The Computational Complexity Column

Eric Allender[1]

Rutgers University, Department of Computer Science
PO Box 1179, Piscataway, NJ 08855-1179 USA
allender@cs.rutgers.edu

I would like to say that I am thrilled to take over as editor of this column. But I am more "daunted" than "thrilled". Juris Hartmanis started this column back in issue # 31 of the Bulletin, and it has been an extremely successful run. I'm not eager to invite comparison.

In upcoming editions of this column, you should expect to find concise and readable accounts of some exciting developments and trends in complexity theory. Often, this may take the form of a survey or tutorial on some body of recent work.

In this issue of the Bulletin, however, I will survey some aspects of complexity theory that can hardly be considered "recent". Instead, I will focus on some fundamental truths relating complexity theory to practical computing. These truths deserve repeating, lest they become forgotten.

As a bonus, this edition of the Computational Complexity Column concludes with a report by Jack Lutz on the recent sixtieth birthday celebration for Ron Book. Book has had a profound and lasting influence on the field of complexity theory, and I am happy to be able to include a report on such an event in this inaugural column. Thanks, Jack!

1 Some Pointed Questions Concerning Asymptotic Lower Bounds

I am surprised at how often I encounter skepticism about the relevance of an asymptotic lower bound on the computational complexity of a function. Admittedly, the grounds for this skepticism may initially sound plausible, as the following "straw man" argument may illustrate:

> Let's assume that you've shown that function f requires time $2^{\Omega(n)}$ to compute, and also requires circuit size $2^{\Omega(n)}$. This still doesn't tell me that I won't be able to compute f for my application. I'm never going to need

[1]Supported in part by NSF grant CCR-9509603. This piece was written while the author was a visiting scholar at the Wilhelm-Schickard Institut für Informatik, Universität Tübingen, supported by DFG grant TU 7/117-1

> to deal with inputs with more than a billion bits, and your lower bound only says that there is *some* $\epsilon > 0$ such that the circuit size is at least $2^{\epsilon n}$. But if ϵ turns out to be one-billionth, then this tells me nothing. In fact, the set I'm interested in is *finite*, and all finite sets are regular, and hence they're recognizable in *linear* time. I'm afraid that the sort of asymptotic bounds provided by complexity theory miss the boat completely!

If you find this argument convincing, please read further.

It is true that lower bounds in complexity theory are usually stated in terms of asymptotic bounds (in the rare cases when actual lower bounds are able to be presented at all, that is). And it is equally true that an asymptotic lower bound, by itself, tells us absolutely nothing about the difficulty of computing a function on inputs of a specific given size. Yet, inside essentially every asymptotic lower bound in complexity theory, there hides a concrete statement about physical reality. It is best to illustrate this with an example. Here's my personal favorite:

In Larry Stockmeyer's Ph.D. thesis [St74], he shows that any circuit that takes as input a formula (in a certain logical system) with up to 616 symbols and produces as output a correct answer saying whether the formula is valid, requires at least 10^{123} gates.

It is important to note that there is nothing asymptotic about this theorem. Input sizes of 616 are well within the realm of real-world computation. In contrast, circuit sizes of 10^{123} are not within the realm of real-world computation, and they never will be.

Stockmeyer's result can be proved by first proving an *asymptotic* lower bound, consisting of two pieces:

1. The validity problem is shown to be hard for a complexity class under efficient reducibilities.

2. Diagonalization is used to show that some problem in the complexity class requires exponential-size circuits.

These two pieces are usually proved using the Turing machine model. Since Turing machines can accept any finite set in linear time (via the "linear speed-up theorem") something more must be done in order to get a concrete result about inputs of size 616. What has to be done is to look closely at the proof of the asymptotic result, and remove all mention of Turing machines, and carry out the same argument directly with circuits (which do not have a "linear speed-up theorem"). It is fairly clear that this can be done, since circuits can simulate Turing machines fairly efficiently. It's a lot more work to carry out a proof in this format, since you can't just conveniently forget the constants involved. However, at the end, you have an irrefutable statement about the impossibility of a certain transformation from input to output being computable in the real world.

It is important to note that essentially *any* asymptotic lower bound can be analyzed in this way to obtain concrete bounds for a given computing technology. However, it is almost never done. In fact, I am not aware of *any* other instance where concrete bounds of this sort were obtained. (Stockmeyer [Sto87] does mention the earlier work of Ehrenfeucht [Eh75] which has a similar flavor.) Why are results of this sort so rare?

The answer, of course, is that this approach first requires an asymptotic lower bound, and we don't have many of those, either! But this should be seen as additional motivation for obtaining asymptotic lower bounds. The question of whether or not NP has polynomial-sized circuits is not merely of mathematical and philosophical interest. If problems in NP really do require large circuits in the asymptotic sense, then it is almost certain that any proof of this fact will yield concrete bounds on instance sizes that interest us.

