SOME POINTED QUESTIONS CONCERNING ASYMPTOTIC LOWER BOUNDS, AND NEWS FROM THE ISOMORPHISM FRONT

ERIC ALLENDER

Rutgers University, Department of Computer Science, Piscataway, NJ 08855-8019 USA

E-mail: allender@cs.rutgers.edu

For this volume, I have combined columns I wrote for the June, 1997 and October, 1998 Bulletin. As both are brief, and the second column refers to the first, this seemed appropriate. The first column discusses the general question of whether asymptotic lower bounds are of any practical use. (My answer: They are.) The second column is in the form of a dialogue between "Q" and "A" in the format used so successfully by Yuri Gurevich in many of his columns for the Bulletin. The reader will kindly pardon my attempts to inject a bit of humorous style (*á la* Dave Barry) into this discussion of recent progress on the question of whether or not all NP-complete sets are isomorphic. The serious message of this second part of the article: there are some very practical reasons for being interested in the very abstract notion of isomorphism of complete sets.

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 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, [43] 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 worth noting that Stockmeyer's analysis can be carried out to show that the validity problem requires huge circuit size, even if *quantum* circuits are allowed. The constants will be slightly different, but the flavor of the theorem remains unchanged.

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 [44] does mention the earlier work of Ehrenfeucht [16] 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 by Edmonds *et al* [15] 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, [13] or in the literature on Parameterized Complexity [22]) 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.[a]

In contrast, complexity theory has been *extremely* successful in classifying the complexity of problems, by associating them with complexity classes. For

[a] Of course, there is also the possibility that some of these problems *have* no optimal algorithm. [10]

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). [23b]

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 Cook and Karp [14,29] 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 Meyer and Stockmeyer and Jones [45,26] and others), showing that some other problems are 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, as is discussed in more detail later on in this article, we now know that such problems are all *isomorphic* to the standard complete set for the complexity class, under depth-three AC^0 reductions. [2] 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 $NC^1 = DLOG$ [7]). 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 partic-

[b] Reminder for the reader: AC^0 refers to polynomial-size circuits of unbounded-fan-in AND and OR gates, having depth $O(1)$.

ular 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 [24,46] 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; [25] some computational problems had been thought to live on different levels of that hierarchy. [41]) 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.[c] Complexity classes are the fundamental objects of study in complexity theory.

[c]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.

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 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. The following sections give more detailed information about one particular arena where progress has been made: isomorphisms between complete sets. This discussion takes the form of a Question and Answer dialogue.

3 News from the Isomorphism Front

3.1 The Interview Begins: Questions are Raised about Factorization

Q: In the preceding sections of this article, you gave a very persuasive and highly readable account, arguing that the study of *complexity classes* offers the best hope for understanding the computational complexity of important computational problems.

A: You are too kind.

Q: Yes, I *was* being too kind. I'll try to make up for it now. Honestly, I don't think that the theory of complexity classes has *anything* to say about some of my favorite computational problems. Consider the list:

- Factorization
- Graph Isomorphism
- Graph Automorphism
- Primality Testing

- Greatest Common Divisor
- Perfect Matching

A: I see your point. None of these problems is known to be complete for any complexity class. But this list simply supports my point: the very fact that this list is so short is one of the unexpected triumphs of complexity theory.

Q: But it is *not* short! There are infinitely many problems that are not complete for any reasonable complexity class. Ladner's construction [35] shows how to construct such sets.

A: True, but the "computational problems" given by Ladner's lovely construction are very artificial. Empirically, the *overwhelming majority* of the computational problems that people actually want to solve can be shown fairly easily to be complete for one of a small number of natural complexity classes, even under very restrictive notions of reducibility, such as $\leq_m^{AC^0}$ reductions.

Q: Let's get back to my list of problems. Certainly there are some very important problems on this list. And it seems that computational complexity can't say *anything* about *any* of these problems. For instance, are any of these problems even *hard* for any complexity class?

