SIGACT News Complexity Theory Column 14

Lane A. Hemaspaandra
Dept. of Computer Science, University of Rochester
Rochester, NY 14627, USA lane@cs.rochester.edu

Introduction to Complexity Theory Column 14

As you probably already know, there is an active discussion going on—in forums ranging from lunch-table conversations to workshops on "strategic directions" to formal reports[1]—regarding the future of theoretical computer science. Since your complexity columnist does not know The Answer, I've asked a number of people to contribute their comments on the narrower issue of the future of complexity theory. The only ground rule was a loose 1-page limit; each contributor could choose what aspect(s) of the future to address, and the way in which to address them. The first installment of contributions appears in this issue, and one or two more installments will appear among the next few issues.

Also coming during the next few issues: the search for the perfect theory journal, and (for the sharp-eyed) Lance Fortnow dons a clown suit. Finally, let me mention that work of Russell Impagliazzo resolves one of the open questions from Complexity Theory Column 11.[2]

Guest Column:

The Future of Computational Complexity Theory: Part I

Comments by C. Papadimitriou, O. Goldreich/A. Wigderson, A. Razborov, and M. Sipser

1 *On Extroverted Complexity Theory, by Christos H. Papadimitriou*[3]

Complexity theory has come a long way in the thirty years since its inception. It has identified some of the most fundamental and deep problems related to computation, it has developed a powerful methodology for attacking them, and it is broadly considered as one of the most challenging mathematical frontiers. Considered as a pure mathematical discipline in the pursuit of mathematical insight and depth, complexity is successful and well-established. In this note, however, I would like to concentrate on complexity as an applied mathematical discipline whose function is to gain insights into the problems of the natural, social, and applied and engineering

[1] For a quick and bracingly varied path into the discussion, I'd point to two documents: ftp://ftp.cs.washington.edu/tr/1996/03/UW-CSE-96-03-03.PS.Z and http://theory.lcs.mit.edu/~oded/toc-sp.html.

[2] In more detail: Impagliazzo [Imp96] resolves Problem 3 of the "Worlds to Die For" Complexity Theory Column [HRZ95]. In fact, his result is even stronger than what we conjectured, as he shows that with probability $1-2^{-\Omega(n)}$ over the set of oracles there is a relativized pseudorandom generator that is secure against oracle circuits of size 2^{cn} for some constant $c > 0$. He first shows that a random function behaves for almost all oracles like a one-way function and, thus, the general method of Håstad, Impagliazzo, Levin, and Luby [HILL91] for converting a one-way function into a pseudorandom generator can be used to obtain the above result. As we noted in the column, this method provides a provably secure way of privatizing random bits (for more details, see [Zim96]).—L. Hemaspaandra and M. Zimand

[3] Computer Science Division, University of California Berkeley; christos@cs.berkeley.edu. Adapted from the introduction of [Pap96].

sciences. This aspect has been somewhat peripheral to what we usually mean by "complexity theory," but I believe it is important for its future.

Understanding the position of complexity theory within the realm of scientific inquiry is an important project which is, of course, well beyond the scope of this note; here I shall only refer anecdotally to six distinct ways in which complexity theory has reached out and touch other fields:

Complexity as NP-completeness. One of the major achievements of complexity theory is the living connection it has forged between application problems and modes of resource-bounded computation. This is typically done through the key notion of *completeness*, of which NP-completeness is of course the most popular kind. Completeness comes so natural to a complexity theorist, that it is easy to forget what an important and influential concept it has been. In fact, outside theoretical computer science, "complexity theory" is often understood—unjustly, to be sure—as synonymous to "NP-completeness."[4]

Complexity as mathematical poverty. One of the fundamental theses that seems to be almost universally accepted and practiced in computer science is that algorithms—and efficient algorithms in particular—are the natural outflow of mathematical structure discovered in applications.[5] If we accept this implication, then we must also espouse the contrapositive one, namely, that *complexity is the manifestation of mathematical nastiness.* Complexity has been often and brilliantly used within computer science and mathematics in this *allegorical* way; a computational problem is formulated and proved hard for the sole purpose of pointing out the mathematical difficulties involved in an area or approach (see [HLY80] for an early example from database theory).

Complexity as metaphor. Often the implication discussed in the previous paragraph is composed with the metaphors of an application domain or other scientific discipline with sometimes exquisite results. "Complexity" may mean "chaos" in the domain of dynamical systems [BPT91], "unbounded rationality" in game theory [PY94]—and perhaps "genetic indeterminism" in genetics, "cognitive implausibility" in artificial intelligence, and so on.

Complexity as blessing in disguise. Cryptography is of course the best-known example here, but not the only one. For example, [BTT92] point out that complexity can be desirable in political science, as evidence that an electoral protocol is resistant to manipulation.