2 Complexity Theory and Empirical Observations

Complexity theory has another role to play in understanding physical reality. Curiously, it is not the role that one might originally have hoped for.

Is it possible to sort in linear time? Can a maximal matching in a graph be found in time $n \log n$? Just as the goal of algorithm design is to find faster programs, it once seemed reasonable to expect that a goal of complexity theory should be to show that certain algorithms are essentially optimal. That hasn't happened.

For a very few problems in P, there are lower bounds on so-called "restricted models of computation" (such as the model used in [EP95], for example, or the comparison-based sorting model, as another example). These bounds do provide an explanation for why some problems require lots of resources if a *certain kind* of algorithm is used. However, this type of argument has been applied to only a handful of computational problems, and it proves nothing about the complexity of these problems on unrestricted models of computation. There are some special subclasses of P (such as classes studied in the work of Buss and Goldsmith [BG93], or in the literature on Parameterized Complexity [HT96]) that help explain our current lack of a linear- or quadratic-time algorithm for a relatively small number of problems in P having a special structure. However, for the overwhelming majority of problems for which polynomial-time algorithms have been written, complexity theory currently offers no help in understanding if the running-time of a given algorithm is optimal.[2]

[2] Of course, there is also the possibility that some of these problems *have* no optimal algorithm [Bl67].

In contrast, complexity theory has been *extremely* successful in classifying the complexity of problems, by associating them with complexity classes. For the overwhelming majority of computational problems that have been considered in the computing literature, one can find a complexity class for which the problem is complete. In fact, in almost all cases, the problem is complete for the complexity class using a *very* restrictive notion of reducibility (such as AC^0 reducibility, or even projections; see e.g. [Imm87]).

Why is this significant?

First, I would like to argue that it is significant because it was unexpected. The initial shock, of course, came with the papers of [Co71, Ka72] showing that the notion of NP-completeness is wildly successful at explaining the seeming intractability of many problems. Then there followed some results with a similar flavor (such as [SM73, Jon75] and others), showing that some other problems were complete for larger and smaller complexity classes. Over the course of the years, more complexity classes have been defined and studied and been shown to capture the complexity of various important computational problems. Today, it seems safe to say that, when one is presented with a computational problem, the probability is quite high that it is complete for some complexity class in the existing literature, under AC^0 reducibility. (In fact, we now know that such problems are all *isomorphic* to the standard complete set for the complexity class, under depth-three AC^0 reductions [AAI97]. That is, the complete sets are all simple re-encodings of each other.)

Second, note that the current list of complexity classes is defined using a fairly small number of basic concepts such as nondeterminism, counting, circuits, and reducibility. The point is, that real-world computational problems have surprisingly close connections to these notions in complexity theory. Complexity theory gives us the vocabulary to classify and understand this aspect of our world.

What is the practical significance of the fact that a problem is complete for DLOG under AC^0 reductions? It tells us that the problem is not likely to have NC^1 circuits or polynomial-size bounded-width branching programs (as this would imply DLOG = NC^1 [Bar89]). But how significant is this for a practitioner? There aren't many people building bounded-width branching programs for problems, and since any problem in DLOG has size $n^{O(1)}$ circuits of depth $O(\log^2 n)$, an $\omega(\log n)$ lower bound is of questionable significance for the real world. The constants hiding in the "big Oh" are likely to be at least as important.

I would argue that the real practical significance lies not in any particular completeness result, but rather in the fact that, taken as an ensemble, completeness results present a coherent picture of the computational world. The classification of computational problems using complexity classes and reducibilities is so successful in describing observed reality, that it builds confidence in our belief that this classification is actually telling us something about the real world. (That is, since problems seem to announce themselves as being complete for, say, NP or DLOG, surely this is because NP and DLOG are not really the same thing!) In the absence of *proof* that complexity classes such as P and NP are different, we have come to rely on the utility of the scientific hypothesis "P $\neq$ NP" in explaining our observations of apparent intractability.

Seen in this light, it is clear why the breakthrough results of Immerman and Szelepcsényi [Imm88, Sze88] were shocking at the same time that they were exhilarating. It had seemed that many problems naturally classified themselves as being complete for exactly one of {NLOG, coNLOG}. There seemed to be as much empirical grounds for the belief that these classes were distinct as for much of the rest of the framework provided by complexity theory. (There were smaller tremors provided by the earlier discovery that the alternating logspace hierarchy collapses [JKL89]; some computational problems had been thought to live on different levels of that hierarchy [RY86].) If the "empirical evidence" that NLOG $\neq$ coNLOG was really just an illusion, how much can we trust the rest of the classification given by complexity classes? We need to understand these classes much better!

