The Computational Complexity Column

by

Vikraman Arvind

Institute of Mathematical Sciences, CIT Campus, Taramani
Chennai 600113, India
http://www.imsc.res.in/~arvind

Eric Allender recently completed sixty years. In a research career spanning over thirty-five years, and still flourishing, Eric has been a leader in the field of computational complexity. His contributions are to many different aspects, including circuit complexity, space-bounded complexity classes, Kolmogorov complexity, and, broadly, the structure of complexity classes. He brings a meticulousness to his research which is reflected in his excellent survey articles on the above topics.

In this complexity column, as a tribute to him, Meena Mahajan and I have put together an overview touching upon some highlights of Eric's research.

A Quest for Structure in Complexity

V. Arvind and Meena Mahajan
The Institute of Mathematical Sciences, HBNI, Chennai, India
{arvind,meena}@imsc.res.in

1 Introduction

Eric Allender turned sixty some months ago, and in this October computational complexity column we will describe some highlights of his research work spanning over three decades. His research career began in 1985. The "Structure in Complexity Conference" was born in 1986. Although FOCS and STOC remain the primary conferences for theoretical computer science, "Structures" was a more niche conference devoted to complexity theory. In the first Structures Conference, in 1986, Eric had two papers. The computational complexity column in the EATCS bulletin was started in 1987 (with Juris Hartmanis as editor). All roughly around the same time. The field has rapidly grown over the last three decades, and Eric has played a prominent role shaping modern computational complexity theory in a range of topics. His work has also given rise to several new directions of research.

In this article, we pick some of our favorite themes from Eric's research: Kolmogorov complexity and circuit minimization, the isomorphism problem and the structure of complete problems in complexity classes, the landscape of space-bounded complexity classes, and circuit complexity lower bounds. Eric himself has a number of excellent comprehensive survey articles on these topics [24, 22, 21, 23]. This column article is meant to serve as an appetizer inviting the reader to these surveys.

2 Kolmogorov Complexity and related topics

Eric's interest in Kolmogorov complexity is right from the start of his research career. One of his early papers which appeared in STOC 1987 examines connections between the existence of pseudorandom generators and a suitable notion of Kolmogorov complexity.

Some highlights of his work related to Kolmogorov complexity are (i) the study of the language of Kolmogorov random strings, and (ii) the intriguing Minimum Circuit Size (MCSP) problem. In this section we will briefly discuss both these problems and his contributions. More details can be found in Eric's excellent survey articles on this topic [24, 22].

2.1 Kolmogorov Randomness

Definition 1. *For a given universal Turing machine U, the Kolmogorov complexity of a string $x \in \{0,1\}^*$, denoted $C_U(x)$, is defined as follows*

$$C_U(x) = \min\{|p| \mid U(p) \text{ outputs } x\}.$$

In words, $C_U(x)$ is the length of the shortest "program" p such that when the universal Turing machine U runs p on the blank tape it halts with x as output.

For another universal Turing machine U', clearly $C_{U'}(x)$ is within an additive constant of $C_U(x)$ for all x. We often drop the subscript and denote the measure by $C(x)$.

As there are at most $2^n - 1$ strings $x \in \{0,1\}^n$ such that $C(x) < |x|$, it follows that the set of strings $\{x \in \{0,1\}^n \mid C(x) \geq |x|\}$ is nonempty for all n. These are the Kolmogorov random strings of length n. It is an interesting exercise to show that a constant fraction of length n strings are Kolmogorov random.

It is easy to observe that the problem of checking whether $C(x) \geq |x|$ for a given string x is undecidable. The proof is by contradiction: if p is a program for deciding if $C(x) \geq |x|$, then for any n we can use p as subroutine by cycling through strings in $\{0,1\}^n$ in lexicographic order to find the first string x such that $C(x) \geq |x|$ (which we know exists). This new program has size $O(\log n) + |p|$ which is smaller than n contradicting the randomness of x.

Furthermore, the set of Kolmogorov random strings is a *co-r.e.* set. I.e. its complement is recursively enumerable.[1] This enumeration can be obtained by a dovetailing enumeration procedure in which we consider all pairs (p,t) of programs and running time bounds. The set of Kolmogorov random strings is, in fact, even co-r.e.-complete under Turing reductions (but not under many-one reductions). This co-r.e. completeness is already somewhat surprising at first sight. How could random strings be used to encode instances of some "well-structured" co-r.e complete problem?

