Theoretical Computer Science in Transition

John E. Savage
Department of Computer Science
Brown University
Providence, RI 02912-1910

June 6, 1996

If the past is prologue to the future, computer science will continue to be extremely successful. In a few short decades computer science has lead to revolutions in work, recreation and societal interactions. There are few technologies whose impact has been greater. Theoretical computer science has played a central role in these developments and is destined to play a central role in the future.

This document provides a brief discussion of the role of theory in computer science as well as a quick survey of contributions by the field. It takes the position that the unpredictable nature of research argues for a liberal attitude toward research funding while maintaining standards. Thus, it argues against picking specific research agendas or betting large amounts of support on a few research groups. Finally, it lists important areas of computer science that have the potential to attract the attention of theoretical computer scientists.

1 The Nature of Computer Science

A distinguishing characteristic of computer science is the enormous gap that exists between the simple instruction sets of our computers and the complexity of useful software. This dichotomy produces intellectual challenges of the highest order. Today many important applications contains several million lines of code, too many for one person to read in a year and impossible for one person to fully understand. VLSI chips are now being manufactured containing more than one hundred million transistors, again posing an almost impossible problem for human comprehension.

2 The Role of Theory

Modern computer science presents immense challenges to which both experimental and theoretical computer scientists respond. Researchers in both communities explore the frontiers of the field. Experimental computer science is devoted primarily to harnessing the power of computation whereas theoretical computer science seeks largely to understand the limits on computation and the power of computational paradigms. Theoreticians also develop general approaches to problem solving. The boundary between experimental and theoretical computer science is not firmly marked; theoreticians invent ways of harnessing computation and experimentalists apply analysis in their work. Both communities share a common vision, however, which is to make computation useful.

Experimental computer science is most effective on problems that require complex software solutions such as the creation of software development environments, the organization of data that is not tabular, or the construction of tools to solve constrained optimization problems. The approach is largely to identify concepts that facilitate solutions to a problem and then evaluate the solutions through construction of prototype systems.

Theoretical computer science is most effective on problems that are susceptible to precise formulation such as the characterization of hard computational problems, the encryption messages for large scale use, the use of large degrees of parallelism, the coordination of activity on distributed systems, and the determination of limits on the performance of computers. The approach is to develop computational models and methods of analysis and then apply the analysis to problems that would be executed on these models.

One of theoretical computer science's most important functions is the distillation of knowledge acquired through conceptualization, modeling and analysis. Knowledge is accumulating so rapidly that it must be collected and prepared for transmission to the next generation. While this task is not limited to theoretical computer science, it is one of its principle functions.

3 Contributions of Theoretical Computer Science

The success of theoretical computer science is measured both by its impact on practice and by its contribution to the understanding of computation and the limits thereof.

Theoretical computer science has made contributions to many areas. In the 1930's Church, Godel, Kleene, Turing and others developed fundamental models of computation that continue to be relevant. In the 1940's Shannon, von Neumann and Weiner and advanced computer science through the study of circuits and other models of computation. The 1950's and 60's saw development of formal languages as well as models of computation such as the finite-state machine and the pushdown automaton. This work continues to be used in programming and the design and translation of languages today.

Algorithms and data structures is an area that has seen an enormous amount of effort by theoreticians whose impact on practice has been immense. Activity began in this area as early as the 1950's and has grown in importance to the present time. Among its notable achievements are fast and efficient algorithms for sorting, searching and data retrieval, computations on strings and graphs, computational geometry, scientific computation, and constraint and optimization problems.

During the 1960's the foundations of computational complexity were developed. Hierarchies of problems were identified and speed-up and gap theorems established. This area flourished in the 1970's and beyond, in particular, through the identification of the **NP**-complete, **P**-complete and **PSPACE**-complete languages. In the 1970's connections were established between Turing and circuit complexity, thereby spawning a new examination of this topic. Space-time tradeoff studies were also initiated. Research in all of these areas continues today.

The late 1970's saw the emergence of the VLSI model as well as methods of analysis for it. This led to a burst of activity on the part of theoreticians in the 1980's in this area and to the rebirth of computer architecture under the label of VLSI.

The early 1980's saw the introduction of the first models for memory hierarchies and to the development of I/O complexity as an important area of research. Several different models capturing various aspects of memory hierarchies emerged and research in this area is now leading to a redesign of I/O systems.

Public key cryptography emerged in the mid to late 1970's and has spawned a flood of interesting ideas having to do with secure communication and interactive and zero-knowledge proof systems. Today systems based on some of these ideas are used to provide security for purchases over the Internet.

Beginning in the late 1970's, theoretical research on parallelism has resulted in new parallel computer architectures and new paradigms for programming parallel machines that have been offered in products. We have learned that even the "simple" PRAM parallel machine model is very difficult to program efficiently. New algorithms have had to be invented for old problems because old ones could not exploit parallelism. Through theoretical studies we have come to better understand why some problems are much more difficult to program in parallel than others. Since computers will become increasingly parallel as we approach the limits of performance of serial VLSI-based computers, the impact of research in this area will continue to grow.

The formal study of the semantics of programming languages, an area that became active in the 1960's, has led to a much better understanding of programming constructs, thereby guiding the design of successors to pioneering languages, such as LISP, Prolog, and Simula, that were invented to simplify the design and maintenance of complex applications. The study of programming language semantics also provides a fundamental tool for the development of static analysis and program transformations, two key steps in compiler optimization.

