The Enumerability of P Collapses P to NC ⋆

Alina Beygelzimer [a,*], Mitsunori Ogihara [b]

[a] *IBM T.J. Watson Research Center, Hawthorne, NY 10532.*
Email: beygel@us.ibm.com

[b] *Department of Computer Science, University of Rochester, Rochester, NY 14627.*
Email: ogihara@cs.rochester.edu

Abstract

We show that one cannot rule out even a single possibility for the value of an arithmetic circuit on a given input using an NC algorithm, unless P collapses to NC (i.e., unless all problems with polynomial-time sequential solutions can be efficiently parallelized). In other words, excluding any possible solution in this case is as hard as actually finding the solution. The result is robust with respect to NC algorithms that err (i.e., exclude the correct value) with small probability. We also show that P collapses all the way down to NC^1 when the characteristic of the field that the problem is over is sufficiently large (but in this case under a stronger elimination hypothesis that depends on the characteristic).

Key words: Complexity theory, circuit value problem, enumerative approximations

1 Introduction

We consider the question of whether ruling out a possible solution to a problem can be as hard as actually solving the problem. We investigate the relationship between the property of being "prunable" in the above sense and the property of being inherently sequential, and show that if a canonical P-complete problem is "prunable," then P collapses to NC.

⋆ Preliminary version of this paper appeared in Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS), Lecture Notes in Computer Science 3153, Springer 2004.
* Corresponding Author.

More precisely, suppose that there exists an NC procedure that, given an arithmetic circuit C and an input x to C (both over some finite field), could rule out one possibility (i.e., one field element) for the value of C on x. We show that then the Boolean Circuit Value Problem (i.e., the problem of computing the value of a Boolean circuit on a given input, CVP for short) is in NC. Since CVP is complete for P under NC^1-computable many-one reductions, this would collapse P to NC. Thus one cannot rule out *any* candidate for the value of a given arithmetic circuit on a given input without proving that there are no inherently sequential problems in P.

The task of ruling out possible values can be formalized using the notion of *enumerability*, introduced by Cai and Hemachandra [8] (and used to prove a number of results since then, e.g., [9, 1, 5, 3]). Let $k : \mathbb{N} \to \mathbb{N}^+$ be a monotone non-decreasing function. A $k(n)$-*enumerator* for a function f is a procedure that reduces the set of possibilities for the value of f on a given input x of length n by enumerating a list of $k(n)$ values guaranteed to contain $f(x)$ (thereby eliminating the rest of the values). For such a procedure to be interesting, it has to do something other than simply compute $f(x)$ and include it in the output; to enforce this we consider enumerators that are, in some computational sense, weaker than the best known algorithms for the functions they enumerate. An enumerator yields an approximation of f, but instead of restricting the value of $f(x)$ to an interval (as would a classical approximator), it restricts this value to a set.

Cai and Hemachandra [9], motivated by investigating alternative ways to approximate #P functions, showed that if the permanent is poly-enumerable in polynomial time, then $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$. (See also Amir, Beigel, and Gasarch [1].) Thus it is as hard to substantially prune the space of possible values for the permanent as to compute it exactly. Beygelzimer and Ogihara [5] showed that although the determinant is computationally much easier than the permanent, a similar hardness result holds for the determinant as well: The determinant is not poly-enumerable in logspace unless $\mathrm{L} = \mathrm{L}^{\#\mathrm{L}}$. Beals et al. [3] proved that #GA (the function computing the number of automorphisms of an undirected graph), is not poly-enumerable in polynomial time, unless GI (the set of pairs of isomorphic graphs) is in RP.

Several important properties of sets can be formulated using the notion of enumerability, and similar non-enumerability results are known here as well. Hemachandra and Rudich [HR90] showed that if for every language in P, its ranking function (i.e., the function determining the position of elements in the set) is $O(1)$-enumerable in polynomial time then $\#\mathrm{P} = \mathrm{FP}$. Their construction was built on the result of Cai and Hemachandra [8], and the later, stronger results of Cai and Hemachandra [9], in light of the Hemachandra-Rudich argument, strengthen the result in [HR90] to poly-enumerability. In a similar vein, Goldsmith, Ogihara, and Rothe [14] show that $\#\mathrm{P}_1$ (the tally version of #P in which the inputs are given in unary) is not poly-enumerable in polynomial time, or equivalently, the census function of all P sets is not poly-enumerable in polynomial time, unless $\#P_1$ is in PF. (The census function of a set L maps each $n > 0$, as a unary string 1^n, to the number of strings in L of length n.) They show that $\#P_1 \subseteq$ FP would imply that P=BPP, and moreover, that PH $\subseteq \mathrm{Mod}_k$P for any $k \geq 2$.

The enumerability assumption in this paper is much weaker than used previously: instead of reducing the set of candidates substantially, we just want to exclude a single possibility.

In this paper, we initiate the study of probabilistic enumerators that are allowed to err (i.e., exclude the correct value) with small probability. The result is robust with respect to such enumerators. We also prove that P collapses all the way down to NC^1 for the special case when the characteristic of the ground field is sufficiently large (and under a stronger hypothesis that depends on the characteristic). The collapse mentioned earlier is to NC^3. For the second collapse, let $\mathbb{F}$ be any fixed finite field with characteristic at least k^2 for some integer $k \geq 2$. We show that $\mathrm{CVP}_{\mathbb{F}}$ is not k-enumerable unless $\mathrm{CVP}_{\mathbb{F}}$ is solvable in NC^1, modulo the complexity of the enumerator. Here, and throughout the paper, $\mathrm{CVP}_{\mathbb{F}}$ denotes the arithmetic circuit value problem over the field $\mathbb{F}$. If $\mathbb{F}$ is not fixed but is of size at most polynomial in the size of the input, then the construction is in TC^1.