As shown in a series of papers by Allender and his co-authors [8, 7, 17] it turns out that the set of Kolmogorov random strings exhibit very interesting hardness properties even w.r.t. resource-bounded reductions. We explain some of the results from the first of these papers [8]. The language considered in [8] for various Kolmogorov complexity measures μ is

$$R_\mu = \{x \mid R_\mu(x) \geq |x|/2\}.$$

Thus, R_μ consists of strings of high Kolmogorov complexity w.r.t. the measure μ.[2] The different measures considered in [8] are $\mu \in \{C, KT, Kt, KS\}$. The other Kolmogorov complexity measures are defined as follows:

[1] Recall that r.e. stands for all recursively enumerable sets, and a standard r.e.-complete problem is the Halting problem.

[2] As should be expected, the complexity properties of the language R_μ does not depend upon the factor $1/2$ in the definition.

- $Kt(x) = \min\{|p| + \log t \mid U(p) \text{ outputs } x \text{ in} \leq t \text{ steps}\}$.
- $KT(x) = \min\left\{ |p| + t \;\middle|\; \begin{array}{l} \forall \text{ bits } b \text{ and } \forall i \leq |x| + 1 : \\ U(p, i, b) \text{ accepts in } t \text{ steps iff } x_i = b \end{array} \right\}$.
- $KS(x) = \min\left\{ |p| + s \;\middle|\; \begin{array}{l} \forall \text{ bits } b \text{ and } \forall i \leq |x| + 1 : \\ U(p, i, b) \text{ accepts in space } s \text{ iff } x_i = b \end{array} \right\}$.

These measures have natural relativizations as well.[3]
We list some of the surprising hardness results obtained in [8].

Theorem 2. [8]

1. *R_{Kt} is complete for* EXP *under* P/poly*-computable truth-table reductions.*
2. *The Halting problem is* P/poly *truth-table reducible to R_C.*
3. PSPACE *is polynomial-time Turing reducible to R_C.*

These results are surprising because one would not expect resource-bounded reductions to access and use Kolmogorov random strings in deciding hard computational problems. The proofs turn out to be quite simple, building on the following beautiful insight in [8] about the design of pseudorandom generators from hard computational problems in the seminal work of Nisan-Wigderson [33].

The Nisan Wigderson prg construction

Let $f : \{0,1\}^\ell \to \{0,1\}$ be a "hard-to-approximate" boolean function. The Nisan-Wigderson method of constructing a pseudorandom generator (prg) from f works as follows. The generator $G_f : \{0,1\}^m \to \{0,1\}^n$ takes an m-bit seed and stretches it n bits. The parameters are chosen suitably so that m is reasonably larger than ℓ. For a seed $y \in \{0,1\}^m$, the i^{th} bit of the output $G_f(y)$ is obtained by applying f to y projected on a suitable ℓ-subset of indices $S_i \subset [m]$, where the S_i's have at most $\log n$ size pairwise intersection. To see how secure the prg G_f is, we consider "test sets" $T \subseteq \{0,1\}^n$ that can distinguish the output of G_f from the uniform distribution. The set T is a "good" distinguisher if we have

$$|Prob[w \in T] - Prob[G_f(y) \in T]| \geq \frac{1}{poly(n)},$$

where $w \in \{0,1\}^n$ and $y \in \{0,1\}^m$ are uniformly distributed.

Choosing $m = n^\epsilon$ for suitable constant $\epsilon > 0$ and $\ell = n^{\epsilon/2}$, it turns out that we can obtain polynomial-size oracle circuits, with oracle T, for computing f correctly on $1/2 + 1/poly(n)$ fraction of inputs in $\{0,1\}^{n^{\epsilon/2}}$. Additionally, suppose the hard function f has the property that $\text{PSPACE}^f = P^f$. Then, using random self-reducibility, we

[3] There are technical aspects about the definitions [8] which we will not discuss here.

can obtain a polynomial-size oracle circuit that makes parallel queries to oracle T and computes f on all inputs in $\{0,1\}^{n^\epsilon/2}$.

Paraphrasing the above analysis, suppose $A \subseteq \{0,1\}^*$ is a PSPACE-*robust* language. I.e. $\text{PSPACE}^A = \text{P}^A$. Let $f : \{0,1\}^* \to \{0,1\}$ denote the characteristic function of A, chosen as the hard boolean function in the Nisan-Wigderson construction with $m = n^\epsilon$ and $\ell = n^{\epsilon/2}$ for suitable $\epsilon > 0$. If T is any test set that can distinguish truly random strings from the output of the prg G_f then $A \leq_{tt}^{P/poly} T$. Now, suppose T is the set R_{KT}. Notice that for any $x \in \{0,1\}^n$ in the range of G_f then $C^A(x) \leq n^\epsilon$. Thus such strings are not in R_{KT}. It follows that $R_{KT} \cap \{0,1\}^n$ is a distinguishing test set for all n and hence $A \leq_{tt}^{P/poly} R_{KT}$.