The formal study program verification techniques also began in the late 1960's. It has also had an important impact on programming language design. Recent results in model checking show high promise for a significant industrial impact.

Relational database theory emerged in the early 1970's and, together with many advances in data structures, optimization methods and other contributions, has had a major impact on practice. Today relational database are commonplace commercial products that are widely used throughout the world.

The often subtle behavior of concurrent and distributed systems, which became and active area of research on the late 1960's and early 1970's, has benefited immensely from formal modeling and analysis. Models of concurrent systems have been introduced and tools to understand concurrency applied. Transactions processing, which involves issues of serializability, locking, logging, timestamping, replication and orphan handling, have also been subjected to modeling and analysis, thereby insuring correct handling of these issues. The same is true of I/O which depends on the correct and efficient implementation of data structures used in this context. All of this work has a direct impact on practice.

Learning theory, which saw a burst of activity on the part of theoretical computer scientists in the 1980's, has led to new and deeper understanding of the problems for which active and passive learning is possible, thereby serving as a guide for the intelligent use of computers.

Computational biology and the human genome project have benefited from the algorithm design and complexity analysis that theoretical computer scientists have brought to them. This work will continue to be important in the future.

Approximation algorithms for hard computational problems have long been sought. The development of a theory of polynomially checkable proofs provides a major improvement in our understanding of the degree to which **NP**-complete problems are approximable. This knowledge will serve as an important guide in the search for good approximation algorithms.

4 Research Funding Policies

While research is by its nature highly unpredictable, successful research can be extraordinarily valuable. In fact, a few truly successful researchers can justify support for a large community of researchers.

Research is unpredictable because it is difficult to forecast the future or estimate the skill and stamina of individual researchers. It is hard to determine who has chosen the right problem and whether that group or person will be successful or not.

Faced with these uncertainties, a prudent research funding policy invests in more people with smaller grants in a larger number of areas than the converse. Such a policy increases the odds that some investigators will be successful. A wise funding policy will also increase the interaction between researchers, thereby seeking to encourage cross fertilization and creativity in problem solving.

It is important to note that good researchers are very aware of the objectives of their field and need nurturing rather than direction. Should they stray for too long from a path that appears to be directed toward the problems of the field, this is cause for concern. However, the unpredictability of research demands that patience and tolerance be the hallmarks of a successful research funding policy. A research direction that may seem unpromising to some, for example, an exploration of proof methods in the pursuit of the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ question, may eventually be very productive.

Standards must also be met. Funding should be provided to those who are thought to be most capable of doing the research. Peer evaluation works very well in this regard but there is also a need for enlightened, unilateral decision making on the part of those with funding responsibility. Variety in research funding criteria as well as in research topics reduce risk

As research fields mature, there is an increasing need to apply the fruits of research. Often the application of research results is far from simple and offers important and interesting intellectual challenges. Some researchers may wish to follow research topics as they become more applied, thereby serving the profession as well as themselves.

5 Directions for Research

Computer science is a vibrant field, full of potential. Not only are new products emerging every day, the demands of the field create new computational problems that need solution, many of them highly complex. Since the computer scientist is needed when the computational problem is hard, this means that our field is likely to be in demand for the foreseeable future.

Many areas offer problems of sufficient complexity and importance to warrant the attention of the theoretical computer scientist. Some of these currently lack the precision that usually attracts theoreticians but, through joint work with experimentalists, this can change.

A list of research areas that may attract theoreticians is given below. It is not intended to be complete nor the description of a research agenda. Our community is rich in diversity which should continue to be encouraged. Together we can come to agreement on areas that have high potential, bearing in mind, of course, that it is undesirable to invest too fully in too few areas, especially areas that have received a great deal of attention in the past.

- Parallel and distributed computing are increasing steadily in importance. Manufacturers now offer symmetric multiprocessors as well as support for locally and remotely distributed computing. The problems they encounter in coordinating activity, accessing data, securing transactions while achieving high performance are challenging.

- Many companies are now very dependent on high-performance computing for the simulation of complex designs and for the planning and scheduling of work of all kinds. These areas have profited and will continue to profit from the research of theoreticians.

- The growing complexity of software offers immense challenges to specification and understanding for which the talents of theoreticians are appropriate. This is an area in which interaction with experimentalists will lead to new methods for specifying and verifying software systems. Interestingly, while algorithms and data structures are important in the design of large complex software systems, they occupy a relatively small fraction of a designer's time.

- As computer systems become more complex, understanding their behavior and performance will present daunting challenges. This topic will increase in importance as the profit margins on computer systems shrink.

- Data storage and retrieval in the future will not be limited to relational and object oriented databases. Data in a large variety of formats and types will commingle and search and retrieval mechanisms will be invented to access them. Theoreticians can play a role here by laying the foundations for the logical development of this area.

Most traditional areas of computer science will continue to evolve and require the attention of theoretical computer scientists.

New areas will emerge that are hard to predict. For example, ubiquitous and mobile computing in which many users migrate their work and attention to different computers will offer many challenges including the obvious one of resource allocation. Quantum computing has the tantalizing prospect of being a new and highly parallel computing model that may be used to solve very hard problems in an instant.

Computer science is a thriving area that will continue to thrive for decades to come. The role for the theoretician in computer science is to create new computational models and to explore the power and limits of these models for important problems. As long as the computational problems are very hard there is a role for the theoretician.