A: This is a good question. First, let's remind the reader what is meant by being "hard" for a complexity class. A is *hard for* $\mathcal{C}$ (where $\mathcal{C}$ is a set of languages) if for every $B \in \mathcal{C}$, there is a function f computed by AC^0 circuits, such that $x \in B$ if and only if $f(x) \in A$. This is a complexity-theoretic notion of a lower bound on the complexity of A. Showing that A is in some complexity class provides a notion of upper bound. If A is complete for $\mathcal{C}$, then these upper and lower bounds match.

Now back to your question. We do know something about the Graph Isomorphism problem. It was recently shown that Graph Isomorphism is hard for the class DET of problems that are NC^1-reducible to the determinant (which is a class containing nondeterministic and probabilistic logspace, among others). [47] As regards upper bounds, Graph Isomorphism is in NP $\cap$ co-AM $\cap$ LWPP – and I'm probably leaving some classes out of this list. For more about Graph Isomorphism, consult the text by Köbler, Schöning, and Torán. [33]

The situation is quite a bit better for the perfect matching problem. It has long been known that matching is hard for NLOG, [34] and recently an upper bound was presented that in some ways is fairly "close" to the lower bound (although I don't want to make this precise here because it would involve a digression about notions of circuit uniformity, etc). [6]

However, for the rest of the problems on your list, I'm afraid that there aren't any theorems regarding hardness or completeness for any of these problems. ... At least not yet ...

Q: Are you hinting that you think someone *might* prove that, say, factorization is complete for some natural complexity class?

A: Absolutely not! In fact, I think that factorization is *not* complete for any such class.

Q: Why not?

A: Well ..., it doesn't have the right "shape". It's structure just isn't right. It doesn't allow the type of coding that you need in order to build a reduction.

Q: Those are some of the most pathetic, *least* satisfactory answers I can imagine! Frankly, I'm quite disappointed. Is that the best you can come up with?

A: *Sigh* ... I'm afraid the best I can do is to present some plausible conjectures. Proving any one of these conjectures would prove that factorization is not complete for any "reasonable" complexity class. Although I can't prove any of these conjectures right now, I think that they might be tractable, and I also hope that they'll make for interesting reading. In fact, **presenting these conjectures is the entire point of this article**. First, however, we will need to make a digression, while we discuss some recent progress on the isomorphism conjecture.

Q: I'm always interested in hearing about progress!

4 Progress on the Isomorphism Conjecture

A: The Berman-Hartmanis Conjecture [9] states that all NP-complete sets are isomorphic under bijections computable and invertible in polynomial time. (Here, an isomorphism is nothing more than a bijection; there is no other "structure" that is being preserved. However, this is an abstract way of capturing the intuitive notion that two different "encodings" of a set are still really the "same" set in some sense. For instance, the SAT problem (the set of satisfiable Boolean formulae) can be encoded using round parentheses or square brackets, or with variables $\{x_1, x_2, \ldots\}$ or $\{\mathrm{P}_1, \mathrm{P}_2, \ldots\}$. Such encoding choices are trivial and unimportant from the standpoint of complexity theory. One way to make this precise is to group "equivalent encodings" of a problem into equivalence classes, where an "encoding" can be any "easily-computable" bijection on $\{0,1\}^*$.

Q: I see. So you get one equivalence class for SAT, another class for the Clique problem, one for 3-colorability, and so on.

A: Right – except that all the problems you mentioned are in the *same* class.

Q: Really? Clique is an encoding of SAT?

A: Yes. Berman and Hartmanis [9] gave some simple sufficient conditions for two sets to be isomorphic, and with these conditions, it is easy to see that all of the NP-complete problems in Garey and Johnson [20] are isomorphic.

Q: What about smaller complexity classes, such as P and NLOG?