This yields the first two parts of Theorem 2, essentially because $C(x)$ and $KT^A(x)$ are within constant factors when A is the Halting problem and $Kt(x)$ and $KT^A(x)$ are within constant factors when A is a complete problem in E.

2.2 The Minimum Circuit Size Problem

A binary string x can be viewed as the truth-table (or prefix of truth-table) of an n-variate boolean function, and it is a natural problem to look for an n-input boolean circuit of minimum size computing this function. Let $csize(x)$ denote the minimum circuit size for x. The corresponding decision problem is:

$$\text{MCSP} = \{(x,k) \mid csize(x) \leq k\}.$$

It is not hard to see that $csize(x)$ and $KT(x)$ are polynomially related to each other. Clearly, MCSP is in NP. There is now sufficient evidence [30, 8, 14] to place MCSP as an NP-intermediate problem that exhibits many hardness properties. A compelling hardness evidence [30] is that if MCSP is polynomial time then cryptographically secure one-way functions cannot exist. A few years back, Allender and Das [10] showed that SZK (the class of promise problems with statistically zero knowledge proofs) is in the class BPP^{MCSP} which, in particular, implies that Graph Isomorphism is randomized reducible to MCSP. On the other hand, there is more recent evidence that suggests that proving NP-hardness of MCSP is unlikely to be easy, if true at all. For instance, if MCSP is NP-complete w.r.t. polynomial time truth-table reductions then EXP $\neq$ ZPP [31]. Furthermore, if MCSP is NP-complete under logspace-computable many-one reductions then PSPACE $\neq$ ZPP.

Before we conclude this section, we sketch the nice observation from [10] that Graph Isomorphism is in RP^{MCSP}.[4]

The key observation that is used is again from [8], which in turn is based on the construction of pseudorandom generators from any one-way function [29]. Suppose $f_y : \{0,1\}^n \to \{0,1\}^{r(n)}$ is a collection of functions for $y \in \{0,1\}^{r(n)}$, computable in time $p(n)$ for some polynomials $p(n)$ and $r(n)$. Then there is a randomized polynomial-time

[4] In fact they show the stronger result that all of SZK is in BPP^{MCSP}.

oracle algorithm M^{MCSP} that takes as input $(y, f_y(x))$ for a random $x \in \{0,1\}^n$ and outputs a string in $f_y^{-1}(f_y(x))$ with probability at least n^{-c} for some constant $c > 0$.

For any n-vertex graph X define the function f_X that maps a permutation $\pi \in S_n$ to the n-vertex graph $\pi(X)$.

Now, suppose a pair of n-vertex graphs (G, H) is an instance of Graph Isomorphism. For a randomly picked permutation $\pi \in S_n$, we run the above algorithm M^{MCSP} on input $(H, f_G(\pi))$. If the graphs G and H are isomorphic then, with n^{-c} probability, the algorithm outputs a permutation τ such that $\tau(H) = \pi(G)$. Clearly, if the graphs are not isomorphic the algorithm will never output such a permutation. This yields the desired RP^{MCSP} algorithm for Graph Isomorphism.

3 The Isomorphism Conjecture for Complexity Classes

The well-known Schröder-Bernstein theorem, in naive set theory, states that if A and B are sets such that there are injective maps $f : A \to B$ and $g : B \to A$ then, in fact, there is a *bijection* $h : A \to B$.

Myhill [32] adapted the Schröder-Bernstein theorem to show that all r.e.-complete sets (under recursive reductions) are r.e.-isomorphic. The original back and forth argument used in the Schröder-Bernstein proof was cleverly modified to work for recursive reductions by Myhill. The crucial property that he discovered [32] and used in this proof is that r.e.-complete sets are *recursively paddable*. This property can be used to show that the back and forth construction of Schröder-Bernstein always terminates, yielding a recursively invertible bijection h as reduction between A and B (two r.e.-complete sets).

Hartmanis and Berman conjectured [28], by analogy, that NP-complete sets should be isomorphic under polynomial-time computable and invertible bijections as reductions. They used a stronger notion of paddability (which many natural NP-complete problems have, but probably not all) that allows them to carry out the above construction in the polynomial time setting. It was an influential conjecture in the area, which led to many interesting results in complexity theory of the 1980's and 1990's. Eric has made some fine contributions to research in this sub-field of complexity theory. We will highlight one of these results.