The first result is proven by adapting techniques used in Sivakumar's proof that SAT is not $O(\log n)$-membership comparable [1], unless UniqueSAT is in P [27], which builds on the technique of Ar et al. [2] for reconstructing polynomials from noisy data. The second result is proven using new techniques developed here.

2 Preliminaries

Let k be a positive integer. We denote the set $\{1, \ldots, k\}$ by $[k]$. A *set system* is a subset of $\mathcal{P}(k)$, the power set of $[k]$. The number of elements in a set X is denoted by $|X|$. For any string x, the length of x is also denoted by $|x|$. There will be no chance of confusion. All logarithms are base 2. We write $\langle x_1, \ldots, x_n \rangle$ to denote the n-tuple of $x_1, \ldots, x_n$ as well as the encoding of this n-tuple as a bit string, using a fixed, well-behaved pairing function extended to tuples (computable and invertible in time and space affordable in the context).

[1] A $g(n)$-*membership comparing function* for a language A takes a sequence of $g(n)$ strings, each of length at most n, and outputs a string in $\{0,1\}^{g(n)}$ that is not their characteristic string in A. (See [21, 4].)

An *arithmetic circuit* over some algebraic structure $\mathbb{F}$ is a circuit with gates computing operations over $\mathbb{F}$ and taking elements and constants of $\mathbb{F}$ as inputs. There is a unique output gate. A circuit with n inputs computes a function from $\mathbb{F}^n$ to $\mathbb{F}$ in the natural way. Any such circuit can be described by a string over a finite alphabet, if the constants used can be described this way. A Boolean circuit is just an arithmetic circuit over GF(2). We are essentially using $\mathbb{F}$ as a finite alphabet, and the results can be shown to hold in a more general setting (provided that the operations over the alphabet can be efficiently computed).

We use standard definitions of circuit complexity classes. Since we want to make circuit classes comparable to uniform classes defined in terms of time and space, we need to place uniformity restrictions on circuit families. For our purposes, it will be sufficient to use logspace uniformity, meaning that there exists a logspace machine that, on input 1^n, generates a standard encoding [23] of the n-th circuit in the family.

For $k \geq 0$, the class NC^k is defined as the class of all functions computable by (logspace uniform) polynomial-size, bounded fan-in Boolean circuits of depth $O(\log^k n)$. Similarly, the class AC^k is the class of all functions computable by (logspace uniform) polynomial-size, arbitrary fan-in Boolean circuits of depth $O(\log^k n)$. A *majority gate* outputs 1 if and only if at least half of its inputs have value 1. The class TC^k is defined as the class of all functions computable by (logspace uniform) arbitrary fan-in Boolean circuits composed entirely of majority gates; again, the circuits are of polynomial size and depth $O(\log^k n)$. If $k = 0$, the depth is constant.

We use Cook's generalization [11] of Wilson's model [32] to define circuit-based reductions. (We do not make a distinction between circuits computing functions and circuits deciding languages.) In this model, circuits are allowed to have oracle gates, in addition to the usual gates. An oracle gate takes an ordered list of r inputs, interpreted as a query string. The sequence of s ordered outputs corresponds to the value of the oracle on the query string. For the purpose of defining depth in bounded fan-in circuits, an oracle gate with r inputs and s outputs counts as depth $\lceil \log(r + s) \rceil$.

An instance of the *arithmetic circuit value problem* over $\mathbb{F}$ (denoted $\mathrm{CVP}_{\mathbb{F}}$) is an encoding of an arithmetic circuit C over $\mathbb{F}$, together with an encoding of inputs $x_1, \dots, x_n$ from $\mathbb{F}$. The circuit C has n inputs, and the problem is to compute the value of C on $x = \langle x_1, \dots, x_n \rangle$, denoted $C(x)$.

Ladner [17] introduced the (Boolean) circuit value problem, and showed that it is complete for P (under logspace computable many-one reductions). Cook observed that the problem is actually complete under NC^1 reductions [11]. For infinite fields, the problem is not necessarily in FP, since gate values need not be of polynomial length. It is in FP, however, for any finite field, and remains FP-hard in any field. Correspondingly, it is P-hard to decide whether C on x outputs 1, where 1 is the multiplicative identity of the field. (See [15], Appendix A.1.) The fields we consider are finite. The hardness above is with respect to NC^1-computable many-one reductions; hence it suffices to show that CVP (or $\mathrm{CVP}_{\mathbb{F}}$) is in NC^i to collapse P to NC^i for some $i \geq 1$. We will establish such a collapse under the hypothesis that, given an instance $\langle C, x\rangle$ of $\mathrm{CVP}_{\mathbb{F}}$, one can efficiently eliminate at least *one* of $|\mathbb{F}|$ possibilities for the value of C on x.

