The Role of Theory Today

Jeffrey D. Ullman
Dept. of Computer Science
Stanford University

The work of Hartmanis and Stearns, for which they rightly and belatedly won the Turing award, was both amazingly insightful and widely useful. It led to a long sequence of important innovations, such as work on cryptographic complexity, and it is a principal justification for theoretical research today. Further, theory has had triumphs outside the computational complexity area. Without wishing to be exhaustive, I could cite work on programming-language semantics that led to real advances in language design, the use of automaton-theoretic ideas in practical protocol verifiers, and many specific algorithms that have been used in practice.

Yet looking back to the 1960's, I see a naive optimism that was based on too little evidence. The work on $O(n \log n)$ sorting algorithms or Strassen's matrix-multiplication algorithm, for example, caused us to believe that all problems would soon yield to algorithms that were significantly more efficient than the obvious ones. LR(k) parsing — a wonderful idea deeply rooted in language and automata theory — suggested that all aspects of compiling and other important systems software would soon yield to formalization and theoretical analysis.

The theory community was not alone in its naivety. AI was similarly taken in by early successes, such as Samuels' checker-playing program, to believe that very hard problems were in fact easy. This view has led to periodic, overly optimistic predictions of success that continue to trouble that field.

Thus, despite the successes mentioned above, as the theory field grew there was a tendency to attack problems obliquely, assuming that we were contributing to an important goal. Papers would be written to address the issues raised by other papers. Then, another round of work would address these papers, and so on. In the process, the real problems that spawned the original paper were forgotten, and the community began to imagine that the research embodied in the papers was the ultimate goal. (I confess to being as guilty of this behavior as anyone in the field, by the way.)

Is the theory community therefore doomed to become ineffective within the broader discipline of computer science? I do not believe so for a moment. However, we need to reinvent theory for the next generation if we are to have the same impact now as a generation ago.

Not everything needs to change. First, there is a core of theory, roughly the complexity theory outlined in the Hartmanis article, that needs to be preserved and advanced; this part needs no reorientation. Second, there is a continuing need to teach theory extensively to CS students at all levels. Theory forms an enduring core of the field; automata theory will still form the basis of various software components at a time in the future when C++, WIMP interfaces, and other fashionable elements of today's curriculum are forgotten. Third, we must continue to uphold standards of rigor in all branches of research, to ask the hard questions, and be ready to challenge the bogus.

However, in the research arena, there is a need for most of theory to move much

closer to the practice. As computing becomes more complex, with "all the easy stuff having been done already," there is a progressively greater need for theoretical study of approaches prior to experimentation. Appropriate models and techniques are vital. Theory researchers should be working in teams with those focusing on particular problems, in order to make both the theorists and the pragmatists more effective. Especially, many more of the research students who wish to study theory should be encouraged also to apply that theory in a real system-building project.

Thus, we are faced with a choice. Are we going to spend the next generation writing papers for other theoreticians to read? Or shall we take our special skills and approaches out into the field and have impact on the world? I recommend the latter.