Research soon turned to showing the Berman-Hartmanis conjecture for more restricted types of reductions. Among these, the most interesting reduction was AC^0 reductions (computable by uniform constant-depth reductions). Many NP-complete problems are actually NP-complete under AC^0 reductions (in fact, they are even complete under projection reductions). Allender, Balcázar, and Immerman [6] first showed that sets complete for NP under *projections* are AC^0-isomorphic (where the AC^0 circuits computing the reductions are uniform).

As a next step Eric, along with Manindra Agrawal, investigated the isomorphism conjecture for reductions that are NC^0 computable [1]. Building on their initial work, they could show along with Steve Rudich, that sets complete under AC^0 reductions are

also complete under NC^0 reductions. And this property holds for several complexity classes including NP, P, and NC^1. Furthermore, they could also show for these complexity classes, that sets complete under AC^0 reductions are indeed AC^0 isomorphic (but only w.r.t. nonuniform AC^0 reductions) [1, 4]. Their main isomorphism theorem can be summarized as follows:

Theorem 3. [4] *Let C be any complexity class closed under uniform* NC^1 *computable many-one reductions. Then the languages that are complete for C under* AC^0 *computable reductions are also isomorphic under (non-uniform)* AC^0 *computable isomorphisms.*

The question that remained was to strengthen the above result to obtain dlogtime-uniform AC^0 computable isomorphisms. Following some improvements to the above on uniformity [3, 12], the culminating result in this line of research was Agrawal's theorem [13] that AC^0-complete sets were also AC^0 isomorphic (where the reductions are dlogtime uniform), for all these complexity classes, including NP.

4 The Logspace Complexity Lens

A recurring theme in Eric's work has been small-space classes. More specifically, logarithmic space, in different computation modes. The most restrictive, of course, is deterministic logspace DLOG. Add nondeterminism to get NLOG. Add a stack, to get LogDCFL. Add both, get LogCFL. Add thresholding, get PL. As we add more and more resources, what more can we compute? Impose semantic restrictions: symmetry – SL, few counting paths – FewL, unambiguity – ULOG. How much does the computing power drop? Related to these, yet fundamentally a different question, concerns counting accepting paths in space-bounded computation models. What kind of functions can be so represented? These questions, in their many guises, have intrigued Eric, and over the years, he has contributed significantly to building a good understanding of the landscape here.

4.1 Unambiguous Acceptance

In the eighties, Valiant and Vazirani showed that satisfiability can be reduced, probabilistically, to instances with none or just one solution; fixing the randomness appropriately tells us that NP is contained in $\oplus$P/poly. Mulmuley, Vazirani and Vazirani established the Isolation Lemma, a witness-pruning technique that is more amenable to parallel and/or small-space computation. Gál and Wigderson used this lemma to prune NLOG witnesses, establishing that NLOG/poly is contained in $\oplus$LOG/poly. A crucial ingredient in their proof was the construction of min-unique graphs; between any pair (u, v) of nodes, the shortest distance is achieved by a unique path. However, they only used this property for the pair (s, t). Using it for all pairs of the form (s, u), Eric

and Klaus Reinhardt showed that in fact a different pruning yields a unique witness, establishing the following theorem.

Theorem 4 (Theorem 2.2 and Corollary 2.3 in [34])**.** *NLOG (and even NLOG/poly) is contained in ULOG/poly.*

This is a great result; in the non-uniform world, nondeterminism can be made unambiguous without increasing the space required. And why is this desirable? Well, amongst other things unambiguous computation gives witness functions which could conceivably be useful in designing parallel algorithms for search problems. The NL/poly=UL/poly result is a central result in a body of work studying limiting nondeterminism in space-bounded computation: promise classes with polynomially many accepting paths, strong unambiguity, etc. In an earlier piece of work with Klaus-Jörn Lange [18], Eric had shown that reach-unambiguous logspace computation can be simulated deterministically in space less than $O(\log^2 n)$. Just clarifying various subtleties in differing notions of unambiguity is itself a valuable contribution of that work.

In the late eighties, Immerman and Sclepscenyi had developed the inductive counting technique to complement non-deterministic space. It was this technique that was built on, and generalized to what the paper calls the double counting technique, to disambiguate NLOG and obtain Theorem 4.

4.2 Unambiguous Counting

Counting accepting paths in nondeterministic computation gives us the well-known function classes #P, #LOG, and so on. Their closure under subtraction gives the Gap classes, first defined by Fenner, Fortnow and Kurtz. One way of defining an analogue of unambiguity for the Gap classes is requiring the gap function to be 0 or 1, giving the so-called stoic classes like SPP and its logspace analogue SPL. A language L is in the class SPL if there is a nondeterministic logspace machine with the property that for all words x, the difference between the number of accepting computations of M on X and the number of rejecting computations of M on x is 1 when $x \in L$, and 0 when $x \notin L$. The class SPL contains NLOG and is contained in NC^2.

