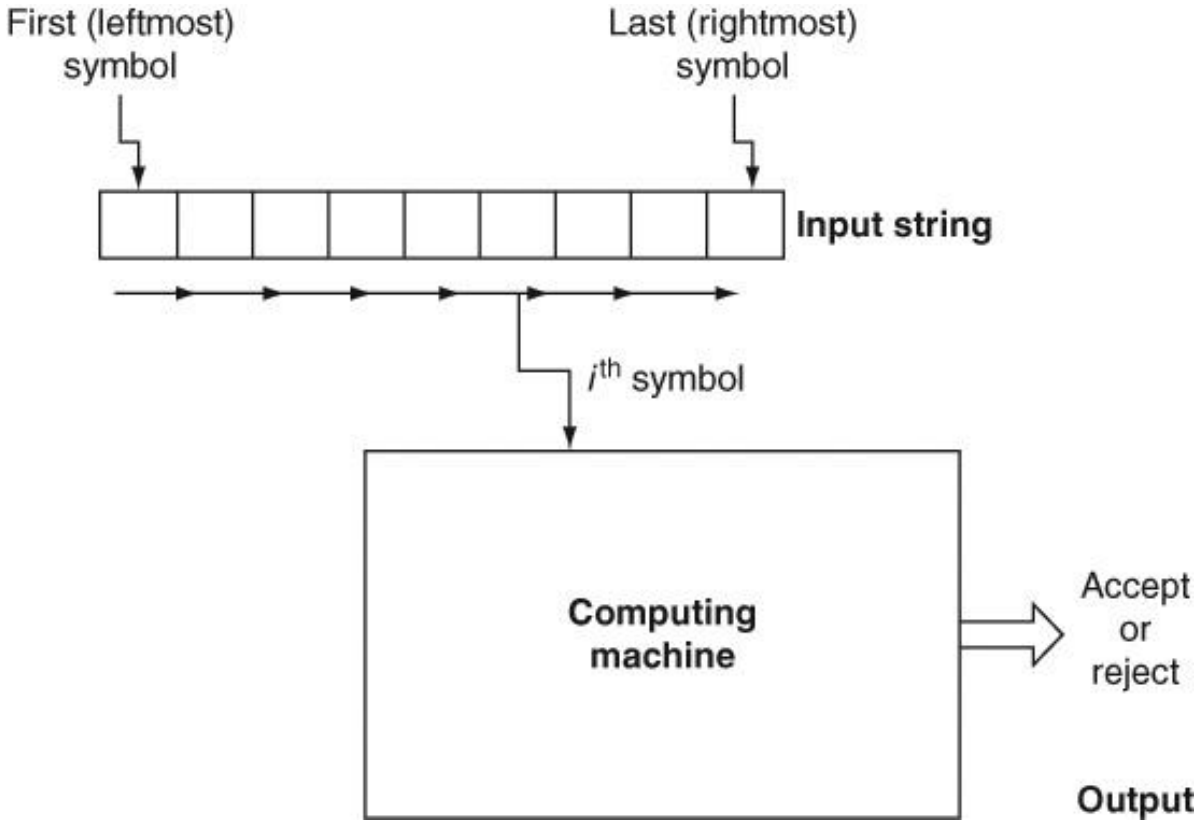
- Need not be a digital electronic machine with display, keyboard, etc.
- Not just a machine with input, output and processor
  - Kitchen grinders also have these!
- A computer is a machine that
  - Accepts inputs
  - Processes them to
  - Produce outputs,
  - Is capable of making decisions, and
  - Usually has storage (i.e., memory)
- We study abstract models of computers

The Idea of Computing

- Transforming input to output
- Usually involves storing in memory
- Usually involves making decisions
- Input and output are strings of symbols
- Computing is manipulating symbols
- Symbols are from a known set called the alphabet
- Strings can also be of null length: λ

Computing Machines

- Automata: Output is binary: yes or no, accept or reject, true or false
- Transducer: Output is a string of symbols



COMPUTING

- Internal states
- Change of states
- Finite number of states
- Efficiency is usually not a concern in this subject
- Whether it can compute and how much memory is used are the primary concerns

Computing Machines and Languages

- A language is a set of strings
- The set of all strings accepted by a computing machine is its language
- Formal languages are different from natural or human languages (e.g., English)
- Languages and problems are the same because any (computing) problem can be stated in the following two ways:
  - As a conventional problem: “Given … as inputs, … find/calculate/compute …”
    - E.g., add 2 and 3
- As a yes-or-no (or true-or-false) question: “Is … the correct answer for …?”
    - E.g., is 5 the sum of 2 and 3?
- The second way transforms any set of problems into a set of strings, i.e., a language

Classes of Machines and Languages

- Finite Automata: Chapters 2 and 3
- Regular Languages: Chapters 4, 5, 6
- Context-Free Languages: Chapter 7 and 9
- Pushdown Automata: Chapter 8
- Turing Machines: Chapter 10
- All other formal languages: Chapter 11
- Computability and undecidability: Chapter 12

What is the Science of Computing?

- What does it mean to compute? What is computable, what is not?
- What is a computer or a computing machine? What is the essential nature of a computer?
- What determines how powerful a computer is? What is computable by which kind of machine?
- What happens if we take memory out from a computer?
- What happens if we change RAM to special types of memory (e.g., a stack)?
- What classes of problems can a computer (be programmed to) solve?
- Are there unsolvable problems? What are the limitations of the very idea of computing?
- What is the relationship between types of computers and classes of mathematical functions?
- What is the relationship between languages, grammars and computing devices?
- How fast or efficient can a computation be?