A: Hartmanis [21] studied analogous conjectures for these classes, using logspace reducibility (and logspace isomorphisms), instead of poly-time. Once again, all of the standard complete sets are easily seen to be isomorphic. Of course, logspace reductions are too powerful to investigate the many interesting subclasses of DLOG (such as ACC^0, TC^0, and NC^1). For these classes, AC^0 reducibility is the most natural notion of reducibility to use, [7,12] and once again the "standard" complete sets are all seen to be isomorphic. [4d]

Q: So, I guess that, with so much written about the isomorphism conjectures, most complexity theoreticians believe that the conjectures are true?

A: Far from it. In fact, probably most complexity theoreticians believe that there is a one-way function f (one that is easy to compute but very hard to invert) such that f(SAT) is *not* isomorphic to SAT. Some really interesting work in this direction has been done ... [28,30,32]

Q: Let me interrupt. You'll never meet your deadline if you are going to survey the entire field. Are there any surveys you can suggest?

A: Yes. Two very nice ones that spring to mind are by Kurtz *et al* [31] and Young. [49] Some exciting developments are too recent to be mentioned in the surveys. [18,27,40]

Q: What does this have to do with factorization?

A: I'm getting to that. In spite of the general feeling that the original Berman-Hartmanis conjecture is probably false, it was shown recently that the Berman-Hartmanis conjecture for AC^0 reductions is actually *true*. [3]

Q: Please state this more clearly.

A: For any "reasonable" complexity class, all of the sets that are complete under AC^0 reductions are isomorphic under bijections computable and invertible by AC^0 circuits. (In fact, they are computable and invertible by *depth three* AC^0 circuits!)

Q: Is depth three optimal?

A: Yes. [2]

Q: What is a "reasonable" complexity class?

[d] Actually much stronger results are proved in Allender *et al*, [4] but it would require another digression to present that material.

A: Any class that is closed under many-one reductions computable in TC^0. In particular, NP, P, NC^1, DLOG, BPP, and just about any other complexity class you've ever heard of is "reasonable" in this sense (except for very small classes, such as AC^0 and ACC^0).

Q: So, in light of your earlier comments about $f(\text{SAT})$, when f is a one-way function, does this mean that there are no one-way functions computable in AC^0?

A: Quite the contrary. It was shown by Nisan [38] that there are functions f computable in AC^0 that are *very* one-way with respect to depth-three AC^0. Nonetheless, even for such a "random-looking" function f, $f(\text{SAT})$ is just a "trivial" re-encoding of SAT (i.e., it is depth-three AC^0-isomorphic to SAT).

Q: Are there some annoying details about circuit uniformity conditions that you're ignoring in this article?

A: Yes.

Q: This seems to be quite a strong result! Complexity classes are the fundamental objects of study in complexity theory, because the natural computational problems that we are interested in tend to cluster into classes of complete sets for complexity classes. The result of Agrawal *et al* [3] shows that, not only are the complete sets interreducible, but in fact they are isomorphic in a *very* restrictive sense.

A: Remember, this is only true for sets that are complete under AC^0 reducibility.

Q: Yes, but you said that all of the sets that are complete under other reducibilities are also complete under AC^0 reducibilities.

A: If that were true, then we'd have a proof that $P \neq NP$! I merely said that all of the "natural" examples of complete sets (for instance, all of the complete sets considered in Garey and Johnson [20] and Greenlaw *et al* [19]) are complete under AC^0 reducibilities. Actually, it is shown by Agrawal *et al* [2] that there are sets complete for NP under poly-time reducibility, but *not* under AC^0 reducibility.

Q: What does this set look like?

A: It is an encoding of SAT, using a certain kind of error-correcting code. This set (let's call it SAT$'$) is clearly poly-time isomorphic to SAT, but it is not NP-complete under AC^0 reductions. In fact, there are regular sets (such as the PARITY language) that are not reducible to SAT$'$ under AC^0 reductions.

Q: Isn't this a counterexample to your claim that all "natural" NP-complete sets are complete under AC^0 reducibility?

A: This depends on your definition of "natural". I think that not many people would say that this is the "natural" way to encode SAT.