The perfect matching problem has fascinated complexity theorists for years. It is a natural problem that does not yet characterize any nice complexity class (that is, we don't know of a class for which it is complete via suitable reductions). Its counting version is famously #P-complete, its decision and search versions are in P and randomized NC. Decision is in non-uniform NC^2, but we do not know if it is in deterministic NC. (Very recent excitement - it is almost there! it is in quasi-NC, and for bipartite graphs it is also in pseudo-deterministic NC.) Combining ideas from the isolation lemma and a combinatorial algorithm for the determinant, Eric and his co-authors Reinhardt and Zhou showed the following:

Theorem 5 (Theorems 3.1, 3.2 in [26])**.** *The perfect matching problem (decision) is in non-uniform SPL. The search problem is in the functional version non-uniform FSPL.*

In the non-uniform world, this is the best upper bound we have for the perfect matching problem.

4.3 Planar Reachability

Matchings in planar graphs turns our intuition that counting is at least as hard as search and decision on its head. Counting perfect matchings in planar graphs is in P and even NC, whereas in general graphs deciding existence via shallow circuits so far seems to need non-uniformity or quasi-polynomial size. Some of Eric's work has delved into how planarity helps in the quintessential NLOG problem reachability; see for instance [5]. It explores bounds for solving REACH on various kinds of grid graphs. In particular, it shows that reachability on layered grid graphs can be decided in ULOG. Research on this theme continues today; we still do not know if planar reachability is provably easier than NLOG, but we do know it is in ULOG. Another nice result from [5] is that planar reachability and planar unreachability are logspace equivalent.

4.4 Symmetry

Adding a stack to logspace computation – it now becomes necessary to explicitly restrict time to polynomial, otherwise you get all of P. With the poly time bound, we have a non-deterministic pushdown automaton with an auxiliary logspace worktape, and it is easy to see that such a machine can simulate a logspace reduction to a context-free language. Sudborough showed that it can in fact do no more; any such computation can be decomposed into a deterministic logspace reduction followed by a non-deterministic pushdown-automata computation. Hence the class of languages accepted by such machines is called LogCFL, and the work of Ruzzo and of Venkateswaran shows that it is also characterized by semi-unbounded log depth poly-size circuits SAC^1 or by poly-size poly-degree circuits. This class has figured frequently in Eric's work. We mention one particular result here. As with NLOG, computation in this class can be made unambiguous non-uniformly. Unlike with NLOG, where the restriction to symmetric computation was long known to be somehow easier – upper bounds on SL included $\oplus$LOG, DTIME,SPACE(poly, $\log n^2$), DSPACE($\log n^{4/3}$) – and was finally shown by Reingold to coincide with determinism (the famed SL=L result from 2004), LogCFL behaves more like a time-bounded class in the context of symmetric computation. More precisely,

Theorem 6 (Theorem 3.1 in [19])**.** *NAuxPDA-TIME($n^{O(1)}$) = SymAuxPDA-TIME($n^{O(1)}$). That is, for polynomial-time-bounded AuxPDA, symmetry and nondeterminism coincide.*

4.5 The PL and $C_=L$ Hierarchies

Probabilistic logspace PL is a bit of a strange class. Here, restricting the time to polynomial is not necessary, the class remains the same. Eric, along with Ogihara, gave a simpler proof of this than was earlier known, using closure properties of GapL. This work [25] also gave us circuit characterizations of various logspace hierarchies. We now know that $AC^0(PL)$ is exactly the PL hierarchy. So no more do we need to deal with messy details while relativizing space-bounded computation; life simplified!

An interesting variant of PL is the exact counting class $C_=L$ defined as follows: A language is in $C_=L$ if for some nondeterministic logspace machine, the numbers of accepting and rejecting computations are equal exactly for words in the language. It is easy to see that all of SPL is contained in $C_=L$, but there could be much more. Whether the class $C_=L$ is closed under complement remains an intriguing open question. The most natural complete language for $C_=L$ is singular integer matrices.

In [25], an analogous result for the exact counting hierarchy was also shown; $AC^0(C_=L)$ equals the $C_=L$ hierarchy. In subsequent work with Beals and Ogihara [9], Eric showed that this hierarchy also equals $NC^0(C_=L)$, and that it collapses to $L^{C_=L}$. He showed that this hierarchy captures the essence of something fundamental in linear algebra – determining whether the rank of a given integer matrix is an odd number is complete for this hierarchy.