Let $k : \mathbb{N} \rightarrow \mathbb{N}^+$ be a monotone non-decreasing function. Recall that a $k(n)$-*enumerator* for a function f, defined by Cai and Hemachandra [8], is a deterministic procedure E_f that on input x of length n outputs a set of $k(n)$ values, one of which is guaranteed to be $f(x)$. This yields a certain notion of approximation: instead of restricting $f(x)$ to an interval (for example, by outputting a value guaranteed to be within some multiplicative factor from $f(x)$), an enumerator restricts $f(x)$ to a small set, namely $E_f(x)$; the values in the set need not be consecutive. The number of elements in this set depends only on n. In contrast, the size of the approximating interval is a function of $f(x)$, and thus if $f(x)$ is exponentially large, then so is the interval (as in the case of the permanent, determinant, or #GA functions). Enumerative counting may also be an approximation of choice in a more general context where there is no natural ordering on the range of f, hence restricting the value to an interval is not particularly meaningful (as, arguably, in the case of computations over finite fields). Such an approximation, however, does not satisfy a natural "nesting" requirement: If we parameterize the enumerator by the size of the set it outputs, then a larger approximating set does not necessarily contain a smaller set for the same input, since the only requirement is the containment of the exact value. Thus there is no locality: the set as a whole approximates the exact value, while every value in the interval can serve as an approximation.

3 Finite Fields of Large Characteristic

We first prove the deeper collapse mentioned in the introduction (for fields with large characteristic), and then turn to arbitrary finite fields.

A Note on Representation To implement the arithmetics in a finite field $\mathbb{F}$ with p^q elements, it suffices to have a monic irreducible polynomial g in $\mathbb{Z}_p[x]$ of degree q, since $\mathbb{F}$ can be viewed as $\mathbb{Z}_p[x]/(g)$, the ring of polynomials over $\mathbb{Z}_p$ modulo g. If $\mathbb{F}$ is fixed, then such g can be precomputed in logspace.

Theorem 1 Let $\mathbb{F}$ be a (fixed) finite field with characteristic at least k^2 for some integer $k \geq 2$, and let $i \geq 1$. If $\mathrm{CVP}_{\mathbb{F}}$ is k-enumerable in NC^i, then $\mathrm{CVP}_{\mathbb{F}}$ is in NC^i, modulo the complexity of the enumerator. If $\mathbb{F}$ is described in the input, then the same result holds with TC^i in place of NC^i.

Proof: Let $\langle C, x\rangle$ be an instance of $\mathrm{CVP}_{\mathbb{F}}$, where C is an encoding of an arithmetic circuit over $\mathbb{F}$ with n inputs, and x refers to the encoding of $\langle x_1, \ldots, x_n\rangle \in \mathbb{F}^n$. To avoid new notation, assume that C has n gates. Notice that, given an assignment to the gates of C, we can check it for validity in NC^1; we just need to locally verify that the value assigned to every gate is in fact the value of the operation it computes on the values assigned to its inputs. Thus all we need is a constructible-in-NC^1 set of gate assignments containing the correct one for C on input x. Of course, we cannot simply run the enumerator on every gate of C (viewing it as an output gate), as it would yield exponentially many assignments. Instead we will build a new circuit on top of C such that the enumerator will be forced to reduce the set of assignments to polynomially many.

Consider any two gates g and h of C. We can run the enumerator to obtain k candidates for the value computed by each on input x. Let the corresponding lists of candidates be $a = \{a_1, \ldots, a_k\}$ and $b = \{b_1, \ldots, b_k\}$. (We may clearly assume that all values on each list are distinct. If not, we can just remove the duplicates and replace them by arbitrary distinct elements of the field that are not already on the list.) We will view g as a univariate degree $k-1$ polynomial $P_a(u) = \sum_{j=0}^{k-1} \beta_j u^j$ in $\mathbb{F}[u]$ such that $P_a(a_i) = d(i-1)$ for every $i \in [k]$, where d is a generator of an additive group of $\mathbb{F}$ of size at least k^2. Recall that the assumption is that $\mathbb{F}$ has characteristic at least k^2. The multiplicative identity 1 of $\mathbb{F}$ is such a generator, so we can take $d = 1$. Similarly define P_b for h. Thus each of the polynomials is given by its value on a set of k distinct points in $\mathbb{F}$ specified by the enumerator. Since both polynomials are of degree $k-1$, the coefficients (we will follow P_a) can be recovered by interpolation as the solution to the $k \times k$ Vandermonde system

$$\begin{pmatrix} a_1^0 & \cdots & a_1^{k-1} \\ a_2^0 & \cdots & a_2^{k-1} \\ \vdots & \ddots & \vdots \\ a_k^0 & \cdots & a_k^{k-1} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{k-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ k-1 \end{pmatrix}.$$

Since $a_1, \ldots, a_k$ are distinct, the system has full rank over $\mathbb{F}$. If k is a constant independent of n, we can find the solution using a circuit of constant size. When k and $|\mathbb{F}|$ are polynomial in n (i.e., our instance of the circuit-value problem is over a field that depends on the size of the instance), we can solve such systems in logspace uniform TC^0 [10]. (See also [12, 22].) The proofs there are for fields of characteristic 2, but extensions to finite fields of characteristic other than 2 are straightforward. The main parameter defining the depth and uniformity is n.