Q: We'll let the readers of this column decide what they think of this

semantic hair-splitting. But let's get back to the factorization problem.

A: Thank you!

5 Factoring and Cylinders

A: First let's be very precise about what we mean by "the factorization problem". Let us define it to be the following language:

FACT $= \{(x, i, b) :$ the ith bit of the prime factorization of x is $b\}$

(where the prime factorization is presented as $p_1^{e_1}, \ldots, p_k^{e_k}$, where each exponent $e_i > 0$, and $p_i < p_{i+1}$, so that each number has a unique prime factorization).

Q: What are the complexity-theoretic "upper and lower bounds" on the complexity of FACT?

A: FACT is in NP∩coNP (and in fact it is even in UP∩coUP [17]). Thus FACT is not NP-complete unless NP = coNP = UP = coUP. On the other hand, FACT is not known to be hard for *any* reasonable complexity class under AC^0 reducibility. It is not even known to be hard for $AC^0(\oplus)$ (the class of languages accepted by constant-depth, polynomial-size circuits of AND, OR, and PARITY gates)!

Q: Wow! Are you saying that it isn't even known if FACT is in $AC^0(\oplus)$ or not?

A: That's *not* what I said! Actually, it was shown by Boppana and Lagarias [11] that the number of 1's in a string $x = x_1 \ldots x_n$ is a multiple of 3 if and only if the number whose binary representation is $x_1 0 x_2 0 \ldots 0 x_n$ is a multiple of 3. Thus, if FACT were in $AC^0(\oplus)$, then certainly we could solve the Mod3 problem with $AC^0(\oplus)$ circuits, in contradiction to Smolensky's lower bound. [42] In fact this shows that FACT is not in $AC^0(\text{Mod}p)$ for any prime $p \neq 3$.

Q: So, is FACT in $AC^0(\text{Mod } 3)$?

A: No. If you look at the finite automaton for the set of binary strings that are a multiple of 5, you will see that the number of 1's in a string $x = x_1 \ldots x_n$ is a multiple of 5 if and only if the number whose binary representation is $000x_1000x_2000 \ldots 000x_n$ is a multiple of 5. Thus, if FACT were in $AC^0(\text{Mod3})$, then certainly we could solve the Mod5 problem with $AC^0(\text{Mod3})$ circuits, again in contradiction to Smolensky. [42] Thus we have that FACT is not in $AC^0(\text{Mod}p)$ for any prime p.

Q: So, we know that FACT is not *in* $AC^0(\text{Mod}p)$ for any prime p, but we don't know if FACT is *hard* for any such class. How about $AC^0(\text{Mod6})$?

A: It remains an open question if there is anything in $\text{NTIME}(2^n)$ that does not have linear-size, depth-three circuits of Mod6 gates.

Q: Thanks for reminding me just how little we know. What does this have to do with the question of whether FACT is complete for some complexity class?

A: I'm getting to that. If FACT were complete for some reasonable complexity class, then, by Agrawal *et al* [3], FACT would be isomorphic to FACT$\times\{0,1\}^*$ (since FACT$\times\{0,1\}^*$ would clearly also be complete for the same complexity class).

Q: This reminds me of something from my computability textbook.[39] A set A that is recursively isomorphic to $A \times \{0,1\}^*$ is called a "cylinder".

A: Yes. This same terminology was imported to complexity theory by Mahaney and Young. [37]

Q: Are we now ready for you to state your conjecture?

A: Yes.

> **Conjecture:** There is no isomorphism computable and invertible in depth-three AC^0 mapping FACT$\times\{0,1\}^*$ onto FACT.

That is; FACT is not a (depth-three) AC^0 cylinder. If this conjecture is true, then factoring is not complete for any reasonable complexity class under AC^0 reductions.

Q: Isn't this conjecture just another example of an impossibly difficult question in complexity theory? Wouldn't it imply NLOG $\neq$ P, or $TC^0 \neq$ NP, or something?

