Part I

Computing

Introduction to Part I

The field of Theoretical Computer Science (TCS) covers many areas of computer science and modern mathematics. The topics studied in TCS include algorithms, data structures, computational complexity, combinatorial search and optimization, parallel and distributed computation, probabilistic computation, cryptography, program semantics and verification, as well as computational aspects of logic, geometry, number theory, and algebra.

New research directions in classical areas such as automata theory and information theory are also part of TCS, as are various areas of AI such as machine learning, computational learning theory, theorem proving, and constraint programming. In recent years, whole new areas have been added to TCS such as quantum computing, computational biology, computational economics and algorithmic game theory. Today, the field is still rapidly expanding as new algorithmic or computational theories of natural and artificial phenomena emerge. For example, recent advances in neuroscience are opening up exciting new directions for research in developing theories of cognitive computation, of brain functionality, and of consciousness.

The field of TCS has many major mathematical open questions at its core. Perhaps the most famous is the $P =? NP$ problem in the area of computational complexity, which is widely regarded as one of the most important open problems in mathematics today. In this theme we introduce just a few of the areas of TCS that have a major significance for innovation in computing and communications technologies.

Mathematics, Models and Architectures

In the first chapter the relationships between mathematics, models of computation, and the design, analysis and optimization of hardware and software architectures are explored. Mathematics and models of computation have been at the heart of computer science since the earliest research on computing by Turing, von Neumann and other pioneers. Today, models guide how we design and analyze algorithms, how we design and compare architectures, and how we design software that can be automatically adapted to run efficiently on different architectures. For the past 50 years a major goal of computer science research has been to develop a universal "post-von-Neumann" parallel model for algorithms, software and architectures. Some of the history of attempts to produce such a new universal parallel model are described. also Mathematics continues to be a central element in all of this research.

Mathematics and Software Verification

Mathematical proofs and computer programs can be modeled as the same types of mathematical objects. This makes it possible to use mathematical theories and logics to prove properties of computer programs. In the second chapter, this area of theoretical computer science is explored. Key mathematical concepts such as the lambda calculus, type systems, Hoare logic, and weakest preconditions are introduced, and software techniques such as program analysis, model checking, and automated and interactive theorem proving, are described. Recent research results are given on specifying and verifying operating system kernels, on security-critical components and protocols, and on correctness of synchronization primitives under weak memory models. The chapter closes with a discussion of some of the practical challenges of applying formal methods in industry, and of the benefits that can be obtained from formal methods and formal verification.

Mathematics for Quantum Computing

The concept of computation as a mathematically precise notion was formalized in the 1930s by Emil Post, Alonzo Church, and perhaps most prominently, by Alan Turing. Today, more than 80 years later, it remains the case that what we regard as computable is precisely that which corresponds to the Church–Turing Thesis. A problem is computable, or computationally solvable, if and only if it can be solved on a Turing machine. Our analysis of the computational complexity of problems also assumes that our machines correspond to these classical models. In recent years, the concept of quantum computing has emerged as an interesting new alternative model of computation. Research on the theory and practice of quantum computing is proceeding on many fronts. The third chapter explores this area. Due to the radically different way in which information is represented and manipulated in quantum systems, there is the potential to search for solutions in a space exponentially larger than its classical counterparts, with the same number of bits. With this potential in mind, quantum algorithms have been developed in fields ranging from number theory and algebra, simulation, optimization, and machine learning. However, as the chapter describes, many significant theoretical and practical problems remain to be solved in order to fully achieve this potential.

Mathematics for AI : Categories, Toposes, Types

The fourth chapter reviews possible connections between theoretical computer science and artificial intelligence. These connections are essentially based on notions from contemporary mathematics, and are heavily influenced by the revolutionary and visionary ideas of Alexander Grothendieck. The generalizing capability of the notions Grothendieck developed (all of them widely using the category language) encompasses computer science and machine learning. After a reminder about category theory, the history of this contemporary mathematical field is described: Grothendieck's discovery of toposes as a category equivalent to the category of sheaves of sets on a site, the existence, in each topos, of a subobject classifier which shows the deep connection with logic.

A description of neural networks as a Grothendieck site is then presented with an explanation of which Grothendieck topology is required to ensure functionality. Many characteristics of neural networks (for example, weights, training datasets and *a priori* knowledge) can then be seen as objects in the topos of sheaves corresponding to the base site of the neural network architecture. We can even take into account the invariance to some symmetries (like CNNs for the group of spatial translations) by generalizing toposes to stacks (fibered in groupoids).

Type intuitionist languages can be interpreted using toposes. In this case, a type is an object of the topos. Martin-Löf's dependent type theory constituted a significant step forward. A further advancement was the construction of new high-level functional programming, which provided effective methods of computation with a large mathematical scope. In turn, this facilitated proof assistance by computers. This path has led us today to programming with homotopy type theory. The definition of infinite-categories/infinite-toposes has paved the way for notions that give all known models of the theories of types of Martin-Löf and the univalent theories of Voevodsky.