Theorem 7 ([9]). *The following problems are complete for $AC^0(C_=L)$:*

$$\begin{aligned} \mathit{FSLE} &= \{(A,b) \mid A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^{m\times 1}, \exists x \in \mathbb{Q}^{n\times 1} : Ax = b\} \\ \mathit{OddRank} &= \{A \mid A \in \mathbb{Z}^{m\times n},\ \mathit{rank(A)\ is\ an\ odd\ number}\} \\ \mathit{Comp.Rank} &= \{(A,i,b) \mid A \in \mathbb{Z}^{m\times n}, \mathit{rank(A) = r,\ bit\ i\ of\ r\ is\ b}\} \end{aligned}$$

An interesting "trick" described explicitly in this paper is a logspace transformation mapping integer matrix M to M' while preserving non-singularity, but with the additional property that if M is singular, then M' has rank exactly one less than full-rank.

5 Circuit Complexity and related questions

5.1 Boolean Circuit Lower Bounds

Obtaining exponential lower bounds against general Boolean circuits is one of the holy grails in complexity theory. Straightforward counting arguments tell us that most languages cannot be recognized by small circuits. But finding and explicitly describing one – searching for hay in a haystack – is notoriously hard. Little wonder then that we sharpen our tools working with restricted circuits. There was the heady excitement in the eighties of discovering that parity needs exponential-size in constant-depth circuits, even if augmented with modulo-3 gates (Ajtai; Furst, Saxe, Sipser; Hastad;

Razborov, Smolensky). Soon thereafter came a different kind of excitement: a single query to the Permanent is enough to decide languages anywhere in the polynomial hierarchy (Toda). In work with Vivek Gore, Eric noticed a most significant fact about the techniques in Toda's result: they could be exploited to give significant lower bounds against *uniform* constant-depth circuits, even such circuits augmented with modular counting (ACC circuits). In particular, he showed that a uniform constant-depth circuit with modular counting cannot compute the permanent unless it has exponential size.

Theorem 8 (Theorem 3.4 in [11])**.** *The permanent function does not have uniform ACC(subexp) circuits.*

The permanent is of course a function; an analogous but slightly weaker result holds for languages in PP. Namely, PP does not have uniform ACC circuits of subsubexp size. (The definition of this kind of size functions is rather technical; suffice to keep in mind that they are super-polynomial!)

The notion of uniformity used here is a polynomial-version of DLOGTIME uniformity; for subexp size, we could think of it as polylogtime uniformity. The lower-bound proof crucially uses this notion, and does not work for less restrictive notions of uniformity. But let us pause to consider: are lower bounds against uniform circuits interesting at all? Well, one of the earliest known separations is the time hierarchy theorem, and it tells us that EXPTIME-complete languages do not have uniform polynomial-size circuits. No one's saying that's not interesting! And as in the case of the Permanent, we do not know how to prove this separation in the non-uniform setting. We do not even know whether the separation holds; whether EXPTIME is in P/poly is wide open.

A crucial ingredient in this uniform-ACC lower bound for the Permanent is a special kind of depth-reduction result that was proved in [11]. Every subsexp-size ACC circuit can be converted, uniformly, to a depth-2 subexp-size circuit of the form SYM$\circ$AND, where the AND gates have relatively small fanin. Essentially this result, in the non-uniform setting, was established by Beigel and Tarui building on preceding work by Yao, by Eric himself, and by Toda. But the notion of uniformity required here is quite restrictive, and much care was needed to establish that it goes through in this setting as well. Formally, the result is as follows.

Theorem 9 (Theorem 3.1 in [11])**.** *Suppose L is accepted by a uniform ACC(subexp) circuit family. Then it is accepted by a uniform depth two circuit family of $s(n)$ sized circuits that have the following properties:*

1. *Level one has a subexponential number of AND gates each with fan-in $(\log s(n))^{O(1)}$. Given the name of an AND gate, its exact fan-in can be computed deterministically in time $(\log s(n))^{O(1)}$.*

2. *Level two has a symmetric gate. Given the number of AND gates that evaluate to one, the symmetric gate can be evaluated deterministically in time $(\log s(n))^{O(1)}$.*

This ACC depth-reduction has proven to be probably more significant than Eric then thought! Over 20 years later, the fact that this depth-reduction is uniform, and hence efficiently computable, was exploited by Ryan Williams while designing an algorithm for circuit satisfiability of subexp-size ACC circuits. This algorithm is a crucial ingredient in his proof separating NEXP from non-uniform ACC.