Let us return to our gates g and h with associated polynomials $P_a(u)$ and $P_b(v)$ (whose coefficients we now know). We will create a new gate f with inputs g and h. The function computed by f will be $F(u,v) = kP_a(u) + P_b(v)$. Thus $F(a_i, b_j) = k(i-1)+(j-1)$ for every $i, j \in [k]$. (Recall that we always implicitly have the multiplicative identity of $\mathbb{F}$ as a factor.) If we run the enumerator on $\langle f, x\rangle$, we will get back k candidates for $f(x)$, each corresponding to a *unique* combination of claimed values for $g(x)$ and $h(x)$, provided that $k^2 \leq p$, where p is the characteristic of $\mathbb{F}$. In particular, if the enumerator claims that $f(x) = c$, then c must be of the form $k(i-1) + (j-1)$ for some $i, j \in [k]$, yielding the claim that $g(x) = a_i$ and $h(x) = b_j$. Otherwise, $f(x) \neq c$.

Recall that we want to substantially reduce the total number of gate assignments to the original gates of C, and we do this as follows: First, we group the gates of C in pairs (arbitrarily) constructing a new level of gates as described above. We repeat the same procedure for the new level; continuing this way, we eventually obtain a single gate (after having constructed a full binary tree on top of the original gates). We run the enumerator on this gate, obtaining k candidates for its value on input x. Each candidate value corresponds, via a downward chain of decodings, to a unique gate assignment for the original gates of C. As long as k is polynomial in n, we can verify all candidate assignments in parallel, singling out the correct one. The value of $C(x)$ is the value assigned to the output gate.

We now describe the structure of the circuit that does this. Our parallel algorithm will first run the enumerator on every gate g_i in C. Each enumerator gate takes a description of C with a marked output gate g_i, together with the original input x, and outputs a list of k numbers, one of which is $g_i(x)$. Let us return to our pair of gates g and h with lists $(a_1, \ldots, a_k)$ and $(b_1, \ldots, b_k)$. All other pairs are treated in the same way in parallel. We feed the lists and the original input x to auxiliary circuitry $\textsc{Aux}(g,h)$ that outputs a description of a new circuit with the output gate f. (The new circuit contains the original C plus the circuitry that has already been built on top of it. The input to the new circuit will be the original input x.) $\textsc{Aux}(g,h)$ reconstructs the polynomials P_a, P_b, and uses them to compute a description of f. It follows from the above discussion that $\textsc{Aux}(g,h)$ can be implemented in NC^0 if k is constant, and in TC^0 if p^q (and thus k) are polynomial in n. (Given P_a and P_b, the description of f can clearly be computed by NC^0 and TC^0 circuits respectively.) Since there are $O(\log n)$ levels, the entire construction is in NC^1 (respectively in TC^1). Note that the output of an NC^0 circuit can depend only on constantly many inputs. Thus, improving the complexity significantly in the case of non-constant k appears impossible. Once we get to a single gate, we run the enumerator on this gate, obtaining k candidates for its value on x. Each value can be uniquely passed back to the original gates of C. Here, "passing" amounts to finding i and j from the value of $k(i-1)+(j-1)$, which can obviously be done in NC^0 or TC^0, depending on whether k is constant. ∎

The proof implies the following corollary.

Corollary 1 If the number of elements in $\mathbb{F}$ is polynomial in n, Theorem 1 holds with TC^1 in place of NC^1.

A Note on Representation If the number of elements in $\mathbb{F}$ is polynomial in n, we allow to supply $\mathbb{F}$ as a part of the input by its characteristic p, order p^q (in unary so that p^q is polynomial in n), and a monic irreducible polynomial $g \in \mathbb{Z}_p[x]$ of degree q. Note that the characteristic of $\mathbb{F}$ is *small* in the sense that the parallel complexity is measured in terms of $\log p$, rather than $\log \log p$. If $\mathbb{F}$ is in some specific form, g can be precomputed. Unfortunately, such explicitly given irreducible polynomials are rare and strongly depend on the structure of p^q. See [26, Chapter 2] and [18, Chapter 3] for excellent expositions. If g is not given, we could search for it exhaustively by going through the monic polynomials of degree n, in logspace, until an irreducible is found. Testing irreducibility, however, is a bottleneck, as it is currently in $\mathrm{NC}^{2.5}$ (again, for fields of small characteristic) [13].

4 Main Theorem

We now turn to arbitrary finite fields. The proof uses ideas underlying Sivakumar's result that SAT is not $O(\log n)$-membership comparable unless Unique-SAT is in P [27].

Theorem 2 Let $\mathbb{F}_q$ be a finite field with q elements, and let $i \geq 3$ be an integer. If $\mathrm{CVP}_{\mathbb{F}_q}$ is $(q-1)$-enumerable in NC^i then $\mathrm{CVP} \in \mathrm{NC}^i$.

Remark. To avoid inessential complications with the presentation (in particular, to simplify the Boolean complexity of field operations), we will take $q = 2^r$. The result readily holds for any structure of $\mathbb{F}$. Similarly to Theorem 1, the result also holds for fields that are not fixed but are given as a part of the input.