A: I don't think so. If I could show that A is not an AC^0 cylinder, where A is some set in NP that is hard for TC^0 (for instance), then I would know that TC^0 is contained in the class of languages that are reducible to A, which is properly contained in NP (since all NP-complete sets *are* AC^0 cylinders), and thus I could conclude that TC^0 is not equal to NP. However, since FACT is *not* known (or even believed) to be hard for TC^0, this obstacle is removed.

Q: What intuition do you have about why the conjecture might be true?

A: First, note that it is not too hard to show that FACT is a P-cylinder. That is, there *is* an isomorphism computable and invertible in polynomial time between FACT and FACT$\times\{0,1\}^*$. (Sketch of how to compute such an isomorphism: It is sufficient to give a length-increasing, invertible reduction from FACT$\times\{0,1\}^*$ to FACT; this yields an isomorphism.[9] Given (x,i,b) and y, we need to find some (x',i',b') such that $(x,i,b) \in F$ if and only if $(x',i',b') \in F$. We can directly compute the largest exponent e_j for each of the first $|y|$ primes, such that $p_j^{e_j}$ to that exponent divides x. Then, by doing some simple encoding using the exponents of the first $|y|$ primes, we can build a new number, whose prime factorization "encodes" the prime factorization of x, along with the bits of y. If we are careful about how we do this, then the

bits of the prime factorization of x that deal with large primes are essentially unchanged (and thus i will be mapped to $i' = i + j$ for some easy-to-compute j). The details are straightforward but tedious.)

All of my attempts to build an isomorphism between FACT and FACT$\times\{0,1\}^*$ involve doing quite a bit of multiplication and division – which is impossible for an AC^0 circuit. One approach to try to prove the conjecture would be to show that *any* circuit computing such an isomorphism would be usable as an oracle to compute multiplication, or at least to compute parity infinitely often.

Q: Still, this seems like an ambitious project.

A: Perhaps. Certainly, showing that two sets are *not* isomorphic often seems to be quite difficult! (Note, however, that Agrawal *et al* [2] do prove results of this form. The sets SAT and SAT$'$ are not AC^0 isomorphic.)

Another approach would be to try to show that there is no length-increasing one-one function f in NC^0 reducing FACT$\times\{0,1\}^*$ to FACT. (Note that the "one-one" condition is crucial here. Otherwise, in NC^0 one could map (x, i, b, z) to $(x0^{|z|}, i', b)$ for appropriate (easy-to-compute) i'.) The results of Agrawal *et al* [3] can again be used to show that this implies that FACT is not complete.

Q: Do these observations apply to problems other than factorization?

A: They certainly do. I would guess that primality testing and computation of the greatest common divisor also fail to be complete for any reasonable complexity class, for the same reason.

It would also be worth trying to say something concrete about the multiplication problem. That is, consider the set MULT $= \{(x, y, i, b) :$ the ith bit of the binary representation of $x \cdot y$ is $b\}$. MULT is certainly an important set, and in some sense its complexity is fairly well understood. It is in TC^0, and it is complete for TC^0 under AC^0-*Turing* reducibility. Since it is still an important open question if TC^0 is equal to NC^1, I believe it is still an open question whether MULT is complete for NC^1 under AC^0 many-one reducibility. I think it is quite possible that one might be able to show that MULT is *not* complete for any reasonable complexity class (under AC^0 many-one reducibility).

Q: Wouldn't this allow one to prove that TC^0 is not equal to NC^1?

A: Not that I see. Note in this regard that it is already known that the MAJORITY function is not complete for NC^1 under AC^0 many-one reducibility. (That is because there is no AC^0 many-one reduction from the PARITY language to MAJORITY. [8]) MAJORITY and MULT have quite similar complexity, in that both are complete for TC^0 under AC^0-Turing reducibility.

Q: Is this the end of the article?

A: Almost. I just want to comment that we really do agree on one point.

Q: What is that?