Complexity as herculean sword. The mythical monster Hydra grows three heads for each one cut off by Hercules.[6] When complexity theorists point out obstacles to an approach, several novel alternative approaches typically develop: "It's NP-complete? Then we'll concentrate on planar graphs, or on random graphs, or else we'll approximate." And so on.

Complexity as beauty contest judge. When many alternative approaches have been proposed for a problem (computational or conceptual), rigorous criteria for evaluating them are needed. Complexity can come in handy in such a situation. For example, in [DP94] we proposed complexity as a criterion for comparing the feasibility and desirability of "solution concepts" (approaches to when a proposed protocol for splitting goods is fair) in mathematical economics, and in [GKPS95] complexity was used in sorting out alternative proposals in knowledge representation—an important subfield of artificial intelligence.

---

[4] Among treatments of complexity theory by scientists outside our field this one is not the most unfair or ignorant; compare with [CPM94], for example.

[5] The only challenge of this principle within computer science comes, I think, from neural networks and other metaphor-based algorithmic paradigms.

[6] Incidentally, this is not complexity's first brush with Hercules: Knuth [Knu74] had proposed "herculean" as one of the possible terms for the concept that is now, thankfully, known as "NP-complete." According to Knuth, Ken Steiglitz counterproposed "augean," a term which Greek mythology buffs will find both hilarious and appropriate.

2 *On The Usefulness of Hard Problems, by Oded Goldreich[7] and Avi Wigderson[8]*

We were asked to write on the future of Complexity Theory. Given the past (and present) of our field, which celebrated so many achievements, many in surprising, unanticipated directions, we feel it unwise to predict the prevailing directions in say 10 or 20 years. Nevertheless, we are confident that if our field continues to attract the same calibre of creative minds, and is given the freedom (and minute resources) to pursue its internal agenda, it will continue to thrive. Moreover, the importance of its findings to other areas of computer science and engineering, as well as to other sciences and (yes!) humanities will continue to grow.

To justify this strong prediction, we consider what complexity theory has done with the classical problem of integer factorization: *given an integer N, find its prime factors.* Mathematicians have studied this problem for centuries, searching for an efficient factoring algorithm even before the notion of an efficient algorithm was defined. This task has failed so far, which may very well mean that factoring is infeasible. SO WHAT? Complexity Theory has managed to use this infeasibility as a pivot for a variety of fundamental discoveries and theories of very general nature. This is true to such an extent that a list of topics as below can be used for a graduate course which will be offered not only to all computer science students, but also to students of other disciplines. In fact, it is high time that our community starts to disseminate the intellectual contents of the theory of computation, and courses of the above form may be a good start.

The items below all follow from the assumed *infeasibility of factoring.* This, as well as the consequences, are stated only informally for obvious reasons, with the understanding that they all have precise statements which make them mathematical theorems.

- **Data Representation is Important.** This point, which is the main focus of our courses on Data Structures and Efficient Algorithms, is driven home forcefully here. If $\sum_j \alpha_j 2^j = N = \prod_i p_i^{e_i}$, then each side of this equation represents the integer N, and the two representations (binary expansion and prime factorization) are equivalent from the information theoretic viewpoint. However, they are drastically different computationally: the LHS can be easily computed from the RHS, but the reverse direction in general is infeasible. Moreover, some computational problems, like solving certain polynomial equations modulo N, are feasible given the RHS but infeasible given the LHS.

- **Pseudorandomness Exists.** There are efficient deterministic procedures that take a few random bits and stretch them to a much longer (pseudorandom) string, which looks random to every efficient algorithm. Thus deterministic procedures can greatly expand computational entropy. This is in stark contrast to the information theoretic analog (deterministic procedures can never increase entropy), and physical intuition (cf., the preservation of mass and energy).

- **Randomness can be Eliminated when Computing Functions.** A straightforward consequence of the previous item is that any probabilistic algorithm (for computing a function) can be replaced by a deterministic one which is almost as efficient. The latter will simply enumerate all the possible pseudorandom strings, arising from all possible short seeds, and decide by a majority vote.

[7]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, ISRAEL. E-mail: oded@wisdom.weizmann.ac.il.

[8]Institute for Computer Science, Hebrew University, Givat Ram, Jerusalem, ISRAEL. E-mail: avi@cs.huji.ac.il.

- **Cryptography is Possible.** Secure public-key encryption, unforgeable digital signatures, distributed coin-flipping, electronic voting schemes and electronic cash transfer can all be performed digitally, by communicating computers. In fact, *every* distributed protocol with arbitrary privacy constraints[9] can be implemented in a way that is resilient to arbitrary faults by any number of the players. In other words, players may emulate the existence of a trusted party in a setting in which no such trusted party exists (and furthermore in which many parties cannot be trusted at all). The effect of these techniques on the real world is already immense and is further growing rapidly.