Proof: Let $\langle C, x\rangle$ be the instance of the (Boolean) circuit-value problem whose solution we wish to find, and let $g_0, \dots, g_{n-1}$ denote the gate values of C on input x. We view the gate values of C on x as the degree $(n-1)$ univariate polynomial $P_g(u) = \sum_{i=0}^{n-1} g_i u^i$ with coefficients from GF(2). Our goal will be to get some information about the value of this polynomial on all elements of an appropriately constructed field GF(2^m), where m is approximately $\log n$ (to be chosen later). Having the power of NC^3, we can find a generating polynomial exhaustively by cycling through monic polynomials of degree m in $\mathbb{Z}_2[x]$ until an irreducible is found, and construct GF(2^m) as the ring of polynomials over $\mathbb{Z}_2$ modulo this polynomial. Alternatively, if m is on the form $2 \cdot 3^l$ for some positive integer l, we can take $y^m + y^{m/2} + 1$ as the generating polynomial [19]. Each polynomial in GF(2^m) can also be represented as an m-bit string, treating bits as the corresponding coefficients.

We will use our $(q-1)$-enumerator to reduce the number of candidates for $P_g(u)$ for each $u \in$ GF(2^m), and then use the technique of Ar et al. [2] (as in [27]) to reconstruct the coefficients of P_g, which are the desired gate values of C on x.

Fix $u \in$ GF(2^m). The value of $P_g(u)$ is an element of GF(2^m), and thus has an m-bit representation. We will use the enumerator to eliminate one possible setting for each subset of r bits of $P_g(u)$ by disguising this question as an instance of CVP$_{\mathbb{F}_q}$ whose solution (viewed as an r-bit string) gives the desired r bits of $P_g(u)$. We will then use Sauer's lemma (as in [27]) to argue that this rules out a large portion of all possible settings for all the m bits.

To encode r-element subsets as instances of CVP$_{\mathbb{F}_q}$, we will need a bit version of CVP$_{\mathbb{F}_q}$, consisting of tuples $\langle D, y, i\rangle$ such that D is an encoding of an arithmetic circuit, y is an input to D, both D and y are over $\mathbb{F}_q$, and the i-th bit of $D(y)$ is 1, when $D(y)$ is viewed as an r-bit string. It is readily seen that the bit version is complete for P. Consider the set

$$A = \{\langle C, x, u, i\rangle \mid \text{the } i\text{-th bit of } P_g(u) \text{ is } 1\},$$

where C, x, u, and $P_g(u)$ are defined as above. Clearly A is in P, since the required GF(2^m)-arithmetic needed to compute any bit of $P_g(u)$ can be easily implemented in NC^1. Thus A many-one reduces (via some NC^1-computable function h) to the bit version of CVP$_{\mathbb{F}_q}$. We will use h to map each bit of $P_g(u)$ to an instance of the bit version of CVP$_{\mathbb{F}_q}$. Our parallel algorithm will first run h (in parallel) on all tuples $\langle C, x, u, i\rangle$ for i from 0 to $m-1$. Given $\langle C, x, u, i\rangle$, h produces an encoding of some circuit D_i together with an input y_i, both over $\mathbb{F}_q$, and a number j_i such that the i-th bit of $P_g(u)$ is precisely the j_i-th bit of D_i on y_i.

Let us fix a subset S of r bits of $P_g(u)$ and turn to the task of eliminating a possible setting of these bits. Denote the r-bit string corresponding to $P_g(u)$ restricted to the bits in S by $w = w_0 \dots w_{r-1}$. Notice that the bits of w are essentially the bits j_i of $D_i(y_i)$, where i ranges over the bits specified by S. We will view these output bits (denoted $w_0, \ldots, w_{r-1}$ to avoid new notation) as specifying the coefficients of the polynomial $P_w(x) = \sum_{i=0}^{r-1} w_i x^i$ over GF(2). Here x is our original input in the Boolean instance of CVP that we are solving. We view $\mathbb{F}_q$ as $\mathbb{Z}_2[z]/(h)$, the ring of polynomials over $\mathbb{Z}_2$ modulo the irreducible polynomial h of degree r in $\mathbb{Z}_2[z]$. (We can find h by exhaustive search.) Thus P_w is an element of $\mathbb{F}_q$. We will construct an arithmetic circuit D_S with an input y_S (both over $\mathbb{F}_q$) such that the value of D_S on y_S, viewed as a polynomial in $\mathbb{Z}_2[z]/(h)$, is equal to P_w. Since the degree of P_w is $r-1$, $D_S(y_S)$ uniquely defines P_w. In fact, we already have such D_S and y_S at hand. Our y_S will be the collection of inputs $\{y_i \mid i \in S\}$, and D_S will contain the circuitry $\{D_i \mid i \in S\}$. Each $D_i(y_i)$ (which is an r-bit number) encodes the corresponding coefficient of P_w as its j_i-th bit. The circuit D_S will output these r bits as an r-bit representation of its output (which is an element of $\mathbb{F}_q$). The circuitry needed to do such compilation of outputs is certainly in logspace uniform NC^1. We can finally run the $(q-1)$-enumerator for $\mathrm{CVP}_{\mathbb{F}_q}$ on $\langle D_S, y_S \rangle$ to get a list of $q-1$ values in $\mathbb{F}_q$. We can similarly eliminate one possible setting of any other subset of r bits of $P_g(u)$. Not only can we do so for all sets S in parallel (u has been fixed up until now), but also for all $u \in \mathrm{GF}(2^m)$. Recall that $r = c \log n$, and $m = d \log n$ with $d > c$ to be fixed later. There are 2^m many u's, and $\binom{m}{r} < 2^m = n^{O(1)}$ many subsets for each such u.