A: Currently, complexity theory has no credible approach toward trying to show that factorization is difficult to compute. The only real evidence of intractability that complexity theory offers currently is based on reducibility and completeness. Factorization seems to fall outside that framework. With luck, however, it might be possible to *prove* that this is the case.

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, [5] and my interactions with them certainly influenced what appears above. Related ideas are discussed another paper of mine. [1] V. Vinay and Jack Lutz have each discussed the role of complexity theory in formulating hypotheses, both with me and in print. [48,36] I gratefully acknowledge their influence. Thanks also to Rolf Niedermeier and Jacobo Torán for their comments on a preliminary draft of some of this work. This work was supported in part by NSF grants CCR-9509603 and CCR-9734918. Some of this work was written as a visiting scholar at the Wilhelm-Schickard Institut für Informatik, Universität Tübingen, supported by DFG grant TU 7/117-1.

References

1. 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.*, pp. 1–18 (Springer-Verlag, New York, 1996).
2. M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich, Reducing the complexity of reductions, in *Proc. 29th ACM Symposium on Theory of Computing (STOC)*, pp. 730–738 (1997).
3. M. Agrawal, E. Allender and S. Rudich, Reductions in Circuit Complexity: An Isomorphism Theorem and a Gap Theorem, *J. Computer and System Sciences* **57**, 127 (1998).
4. E. Allender, J. Balcázar, and N. Immerman, A first-order isomorphism theorem, *SIAM J. Comput.* **26**, 557 (1997).
5. 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, Boca Raton, 1999).

6. E. Allender, K. Reinhardt and S. Zhou, Isolation, Matching, and Counting: Uniform and Nonuniform Upper Bounds, *J. Computer and System Sciences* **59**, 164 (1999).
7. D. A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Computer and System Sciences* **38**, 150 (1989).
8. R. Beigel, When do extra majority gates help? Polylog(n) majority gates are equivalent to one, *Computational Complexity* **4**, 297 (1984).
9. L. Berman and J. Hartmanis, On isomorphism and density of NP and other complete sets, *SIAM J. Comput.* **6**, 305 (1977).
10. M. Blum. A machine-independent theory of the complexity of recursive functions *J. ACM* **14**, 322 (1967).
11. Ravi Boppana and Jeff Lagarias, One-Way Functions and Circuit Complexity, *Information and Computation* **74**, 226 (1987).
12. S. Buss, The Boolean formula value problem is in ALOGTIME, in *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, pp. 123–131 (1987).
13. J. Buss and J. Goldsmith, Nondeterminism within P. *SIAM J. Comput.* **22**, 560 (1993).
14. S. Cook, The complexity of theorem-proving procedures, in *Proc. 3rd Annual ACM Symposium on the Theory of Computing (STOC)*, pp. 151–158 (1971).
15. J. Edmonds, C. Poon, and D. Achlioptas, Tight lower bounds for *st*-connectivity on the NNJAG model, *SIAM J. Comput.* **28**, 2257 (1999).
16. A. Ehrenfeucht, Practical decidability, *J. Computer and System Sciences* **11**, 392 (1975).
17. M. Fellows and N. Koblitz, Self-witnessing polynomial-time complexity and prime factorization, in *Proc. 7th IEEE Structure in Complexity Theory Conference*, pp. 107–110 (1992).
18. S. Fenner, L. Fortnow, and S. Kurtz, The isomorphism conjecture holds relative to an oracle, *SIAM J. Comput.* **25**, 193 (1996).
19. R. Greenlaw, J. Hoover, and W.L. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory*, (Oxford University Press, New York, 1995).
20. M. R. Garey and David S. Johnson, *Computers and Intractability,* (W. H. Freeman & Co., San Francisco, 1979).
21. J. Hartmanis, On the logtape isomorphism of complete sets, *Theoretical Computer Science* **7**, 273 (1978).
22. M. T. Hallett and H. T. Wareham, Parameterized Complexity Home Page, `http://www.cs.mun.ca/~harold/W_hier`.