- **Zero-Knowledge Proofs.** A party having a proof of an arbitrary mathematical theorem T can convince anyone that T is true, without giving away anything besides the validity of T. In particular, after getting such a proof, one will be convinced that T is true and still be unable to prove this to others. This paradoxical notion contradicts popular beliefs that obtaining a proof of an unknown truth necessitates learning something new.

- **Is Learning Feasible?** Within several standard models of learning it is infeasible to learn even simple concepts described by short formulae. This sheds new light on the fundamental scientific task of understanding the learning process.

- $\boldsymbol{P \neq NP}$. In particular, a variety of important computational problems admit no feasible (exact and even approximate) solutions. Thousands of such problems are known, and hundreds are discovered each year, in a variety of scientific and engineering disciplines. These problems include famous examples like Satisfiability of Boolean Formulas, The Traveling Salesman Problem, and Integer Programming.

- **There are no "Natural Proofs" of $\boldsymbol{P \neq NP}$.** This recent research (partly) explains our failure so far to prove that the problems above are indeed infeasible. First, it abstracts all proof techniques which were used in demonstrating the known lower bounds, both structurally (as Natural Proofs) and logically (as a certain fragment of Peano Arithmetic). Then it shows that any proof falling into either of these categories cannot be used to establish a superpolynomial lower bound for general circuits.

When considering the contrapositive of the above implications, one is amazed to discover that any efficient learning algorithm for short formulae, any theorem with no zero-knowledge proof, any function which requires randomness for efficient computation, and any nontrivial circuit lower bound using known methods *can all* be converted into an efficient factoring algorithm!

And what if factoring has an efficient algorithm? Well, first of all, due to the nearly-universal use of the RSA Cryptosystem, the consequences on information privacy and world economy can be devastating. But scientifically speaking, this alone would leave all items above intact, since other "one-way" functions would serve just as well as factoring. Moreover, while some of these items are essentially equivalent to the existence of one-way functions, others hold even under seemingly much weaker conditions.

Meanwhile, the search for efficient factoring algorithms continues. Active collaboration of Mathematicians and Computer Scientists has led to impressive progress in the last decade, using known results in Number Theory as well as establishing new ones. On an orthogonal direction, the recent Quantum polynomial-time algorithm for factoring has greatly enhanced efforts of Physicists towards the study of the feasibility of the Quantum Computer model. What will be next?

[9]For example, suppose some parties wish to play a standard game of Poker over the telephone.

3 *Proofs, Computations and Practical Applications, by Alexander A. Razborov*[10]

The general idea that computations and proofs have much more in common than it may appear from the first sight is being developed in at least three sister communities descending from the classical mathematical logic. Complexity Theory contributed to that purpose notions like *interactive proofs*, *probabilistically checkable proofs* and *transparent proofs* challenging such fundamental issues as rigorousness and verifiability. Feasible Proof Theory suggested that even classical proofs *have their own intrinsic complexity*, and it is extremely closely related to the ordinary complexity of algorithms. Finally, the "LICS community" analyses proofs in formal systems whose *semantics* is intended to capture the process of computation happening in the real software.

In several last years we started to see some renewed communication on this topic between the fields once united in the framework of mathematical logic. My prediction is that in upcoming years these relations between computations and proofs will receive even closer attention on the side of Complexity Theory (other parties involved already do it on the professional basis). Our community certainly has something here both to say (e.g., how to prove anything about the complexity of proofs? Or how to prove efficiently the correctness of programs, rigorously defining first what it means?) and to ask (e.g., what makes our own fundamental problems so difficult to solve?)

Coming to another part of the question, "Where *should* Complexity Theory go?", there's a lot of effort now to try to increase the impact of our field on key application areas. But in my opinion it is not quite clear whether Complexity Theory should go anywhere at all or it would be more useful staying where it is. The mission of this discipline is to provide a bridge for the traffic of ideas and concepts (with a handful of exceptions, not the results themselves!) between pure mathematics and Computer Science. Try to pull it to one side (say, for the purpose of making a junction instead), and firstly you will no longer have anything to cross the river upon, and secondly you may discover that the bridge's remnants are less useful on the land than expected as the construction was designed for different purposes. However, in my opinion that part of semi-applied research in Complexity Theory which *develops according to the* **internal** *logic of our field* has to be strongly encouraged (continuing the analogy with the bridge, it is simply our duty to provide as convenient access to the traffic across it as we can manage). The work on efficient program verification mentioned above in quite a different context is one good example of this.