How many possible bit settings for each $P_g(u)$ does this procedure rule out? This number follows immediately from (a rather special case of) a lemma proven independently by Sauer [24], Perles and Shelah [25], and Vapnik and Chervonenkis [28]. (For an excellent exposition, including the lemma's proof, see Bollobás [6, p.131].) To state the lemma, we will say that a set $I \subset [m]$ is *traced* by a set system $\mathcal{F} \subset \mathcal{P}(m)$ if the collection of intersections $\{F \cap I \mid F \in \mathcal{F}\}$ contains every element of $\mathcal{P}(I)$. The lemma states that any $\mathcal{F}$ traces at least $|\mathcal{F}|$ sets in $\mathcal{P}(m)$. In particular, this says that if $|\mathcal{F}| > \sum_{i=0}^{r-1} \binom{m}{i}$, then $\mathcal{F}$ must trace some set of size r. For us, $\mathcal{F}$ is the set of the remaining possible assignments to the m bits of $P_g(u)$. We know that it does not trace any subset of r bits (since we excluded at least one possible intersection for each such subset), thus the contrapositive of the lemma says that we are left with at most $\sum_{i=0}^{r-1} \binom{m}{i} < r\binom{m}{r}$ possibilities for $P_g(u)$ (out of 2^m such possibilities). Likewise for every other $u \in \mathrm{GF}(2^m)$.

Let $a = d/c$. We use a bound on $\binom{m}{r} = \binom{m}{m/a}$ of the form: $\binom{m}{m/a} < C(a) \cdot (ea)^{n/a}$, where $C(a)$ is some constant that depends on a. A number of different estimates of this form are known; which one is used does not affect the result. We want $k \doteq 2^{m/4} > C(a) \cdot (ea)^{n/a}$, so given c, we take $d = ac > 4$ to be any easily computable constant such that a satisfies the condition above.

It remains to reconstruct the coefficients of P_g using the restricted lists of possibilities for its value on a set of 2^m points. As in [27], we will use the reconstruction technique of Ar et al. [2], which reduces to the factorization of an appropriately constructed bivariate polynomial $Q(u, v)$ which is zero at all points (u, v) such that $u \in \mathbb{F}_q$ and v is in the list of possibilities for $P_g(u)$. The key claim is that $(v - P_g(u))$ must appear as an irreducible factor of Q. The degree of Q will be polynomial in n in both u and v. The factorization thus gives a list of polynomially many candidates for P_g, each corresponding to a gate assignment for C on x. All of them can be verified in parallel in NC^1, and the correct one can be singled out.

The statements regarding Q need some justification. For completeness (and since we need to bound the parallel complexity rather than sequential time), we sketch the technique of Ar et al. [2].

Reconstruction Let $\mathbb{F} = \mathrm{GF}(2^m)$ be the extension field. We will construct a bivariate non-zero polynomial $Q(u, v)$ of degree $\deg(u)$ in u and $\deg(v)$ in v (to be fixed momentarily), such that $Q(u, v) = 0$ for all $u \in \mathbb{F}$ and all v in the list of k possibilities for $P_g(u)$, and such that both u and v are present in every monomial of Q. This gives a homogeneous system of $k2^m$ linear equations in $\deg(u) \cdot \deg(v)$ unknowns. If $\deg(u) \cdot \deg(v) \geq k2^m$, we are guaranteed a solution. It remains to show that $v - P_g(u)$ will appear as an irreducible factor. Consider the univariate polynomial $Q_g(u) = Q(u, P_g(u))$, which is of degree at most $\deg(u) + n\deg(v)$ in u. We know that $Q_g(u)$ is 0 on every $u \in \mathrm{GF}(2^m)$. Thus if $\deg(u) + n\deg(v) < 2^m$, $Q_g(u)$ must be identically zero, implying that $(y - P_g(u))$ divides $Q(u, v)$. Let us choose $\deg(v)$ and $\deg(v)$ leading to this desired situation (for sufficiently large n). Recall that $k = m2^{m/4}$. We have to satisfy $\deg(u)\deg(v) \geq m2^{5m/4}$ and $\deg(u) + 2^{m/d}\deg(v) < 2^m$, which can be done by setting, for example, $\deg(u) = \deg(v) = 2^{2m/3}$.

Note, crucially, that $\mathbb{F}$ has polynomially in n many elements, and thus its characteristic p is small, i.e., we need the algorithms to run in depth polynomial in $\log p$ (rather than $\log\log p$). In what follows, n (the length of the main input) determines the complexity and uniformity of circuits. All circuits are logspace uniform. Univariate polynomials of degree $n^{O(1)}$ over $\mathbb{F}$ (not necessarily square-free) can be factored in NC^3. A deterministic version of the univariate factoring algorithm of von zur Gathen [29], obtained by plugging in the deterministic NC^2 matrix rank algorithm of Mulmuley [20], works in depth $O(\log^2 n \log(np))$ (bounded fan-in, polynomial time). Thus if p is polynomial in n, we are in NC^3 (as noted in [31]). Bivariate polynomial factoring can be reduced to univariate factoring using Hensel's lifting. Let $f(x, y) \in \mathbb{F}[x, y]$ be the polynomial that we wish to factor into irreducible factors. If $\mathbb{F}$ is a unique factorization domain (which it is, since $\mathbb{F}$ is a field), then so are $\mathbb{F}[x]$ and $\mathbb{F}[x, y]$, thus the factorization is unique (up to the order of factors). We will only seek some irreducible factor of f, hence we will then be able to compute the complete factorization in parallel for all factors.