Incidentally, a few years later, Eric went on to extend the lower bound to uniform constant-depth threshold circuits. He showed:

Theorem 10 (Corollary 1 in [20])**.** *The Permanent does not have uniform poly-size or even quasi-poly-size TC^0 circuits.*

5.2 Arithmetic Circuits

Counting classes have been mentioned above; they typically consist of functions that count accepting paths in non-deterministic computation models, or the closure under subtraction of such functions. They are often naturally characterized by arithmetized versions of the Boolean circuits that correspond to the nondeterministic computation. Arithmetization over various algebras gives different kinds of counting. For instance, the class LogCFL (which coincides with the class of languages accepted by polynomial-time AuxPDA) is characterized by uniform semi-unbounded circuits SAC^1. Arithmetizing them over $+, \times$ gives circuits counting accepting paths; the class $\#\mathrm{SAC}^1$. One of Eric's early investigations concerned arithmetization over non-commutative algebras. Over commutative rings, a striking result from the work of VSBR (Valiant, Skyum, Berkowitz and Rackoff) states that a polynomial-size polynomial-degree circuit can be restructured to an equivalent semi-unbounded log-depth circuit. Kosaraju and Nisan gave explicit examples of non-commutative circuit classes where such a depth-reduction is provably not possible; the underlying rings are (union, concat). Eric's contribution was two-fold: firstly, he re-examined the VSBR depth-reduction and gave a uniform version of it (the VSBR construction is non-uniform; it needs polynomial identity testing). Secondly, he showed that a somewhat weaker depth-reduction, to unbounded rather than semi-unbounded circuits, also works for the non-commutative ring of (max, concat).

Theorem 11 (Theorem 3.1 in [15]; see also [16])**.** *If f is computed by a family of arithmetic circuits over* $(\Sigma^*, \max, \mathit{concat})$ *of polynomial size and degree, then f is computed by a family of arithmetic circuits over* $(\Sigma^*, \max, \mathit{concat})$ *of polynomial size with depth* $O(\log^2 n)$.

This was the first instance of depth-reduction for a non-commutative ring, and showed that function classes like OptLOG and OptLogCFL have NC algorithms.

It is intuitively clear that the thresholding operation is intimately connected with counting. This is why, for instance, PP and #P are equivalent with respect to polynomial-time Turing reductions. This obvious connection, however, becomes non-obvious when dealing with very small circuit classes. In another interesting piece of

work with Manindra Agrawal and Samir Datta, Eric helped make this intuition precise for constant-depth circuits. This work characterizes the Boolean class TC^0 in terms of the counting class $\#AC^0$ with one threshold gate or equality check on top, establishing the following:

Theorem 12 (Theorem in [2])**.** *$TC^0 = PAC^0 = C_{=}AC^0$.*

The paper has several caveats about the type of uniformity, and the potential difference between $DiffAC^0$ (functions reporting the difference of $\#AC^0$ functions) and $GapAC^0$ (functions computed by constant-depth arithmetic circuits with $x_i, 0, 1, -1$ at leaves). However subsequent work has ironed out all these niggling questions; the Diff and Gap AC^0 classes coincide, and with an exact threshold on top, characterize TC^0 with all notions of uniformity discussed there.

A lovely construction by Barrington shows that polynomial-size Boolean formulas (and hence, by Brent's depth-reduction, languages in the class NC^1) have equivalent polynomial-size, width-5 branching programs. An equally lovely construction by Ben-Or and Cleve lifts this idea to arbitrary commutative rings, giving width-3 branching programs. An equivalent way of stating this result is that $IMM_{3,n}$ – evaluating the product of n 3×3 matrices of indeterminates – is complete for algebraic NC^1. What about multiplying 2×2 matrices? Eric and Fengming Wang showed that this is not hard enough, by showing that $IMM_{2,n}$ is not universal.

Theorem 13 (Theorem 1.2 in [27])**.** *For $k \geq 8$, the $2k$-variate polynomial*

$$f(x_1, \ldots, x_{2k}) = \sum_{i=1}^{k} x_{2i-1}x_{2i}$$

cannot be computed by algebraic programs of width 2 over any field. Hence $IMM_{2,n}$ is not complete for algebraic NC^1 under regular projections.

(Projections are restrictive reductions allowing only substitution by variables or field constants. Regular projections allow substitutions by general affine forms.)

This theorem is a great example of a result which intuitively we all believed must be true, but could not (or would not, or whatever; but finally did not) prove, until all the details were ironed out in [27] with a proof that is more complex than what the statement seemed to warrant.