23. N. Immerman, Languages that capture complexity classes, *SIAM J. Comput.* **16**, 760 (1987).
24. N. Immerman, Nondeterministic space is closed under complement, *SIAM J. Comput.* **17**, 935 (1988).
25. B. Jenner, B. Kirsig, and K.-J. Lange, The Logarithmic Alternation Hierarchy Collapses: $A\Sigma_2^{\mathcal{L}} = A\Pi_2^{\mathcal{L}}$, *Information and Computation* **80**, 269 (1989).
26. N. D. Jones, Space bounded reducibility among combinatorial problems, *J. Computer and System Sciences* **11**, 68 (1975).
27. D. Joseph, R. Pruim, and P. Young, Collapsing degrees in subexponential time, in *Proc. 9th Structure in Complexity Theory Conference* pp. 367–382 (1994).
28. D. Joseph and P. Young, Some remarks on witness functions for non-polynomial and non-complete sets in NP, *Theoretical Computer Science* **39**, 225 (1985).
29. R. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, ed. R.E. Miller and J.W. Thatcher (Plenum Press, New York , 1972).
30. K.-I Ko, T. J. Long, and D.-Z. Du, On one-way functions and polynomial-time isomorphisms, *Theoretical Computer Science* **47**, 263 (1986).
31. S. Kurtz, S. Mahaney, and J. Royer, The structure of complete degrees, in *Complexity Theory Retrospective* ed. A. Selman (Springer-Verlag, New York, 1990).
32. S. Kurtz, S. Mahaney, and J. Royer, The isomorphism conjecture fails relative to a random oracle, *J. ACM* **42**, 401 (1995).
33. J. Köbler, U. Schöning, and J. Torán, The Graph Isomorphism Problem; Its Structural Complexity, (Birkhäuser, Boston, 1993).
34. R. Karp and E. Upfal and A. Wigderson, Constructing a perfect matching is in random NC, *Combinatorica* **6**, 35 (1986).
35. R. Ladner, On the structure of polynomial-time reducibility, *J. ACM* **22**, 155 (1975).
36. J. Lutz, The quantitative structure of exponential time, in *Proc. 8th IEEE Structure in Complexity Theory Conference*, pp. 158–175 (1993).
37. S. Mahaney and P. Young, Reductions among polynomial isomorphism types, *Theoretical Computer Science* **39**, 207 (1985).
38. N. Nisan, *Using Hard Problems to Create Pseudorandom Generators*, (MIT Press, Cambridge, 1992).
39. H. Rogers, *Theory of Recursive Functions and Effective Computability*, (McGraw-Hill, New York, 1967).

40. J. Rogers, The isomorphism conjecture holds and one-way functions exist relative to an oracle, *J. Computer and System Sciences* **54**, 412 (1997).
41. 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.*, pp. 306–320 (Springer-Verlag, New York, 1986).
42. R. Smolensky, Algebraic methods in the theory of lower bounds for Boolean circuit complexity, in *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, pp. 77–82 (1987).
43. 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).
44. L. Stockmeyer, Classifying the computational complexity of problems, *J. Symb. Logic* **52**, 1 (1987).
45. L. Stockmeyer and A. Meyer, Word problems requiring exponential time: preliminary report, in *Proc. 5th ACM Symposium on Theory of Computing (STOC)*, pp. 1–9 (1973).
46. R. Szelepcsényi, The method of forced enumeration for nondeterministic automata, *Acta Informatica* **26**, 279 (1988).
47. J. Torán, On the hardness of graph isomorphism, submitted for publication.
48. V. Vinay, Rudiments of complexity theory for scientists and engineers, *Sadhana* **19**, 985 (1994).
49. Paul Young, Juris Hartmanis: Fundamental Contributions to Isomorphism Problems, in *Complexity Theory Retrospective,* ed. A. Selman (Springer-Verlag, New York, 1990).