Let me summarize the points I'm trying to make.

1. We are presented with a bewildering universe. Many problems appear to be difficult to compute, but we have not been able to prove that this is the case.

2. Fortunately, complexity theory does provide the necessary tools to impose order on this chaos. Using the notions of reducibility and completeness for complexity classes, the computational world we observe can be understood as consisting of a relatively small number of equivalent problems, appearing in different guises.[3] Complexity classes are the fundamental objects of study in complexity theory.

3. The characterizations of complexity classes in terms of abstract models of computation provide an intuitive "explanation" for perceived differences in complexity. Sometimes this intuition has been misleading; some of the most important theorems in complexity theory are those showing that what had appeared to be distinct classes in fact coincide. Research in complexity theory seeks to refine our intuition, and arrive at an accurate understanding of this aspect of reality.

4. The theory of complexity classes is based on Turing machines and asymptotic bounds. This makes for a more elegant theory, but by itself it cannot produce

[3] Of course, I am guilty of oversimplification here. Complexity theory also works with the notions of average-case complexity and complexity of approximation. Although complexity classes and reducibilities have been some of our best tools for understanding these aspects, too, the "compartmentalization" provided by equivalence under interreducibility is not quite as tidy when considering these aspects.

concrete bounds about the difficulty of computing transformations on inputs of a given size. Fortunately, the theory is closely enough connected to reality, so that essentially any asymptotic lower bound can be reworked to provide a concrete bound for a given computing technology.

The good news is that we have actually learned quite a bit about many complexity classes in the last few years. Watch this space for continuing updates.

Acknowledgments

Observations similar to the ones expressed above have found their way into some chapters I recently collaborated on with Michael Loui and Ken Regan [ALR97], and my interactions with them certainly influenced what appears above. Related ideas are discussed in [Al96]. V. Vinay and Jack Lutz have each discussed the role of complexity theory in formulating hypotheses, both with me and in print [Vi94, Lu93]. I gratefully acknowledge their influence. Thanks also to Rolf Niedermeier and Jacobo Torán for their comments on a preliminary draft.

References

[AAI97] M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich. Reducing the complexity of reductions. To appear in *Proc. 29th STOC*, 1997.

[ALR97] E. Allender, M. Loui, and K. Regan. Complexity Classes, Reducibility and Completeness, and Other Complexity Classes and Measures. Three chapters written for the *Handbook on Algorithms and the Theory of Computation*, ed. M. Atallah, CRC Press (to appear).

[Al96] E. Allender. Circuit complexity before the dawn of the new millennium. In *Proc. 16th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS '96)*, volume 1180 of *Lect. Notes in Comp. Sci.*, pages 1–18. Springer Verlag, 1996.

[Bar89] D. A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

[Bl67] M. Blum. A machine-independent theory of the complexity of recursive functions. *J. Assn. Comp. Mach.*, 14:322–336, 1967.

[BG93] J. Buss and J. Goldsmith. Nondeterminism within P. *SIAM Journal on Computing* 22:560–572, 1993.

[Co71] S. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Annual ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[EP95] J. Edmonds and C. Poon. A nearly optimal time-space lower bound for directed *st*-connectivity on the NNJAG model. In *Proc. 27th STOC*, 1995, pp. 147–156.

[Eh75] A. Ehrenfeucht. Practical decidability. *Journal of Computer and System Sciences*, 11:392–396, 1975.

[HT96] M. T. Hallett and H. T. Wareham. Parameterized Complexity Home Page. `http://www-csc.uvic.ca/home/harold/W_hier/W_hier.html`

[Imm87] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16:760–778, 1987.

[Imm88] N. Immerman. Nondeterministic space is closed under complement. *SIAM Journal on Computing*, 17:935–938, 1988.

[JKL89] B. Jenner, B. Kirsig, and K.-J. Lange. The Logarithmic Alternation Hierarchy Collapses: $\mathrm{A}\Sigma_2^{\mathcal{L}} = \mathrm{A}\Pi_2^{\mathcal{L}}$. *Information and Computation* 80:269–288 1989.

[Jon75] N. D. Jones. Space bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11:68–85, 1975.

[Ka72] R. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, 1972.

[Lu93] J. Lutz. The quantitative structure of exponential time. In *Proc. 8th IEEE Conference on Structure in Complexity Theory*, 1993, pp. 158–175.

[RY86] L. Rosier and H. Yen. Logspace hierarchies, polynomial time, and the complexity of fairness problems concerning ω-machines. In *Proc. 3rd Symposium on Theoretical Aspects of Computer Science (STACS '86)*, volume 210 of *Lect. Notes in Comp. Sci.*, pages 306–320. Springer Verlag, 1986.