Theorem 13 is in contrast to the Boolean case, where Lipton and Zalcstein showed that (Boolean) $IMM_{2,n}$ is indeed hard for NC^1, albeit under slightly less restrictive AC^0 reductions.

6 To conclude ...

After reading this article, some of you may want to know more about these areas of complexity theory. We recommend reading Eric's expository articles and surveys, all of which are available online from his publications page at https://www.cs.rutgers.edu/ allender/publications/. Happy reading!

References

[1] Manindra Agrawal and Eric Allender. An isomorphism theorem for circuit complexity. In *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity, Philadelphia, Pennsylvania, USA, May 24-27, 1996*, pages 2–11, 1996.

[2] Manindra Agrawal, Eric Allender, and Samir Datta. On TC^0, AC^0, and arithmetic circuits. *J. Comput. Syst. Sci.*, 60(2):395–421, 2000.

[3] Manindra Agrawal, Eric Allender, Russell Impagliazzo, Toniann Pitassi, and Steven Rudich. Reducing the complexity of reductions. *Computational Complexity*, 10(2):117–138, 2001.

[4] Manindra Agrawal, Eric Allender, and Steven Rudich. Reductions in circuit complexity: An isomorphism theorem and a gap theorem. *J. Comput. Syst. Sci.*, 57(2):127–143, 1998.

[5] Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and grid graph reachability problems. *Theory Comput. Syst.*, 45(4):675–723, 2009.

[6] Eric Allender, José L. Balcázar, and Neil Immerman. A first-order isomorphism theorem. *SIAM J. Comput.*, 26(2):557–567, 1997.

[7] Eric Allender, Harry Buhrman, and Michal Koucký. What can be efficiently reduced to the Kolmogorov-random strings? *Ann. Pure Appl. Logic*, 138(1-3):2–19, 2006.

[8] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[9] Eric Allender, Robert Beals, and Mitsunori Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2):99–126, 1999.

[10] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 25–32, 2014.

[11] Eric Allender and Vivek Gore. A uniform circuit lower bound for the permanent. *SIAM J. Comput.*, 23(5):1026–1049, 1994.

[12] Manindra Agrawal. Towards uniform AC^0 - isomorphisms. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 13–20, 2001.

[13] Manindra Agrawal. The isomorphism conjecture for constant depth reductions. *J. Comput. Syst. Sci.*, 77(1):3–13, 2011.

[14] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *Computational Complexity*, 26(2):469–496, 2017.

[15] Eric Allender and Jia Jiao. Depth reduction for noncommutative arithmetic circuits. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 515–522, 1993.

[16] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-commutative arithmetic circuits: Depth reduction and size lower bounds. *Theor. Comput. Sci.*, 209(1-2):47–86, 1998.

[17] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:51, 2009.

[18] Eric Allender and Klaus-Jörn Lange. Ruspace($\log n$) $\subseteq$ DSPACE($\log^2 n/\log\log n$). *Theory Comput. Syst.*, 31(5):539–550, 1998.

[19] Eric Allender and Klaus-Jörn Lange. Symmetry coincides with nondeterminism for time-bounded auxiliary pushdown automata. *Theory of Computing*, 10:199–215, 2014.

[20] Eric Allender. The permanent requires large uniform threshold circuits. *Chicago J. Theor. Comput. Sci.*, 1999, 1999.

[21] Eric Allender. Arithmetic circuits and counting complexity classes. In *Complexity of Computations and Proofs, Quaderni di Matematica*, pages 33–72, 2004.

[22] Eric Allender. Curiouser and curiouser: The link between incompressibility and complexity. In *How the World Computes - Turing Centenary Conference and 8th Conference on Computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012. Proceedings*, pages 11–16, 2012.

[23] Eric Allender. Investigations concerning the structure of complete sets. In *Perspectives in Computational Complexity – the Somenath Biswas Anniversary Volume*, volume 26 of *Progress in Computer Science and Applied Logic*, pages 23–35. Springer Verlag, 2014.

[24] Eric Allender. The complexity of complexity. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pages 79–94, 2017.

[25] Eric Allender and Mitsunori Ogihara. Relationships among PL, #L, and the determinant. *ITA*, 30(1):1–21, 1996.

[26] Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *J. Comput. Syst. Sci.*, 59(2):164–181, 1999.

[27] Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *Computational Complexity*, 25(1):217–253, 2016.

[28] Leonard Berman and Juris Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM J. Comput.*, 6(2):305–322, 1977.

[29] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[30] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79, 2000.

[31] Cody D. Murray and Richard Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 365–380, 2015.

[32] John Myhill. Creative sets. *Mathematical Logic Quarterly*, 1(2):97–108, 1955.

[33] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[34] Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000.