Basrah University

Faculty of Education for Pure Sciences

Computer Department

A.L. Hend Muslim Jasim

COMPUTATION THEORY

Introduction

The **theory of computation** is the branch that deals with what problems can be solved on a model of computation, using an algorithm, how efficiently they can be solved or to what degree (e.g., approximate solutions versus precise ones). The field is divided into three major branches:

1- automata theory and formal languages,

2- computability theory,

3- computational complexity theory,

which are linked by the question:

"What are the fundamental capabilities and limitations of computers?".

In order to perform a detailed study of computation, computer scientists work with a mathematical abstraction of computers called a model of computation. There are several models in use, but the most examined is the Turing machine. Computer scientists study the Turing machine because it is simple to formulate, can be analyzed and used to prove results, and because it represents

what many consider the most powerful possible "reasonable" model of computation. It might seem that the potentially infinite memory capacity is an unrealizable attribute, but any decidable problem solved by a Turing machine will always require only a finite amount of memory. So, in principle, any problem that can be solved (decided) by a Turing machine can be solved by a computer that has a finite amount of memory.

1-a: Automata Theory

Automata theory is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata. Automata comes from the Greek word (Αυτόματα) which means that something is doing something by itself. Automata theory is also closely related to formal language theory, as the automata are often classified by the class of formal languages, they are able to recognize. An automaton can be a finite representation of a formal language that may be an infinite set. Automata are used as theoretical models for computing machines, and are used for proofs about computability.

1-b: Formal Language Theory

Language theory is a branch of mathematics concerned with describing languages as a set of operations over an alphabet. It is closely linked with automata theory, as automata are used to generate and recognize formal languages. There are several classes of formal languages, each allowing more complex language specification than the one before it, i.e., Chomsky hierarchy, and each corresponding to a class of automata which recognizes it. Because automata are used as models for computation, formal languages are the preferred mode of specification for any problem that must be computed.

2- **Computability theory**

Computability theory deals primarily with the question of the extent to which a problem is solvable on a computer. The statement that the halting problem cannot be solved by a Turing machine is one of the most important results in computability theory, as it is an example of a concrete problem that is both easy to formulate and

impossible to solve using a Turing machine. Much of computability theory builds on the halting problem result.

Another important step in computability theory was Rice's theorem, which states that for all non-trivial properties of partial functions, it is undecidable whether a Turing machine computes a partial function with that property.

Computability theory is closely related to the branch of mathematical logic called recursion theory, which removes the restriction of studying only models of computation which are reducible to the Turing model. Many mathematicians and computational theorists who study recursion theory will refer to it as computability theory.

3- Computational complexity theory

Complexity theory considers not only whether a problem can be solved at all on a computer, but also how efficiently the problem can be solved. Two major aspects are considered: time complexity and space complexity, which are respectively how

many steps does it take to perform a computation, and how much memory is required to perform that computation.

Computer scientists express the time or space required to solve the problem as a function of the size of the input problem. For example, finding a particular number in a long list of numbers becomes harder as the list of numbers grows larger. If we say there are n numbers in the list, then if the list is not sorted or indexed in any way, we may have to look at every number in order to find the number we're seeking. We thus say that in order to solve this problem, the computer needs to perform a number of steps that grows linearly in the size of the problem.

To simplify this problem, computer scientists have adopted Big O notation, which allows functions to be compared in a way that ensures that particular aspects of a machine's construction do not need to be considered, but rather only the asymptotic behavior as problems become large.

Perhaps the most important open problem in all of computer science is the question of whether a certain broad class of problems denoted NP (nondeterministic polynomial time) can be solved efficiently.

Models of computation

Aside from a Turing machine, other equivalent models of computation are in use.

- Lambda calculus
- Combinatory logic
- μ-recursive functions
- Markov algorithm
- Register machine