[St74] L. Stockmeyer. The complexity of decision problems in automata theory and logic. Technical Report MAC-TR-133, Project MAC, M.I.T., Cambridge, Mass., 1974.

[Sto87] L. Stockmeyer. Classifying the computational complexity of problems. *J. Symb. Logic*, 52:1–43, 1987.

[SM73] L. Stockmeyer and A. Meyer. Word problems requiring exponential time: preliminary report. In *Proc. 5th STOC*, 1973, pp. 1–9.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Vi94] V. Vinay. Rudiments of complexity theory for scientists and engineers. *Sadhana*, Vol. 19, part 6, Dec. 1994, pp. 985–994.

Special Supplement to the Computational Complexity Column

Report on the
Workshop on Languages, Algorithms, and Complexity
University of Minnesota
Minneapolis, MN, USA
April 12, 1997
Celebrating the Sixtieth Birthday of Professor Ronald V. Book

On April 12, 1997, theoretical computer scientists from Europe, Asia, and America gathered at the University of Minnesota to commemorate the scientific career of Professor Ronald V. Book on the occasion of his sixtieth birthday.

Professor Book has served on the faculties of Harvard University, Yale University, and the University of California at Santa Barbara, where he is now a Professor of Mathematics. He is the author or co-author of more than 150 scientific publications. He has supervised 15 Ph.D. students and 14 postdoctoral research fellows, and has informally helped many more young scientists at the outsets of their scientific careers. In 1990, he was awarded the Alexander von Humboldt research prize for senior U.S. scientists.

The workshop featured fifteen talks spanning the three main areas of Professor Book's research: formal languages, computational complexity, and algorithms. There were also an informal lunch, with plenty of time for discussion, and an evening banquet, followed by numerous testimonials to Professor Book's lasting impact on his colleagues and his field. Taken as a whole, the workshop was truly a celebration of a career of great influence, both through research and through tireless dedication to students and young scientists.

The workshop organizers, Ding-Zhu Du, Ker-I Ko, and Klaus Wagner, did a superb job on all aspects of this occasion. The workshop program is summarized below. Most of the talks are available as chapters in the book, *Advances in Algorithms, Languages, and Complexity*, edited by Ding-Zhu Du and Ker-I-Ko, and published by Kluwer Academic Publishers (1997).

We all enjoyed the opportunity to honor Ron and his wife, Celia, in Minneapolis. We wish Ron many more happy birthdays!

Jack Lutz
Iowa State University

Workshop Program

(Speakers marked with *)

Session 1 (Chair: Ker-I Ko)

8:30-9:15 Juris Hartmanis:
An Assessment of Computer Science and Its Theory

Session 2 (Chair: Ker-I Ko)

9:35-10:00 Jose L. Balcazar, Ricard Gavalda, Osamu Watanabe*:
Coding Complexity: The Computational Complexity
of Succinct Descriptions

10:00-10:25 Johannes Koebler, Uwe Schoening*: High Sets for NP

10:25-10:50 Amy K. Lorentz, Jack H. Lutz*:
Genericity and Randomness over Feasible Probability
Measure

10:50-11:15 Dieter van Melkebeek, Mitsunori Ogihara*:
Sparse Hard Sets for P

11:15-11:40 Heribert Vollmer, Klaus W. Wagner*:
Measure One Results in Computational Complexity Theory

11:40-12:05 Jie Wang*: Average-Case Intractable NP Problems

12:05-1:35 Lunch

Session 3 (Chair: Klaus W. Wagner)

1:35-2:00 Robert McNaughton*: Book's Impact on the Theory of Thue Systems

2:00-2:25 Franz-J. Brandenburg*:
Homomorphic Representations of Certain Classes of Language

2:25-2:50 Jose L. Balcazar, Josep Diaz, Ricard Gavalda*, Osamu Watanabe:
Algorithms for Learning Finite Automata from Queries:
A Unified View

2:50-3:15 F. Otto*, Y. Kobayashi:

	Properties of Monoids That Are Presented by Finite Convergent String-Rewriting Systems - A Survey
Session 4	(Chair: Ding-Zhu Du)
3:35-4:00	Eljas Soisalon-Soininen*, Peter Widmayer: Relaxed Balancing in Search Trees
4:00-4:25	Helmut Jurgensen, Kai Salomaa*: Block-Synchronization Context-Free Grammars
4:25-4:50	Pekka Orponen*: A Survey of Continuous-Time Computation Theory
4:50-5:15	Jin-yi Cai: Improving Ajtai's Connection of Average Case and Worst Case Complexity for Lattice Problems
6:30	Banquet