Let the degree of f be at most $d = n^{O(1)}$ in each variable. We seek an irreducible factor $g \in \mathbb{F}[x, y]$ of f. Hensel's lifting extracts information about the factors of $f(x, y)$ by lifting up a univariate factorization of $f_0(x) = f(x, 0)$. Factoring the univariate polynomial $f_0(x) = g(x, 0)h(x, 0)$ is equivalent to factoring $f(x, y) = g(x, y)h(x, y) \bmod y$. Hensel's construction will lift this factorization to the one modulo $y^2, y^{2\cdot 2}$, and so on, until we have a factorization modulo y^{2^t}, where t is sufficiently large to yield the true factorization of $f(x, y)$. Due to space limitations, we will not describe the reduction. (See, for example, a book by von zur Gathen and Gerhard [30].) We only need to show that it is in logspace uniform NC^3. The univariate factorization algorithm runs in NC^3. The cost of each lifting step is essentially the cost of computing the quotient and remainder of polynomials of degree at most d in $\mathbb{F}[x]$, and is in NC^1 [12], putting $t = O(\log n)$ iterations in NC^2. Once we have a factorization $f(x, y) \equiv g_t(x, y)h_t(x, y) \pmod{y^{2^t}}$, we need to find polynomials $G(x, y)$ and $D(x, y)$ satisfying $G(x, y) \equiv g_t(x, y)D(x, y) \pmod{y^{2^t}}$, as well as certain conditions on the degrees. This step involves solving a system of linear equations with unknowns being the coefficients of $G(x, y)$ and $D(x, y)$, which is in NC^2 [7]. If $\gcd(G, f)$ is non-trivial, G is a factor of f; otherwise f is irreducible. The gcd computation is in NC^2 [7]. The computation of all factors (originating from different ways to factorize $f_0(x)$ as the product of a monic irreducible $g_0(x)$ and a relatively prime $h_0(x)$) can be done in parallel. ∎

Corollary 2 Let $\mathbb{F}_q$ be the finite field with q elements, and let $i \geq 3$ be an integer. If $\mathrm{CVP}_{\mathbb{F}_q}$ is $(q-1)$-enumerable in NC^i then $\mathrm{P} = \mathrm{NC}^i$. The field $\mathbb{F}_q$ can be given by q (in unary notation) as a part of the input.

Probabilistic Enumerators Theorem 2 holds in the randomized setting when the enumerator is allowed to err with small probability. For the definition of randomized Boolean circuits and the class RNC, the reader is referred to Cook [11]. It is believed highly unlikely that $\mathrm{P} \subseteq \mathrm{RNC}$. [2]

An enumerator E_f for a function f is said to have *error* $\epsilon(\cdot)$ if for all inputs x in the domain of f, the set $E_f(x)$ includes $f(x)$ with probability at least $1 - \epsilon(|x|)$, where the probability is taken over the random bits fed to E_f. Here $\epsilon : \mathbb{N} \to (0, \frac{1}{2})$ is some monotone non-increasing function.

Theorem 3 (Randomized analogue of Theorem 2) Let $\mathbb{F}_q$ be as in Theo-

[2] Note, however, that randomized cicuits inherently have multiple access to randomness (while randomized space-bounded machines do not), and RNC^1 is not known (or believed) to be contained in randomized logspace.

rem 2. There is no $(q-1)$-enumerator for $\mathrm{CVP}_{\mathbb{F}_q}$ with error at most inversely polynomial unless $\mathrm{P} \subseteq \mathrm{RNC}^3$.

Proof: As in the deterministic case, we will attempt to obtain restricted lists of candidates for the value of the gate polynomial P_g on the elements of the extension field $\mathrm{GF}(2^m)$. Let $\epsilon(n) = n^{-\Omega(1)}$ be the error of the enumerator, and fix $u \in \mathrm{GF}(2^m)$. What is the probability that the list of candidates for $P_g(u)$ excludes $P_g(u)$? No more than the probability that the enumerator errs on at least one of all r-element subsets of the m bits of $P_g(u)$, and this is at most $\binom{m}{r}\epsilon(s(n))$, where $s(n)$ is the polynomial bounding the encoding length of $\mathrm{CVP}_{\mathbb{F}_q}$ instances that the subsets were mapped to by the reduction function in the proof of Theorem 2. We obviously want $\epsilon(s(n)) \leq (\frac{1}{2}-\delta)\cdot\binom{m}{r}^{-1} = n^{-\Omega(1)}$ for some $\delta > 0$ (δ can be as small as inverse polynomial in n). This can be satisfied by appropriately altering m by a constant factor (which we have control over, and which does not affect the proof in any significant way). We can then make the probability that the list excludes $P_g(u)$ vanish exponentially in the input length at the price of a polynomial increase in parallelism. Indeed, we can run polynomially many instances of the enumerator in parallel and exclude the majority vote. Chernoff bounds can be used in the straightforward way to bound the probability that $P_g(u)$ is excluded (by an inverse exponential). Hence for every $u \in \mathrm{GF}(2^m)$, we have a list of at most k values (in the notation of Theorem 2), one of which is $P_g(u)$ with probability at least $1-\gamma$ for some inverse exponential γ, implying that the expected number of bad lists is at most $\gamma 2^m$. By Markov's inequality, the probability that there are more than $3 \cdot \gamma 2^m$ such lists is at most $1/3$. Assume from now on that there are in fact no more than $3 \cdot \gamma 2^m$ bad lists. As before, we will attempt to find a polynomial $Q(u, v)$ which is zero on all $u \in \mathrm{GF}(2^m)$ and all v in the list of possibilities for $P_g(u)$. The only difference is that in order to guarantee that $v - P_g(u)$ is an irreducible factor, we need to satisfy the constraint $\deg(u) + 2^{m/d}\deg(v) < 2^m(1-3\gamma)$. This would yield that $Q(u, P_g(u))$ has more zeros than its degree, and thus must be identially zero, implying that $v - P_g(u)$ divides $Q(u, v)$. Recall that $k = m2^{m/4}$. We have to satisfy the constraints $\deg(u)\deg(v) \geq m2^{5m/4}$ and $\deg(u) + 2^{m/d}\deg(v) < 2^m(1-3\gamma)$, and our choice $\deg(u) = \deg(v) = 2^{2m/3}$ does so (for sufficiently large n). Thus, finding Q and factoring it gives a small list of polynomials that includes P_g. We know how to single out P_g. This completes the proof. ∎