4 Is the Handwriting on the Wall ?

Reflections on the future of theoretical computer science

by Michael Sipser[11]

During dinner one time with several MIT computer science theorists at Joyce Chen's Chinese restaurant in Cambridge, discussion turned to the future of theory. One of the group painted a rather bleak picture, because of a tight job market and a concern that few theoretical results bear upon practice. He said, "the handwriting is on the wall" for theory. I recall it quite distinctly because job interviews are memorable events, and that dinner occurred when I was interviewed for a position at MIT. I was a fresh Ph.D. then from Berkeley. It was 1979.

[10]Steklov Mathematical Institute, Vavilova 42, 117966, GSP–1, Moscow, RUSSIA. Supported by grant #96-01-01222 of the Russian Foundation for Fundamental Research.

[11]Mathematics Department, MIT. Supported by NSF Grant 9503322 CCR.

The future, viewed from 1979, turned out to be much brighter. Indeed, our field has been rich with beautiful, big ideas. We can take credit for major applications of theory, invented by theorists, that soon will affect everyone's daily life. Our currency has become high among scientists and mathematicians in other fields. Talented students have been electing to study theory and nearly all have found positions as theorists in academia or industry. Theory lives!

Today, I sometimes hear pessimistic views similar to those expressed at that 1979 dinner. Will our field remain intellectually alive? I believe the future of theory remains as bright now as it was then. The pace of discovery shows no sign of slowing. Students still find our questions challenging and wish to join us in solving them. The shortage of jobs and grants is an important problem. But any field that is interesting enough to continue to attract new people must eventually cope with such shortages, painful though they remain.

As to where theoretical computer science, or complexity theory, *should* go, I would be happy to see them continue on their current paths. Good research aims at depth, elegance, and practicality. By its nature, theory focuses on the first two, but even in complexity theory, many papers do not ignore the third. Some even manage to attain all three simultaneously, though such results are rare in any field. Various people argue that we ought to tilt our field toward practice, but I disagree. Competing with industry on its turf becomes increasingly harder as it grows richer. We must continue to reserve some of our academic effort for highly speculative and pure research that industrial companies will never pursue.

As to where theory *will* go, I'll make only one prediction. The next few years, say five to be conservative, will see another theoretical breakthrough as amazing and unexpected as were those previous. And to those still writing by hand on walls, I say an upgrade in technology—and thinking—is overdue.

References

[BTT92] Bartholdi, J.J., III; Tovey, C.A.; Trick, M. "How hard is it to control an election?" *Mathematical and Computer Modelling*, Aug.-Sept. 1992, vol.16, (no.8-9):27-40.

[BPT91] Buss, S.R.; Papadimitriou, C.H.; Tsitsiklis, J.N. "On the predictability of coupled automata: an allegory about chaos," *Complex Systems, Oct. 1991*, vol.5, (no.5):525-39. Also, *Proc. 1990 FOCS.*

[CPM94] Cowan, G.A.; Pines, D.; Meltzer, D. *Complexity: Metaphors, models, and reality*, Santa Fe, 1994.

[DP94] Deng, X.; Papadimitriou, C.H. "The complexity of solution concepts," *Mathematics of Operations Research, 19*, 2, pp. 257–266, 1994.

[GKPS95] Gogic, G.; Kautz, H.; Papadimitriou, C. H.; Selman, B. "The comparative linguistics of knowledge representation," *Proc. 1995 IJCAI.*

[HILL91] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. Construction of a pseudorandom generator from any one-way function. Technical Report 91-068, ICSI, Berkeley, 1991.

[HLY80] Honeyman, P.; Ladner, R.E.; Yannakakis, M. "Testing the universal instance assumption," *Information Processing Letters, 12* Feb. 1980, vol.10, (no.1):14-19.

[HRZ95] L. Hemaspaandra, A. Ramachandran, and M. Zimand. Worlds to die for. *SIGACT News*, 26(4):5–15, 1995.

[Imp96] R. Impagliazzo. Very strong one-way functions and pseudo-random generators exist relative to a random oracle. Manuscript, January 1996.

[Knu74] Knuth, D.E. "A terminological proposal," *SIGACT News, 6*, 1, 12–18, 1974.

[Pap96] Papadimitriou, C. H. "The complexity of knowledge representation," invited paper in the *Proc. 1996 Computational Complexity Conference.*

[PY94] Papadimitriou, C.H.; Yannakakis, M. "Complexity as bounded rationality," *Proc. 1994 STOC.*

[Zim96] M. Zimand. How to privatize random bits. Technical Report TR-616, University of Rochester, Department of Computer Science, Rochester, NY, April 1996.