Computing Questions: Examples

- What is the minimal number of steps involved in sorting a set of numbers? Why does it take that many steps?
- Why are trees more efficient than arrays for searching?
- Why can't we do binary search on a linked list?
- Why does sorting take the same number of steps no matter whether we sort:
  - Heavy or light objects?
  - Solids, liquids, colors or names?
  - On Earth, in outer space or in the Andromeda Galaxy?

Computing Model: Assumptions

- Stored table of instructions
- Linear sequence of memory units
- Single reading head
- Behavior depends only on current configuration
- Minimalistic instruction sets
- No awareness

What is Difficult for Computers?

- Why computers have all our pictures in high definition but cannot recognize any of us?
- Why computers can be our accountant but not our secretary?
- Why computers can calculate so accurately but still have errors (as in “computer errors”)?
- Why computers have entire dictionaries but don’t understand a word?
- Why computers need programming?
- Why computers can run medical systems but can’t cure themselves of viruses?
- Why a Web search engine does not distinguish mouse (the animal) from mouse (the computer gadget)?

What if Computers had..

- Multi-dimensional memory so that every memory cell has many immediate neighbors.
- Self-aware processors that “naturally” remember where they have been and “know” what they are doing.
- Human-like short-term memory so that the computer knows “naturally” what it just did or where (in which state) it had just been.
- Data-centric model, not a processor-centric one, wherein instead of specifying a table of instructions for processing the input we specify input and output data patterns.
- Associative memory so that any data element stored in memory can be retrieved instantly without an expensive search or indexing mechanisms.
- Non-deterministic processing or truly unlimited parallel processing or the ability to always guess the right choice! (See Chapter 3.)

How is Theory of Computation Related to..

- Data structures?
- Algorithm design and analysis?
- Programming?
- Computer organization?
- Operating systems?

Birth of the Science of Computing

- Can everything be known to man?
- Can all theorems be proven?
- Kurt Gödel: Incompleteness Theorem
- David Hilbert: Entscheidungsproblem
- Alan Turing: Computable numbers and Turing machines
  - Also: Computing Machinery and Intelligence
- Alonzo Church: Recursive functions
- The rest is.. history!

Computability, Intractability & Intelligence

- Not all problems are solvable
- The vast majority of practical problems are computable
- There are undecidable languages
- Many computable ones are too inefficient, i.e., intractable
- Even practical solutions do not often exhibit any intelligence
  - i.e., many solutions are rather dumb when compared to the human mind

Why Study Theory of Computation?

- It constitutes the essential science behind all computing.
- It takes one through the historical development of computing machines.
- The theory of formal languages came from linguistics and has influenced the further progress of language sciences including (computational) linguistics, natural language processing, search engines and the Web.
- Similarly, the design of programming languages and compilers is a direct application of the theory of computing.
- Moreover, the design and parsing of data representation languages such as XML are also applications of the theory of formal languages.
- The study of automata originated from attempts in designing automatic machines such as vending machines.
- Automata and formal language theory also forms the basis for significant aspects of digital and VLSI design, and hardware as well as software testing and verification.
- Finally, it appears that one who has a solid foundation in the theory of computing can cope with the rapidly changing world of new computing technologies and ever more programming languages and paradigms.

Key Ideas

- A computer is a machine that accepts inputs, processes them to produce outputs, and is capable of storing and making decisions about its inputs and outputs. A computer is a machine that computes an output for a given input while being controlled by a program or table of instructions. A computing machine is also called an automaton.
- The process of transforming the input of a computing machine to its output is called computing. It usually involves storing some of the input, output, or their intermediate forms in memory. It also involves making decisions during processing.
- Inputs and outputs are strings made up of symbols from an alphabet.
- In general, a computing machine is expected to process its input in a single left-to-right pass.
- The output is usually binary: the machine either accepts or rejects the input string. If the output is not just a yes-or-no answer, the computing machine is called a transducer.
- During a computation, an automaton changes its internal state from one state to another.
- If, at the end of processing the entire input, the automaton halts in a final or accepting state, it is said to accept the input. If it halts in a non-final state, it is said to reject the input.

KEY IDEAS (CONTD..)

- A formal language is a set of strings. A set of related problems can also be considered a formal language. The set of all strings accepted by a computing machine is its language.
- A formal language is specified by the rules of its grammar.
- We study classes of formal languages, their grammars and computing machines that accept them.
- Computing has a science of its own. We can define the science of computing or computer science as the science explains computing phenomena that cannot be explained by traditional sciences. The science of computing is often also called Theory of Computation or Theory of Computing.

KEY IDEAS (CONTD..)

- The science of computing makes several significant assumptions about the model of computing. The science is all about figuring out the consequences of the assumptions made in the computing model on computability, data structures, algorithms and programming.
- Computer science was born out of philosophical and mathematical challenges faced in trying to determine the truth or falsehood of all possible formal statements in mathematics.
- Not everything is computable. There are limitations to the very idea of computing. Even among computable problems, many are intractable, that is, they take too long to compute.
- The behavior of computers built on Turing's model rarely can be seen as intelligent. Computers, although they are capable of elaborate calculations and intricate symbol manipulations, have remained rather dumb. It is not clear whether Turing's model of computation is good enough to explain how human beings behave intelligently.
- Apart from being the foundational theory of computer science, the theory of formal languages and automata has a number of important practical applications in areas ranging from programming language design and compilers to XML, web technologies and hardware and software testing.

End of Chapter 1