Acknowledgements

We would like to thank Joachim von zur Gathen for referring us to the results on the parallel complexity of polynomial factorization. We would also like to thank Eric Allender for pointing out an alternative interpretation of the results in terms of resource-bounded Kolmogorov complexity. This work was supported in part by NSF grants EIA-0080124 and EA-0205061.

References

[1] A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query classes and nonuniform complexity. In the *5th Structure in Complexity Theory Conference*, pages 232–243, 1990.

[2] S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):487–510, 1998.

[3] R. Beals, R. Chang, W. Gasarch, and J. Torán. On finding the number of graph automorphisms. *Chicago Journal of Theoretical Computer Science*, 1999(1).

[4] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Information and Computation*, 120(2):304–314, 1995.

[5] A. Beygelzimer and M. Ogihara. The (non)enumerability of the determinant and the rank. *Theory of Computing Systems*, 36(4): 359–374.

[6] B. Bollobás. *Combinatorics*. Cambridge University Press, 1986.

[7] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52:241–256, 1982.

[8] J. Cai and L. Hemachandra. Enumerative counting is hard. *Information and Computation*, 82(1):34–44, 1989.

[9] J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215–219, 1991.

[10] J. Cai and D. Sivakumar. Resolution of Hartmanis' conjecture for NL-hard sparse sets. *Theoretical Computer Science*, 240:257–269, 2000.

[11] S. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[12] W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18(5):955–976, 1989.

[13] G. Frandsen. Parallel Construction of Irreducible Polynomials. *Technical Report DAIMI-PB-358*, University of Aarhus, Denmark, 1991.

[14] J. Goldsmith, M. Ogihara, J. Rothe. Tally NP Sets and Easy Census Functions. *Inf. Comput.*, 158(1): 29–52, 2000.

[15] R. Greenlaw, H. Hoover, and W. Ruzzo. *Limits to parallel computation*. Oxford University Press, NY, 1995.

[HR90] L. Hemachandra and S. Rudich. On the complexity of ranking. *Journal of Computer and System Sciences*, 41:251–271, 1990.

[16] E. Kaltofen. Polynomial factorization, 1987–1991. In I. Simon, editor, LATIN 92, *Lecture Notes in Computer Science*, 583:294–313, 1992.

[17] R. Ladner. The circuit value problem is log space complete for P. *ACM SIGACT News*, 7(1):18–20, January 1975.

[18] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, 1994.

[19] J. van Lint. *Introduction to coding theory*. Springer-Verlag, 1991.

[20] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7:101–104, 1987.

[21] M. Ogihara. Polynomial-time membership comparable sets. *SIAM Journal on Computing*, 24(5):1068–1081, 1995.

[22] F. Preparata. Inverting a Vandermonde matrix in minimum parallel time. *Information Processing Letters*, 38:291–294, 1991.

[23] W. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

[24] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory (A)*, 13:145–147, 1972.

[25] S. Shelah. A combinatorial problem: Stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41:247–261, 1972.

[26] I. Shparlinski. Computational and Algorithmic Problems in Finite Fields. *Kluwer Academic Publishers*, 1992.

[27] D. Sivakumar. On membership comparable sets. *Journal of Computer and System Sciences*, 59(2):270–280, 1999.

[28] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Prob. Appl.*, 16:264–280, 1971.

[29] J. von zur Gathen. Parallel algorithms for algebraic problems. *SIAM Journal on Computing*, 13:802–824, 1984.

[30] J. von zur Gathen and J. Gerhard. Modern Computer Algebra. *Cambridge University Press*, Cambridge, 1999.

[31] J. von zur Gathen and G. Seroussi. Boolean circuits versus arithmetic circuits. *Information and Computation*, 91:142–154, 1991.

[32] C. Wilson. Relatizived circuit complexity. *Journal of Computer and System Sciences*, 31(2):169–181, 1985.