Graph Nonisomorphism Has Subexponential Size Proofs Unless The Polynomial-Time Hierarchy Collapses

Adam R. Klivans*
Department of Mathematics, MIT
Cambridge, MA 02139
klivans@math.mit.edu

Dieter van Melkebeek†
Department of Computer Science
The University of Chicago
Chicago, IL 60637
dieter@cs.uchicago.edu
http://www.cs.uchicago.edu/~dieter

Abstract

We establish hardness versus randomness trade-offs for a broad class of randomized procedures. In particular, we create efficient nondeterministic simulations of bounded round Arthur-Merlin games using a language in exponential time that cannot be decided by polynomial size oracle circuits with access to satisfiability. We show that every language with a bounded round Arthur-Merlin game has subexponential size membership proofs for infinitely many input lengths unless the polynomial-time hierarchy collapses. This provides the first strong evidence that graph nonisomorphism has subexponential size proofs.

We set up a general framework for derandomization which encompasses more than the traditional model of randomized computation. For a randomized procedure to fit within this framework, we only require that for any fixed input the complexity of checking whether the procedure succeeds on a given random bit sequence is not too high. We then apply our derandomization technique to four fundamental complexity theoretic constructions:

- The Valiant-Vazirani random hashing technique which prunes the number of satisfying assignments of a Boolean formula to one, and related procedures like computing satisfying assignments to Boolean formulas *non-adaptively* given access to an oracle for satisfiability.
- The algorithm of Bshouty et al. for learning Boolean circuits.
- Constructing matrices with high rigidity.
- Constructing polynomial-size universal traversal sequences.

We also show that if linear space requires exponential size circuits, then space bounded randomized computations can be simulated deterministically with only a constant factor overhead in space.

*Supported in part by NSF grant CCR-9701304.

†Supported in part by the NSF through grants CCR 92-53582 and CCR 97-32922, by the European Union through TMR grant ERB-4001-GT-96-0783 while visiting CWI and the University of Amsterdam, and by the Fields Institute while visiting the Fields Institute at the University of Toronto.

1 Introduction

The power of randomness in computation is a fundamental area of study in computer science. We know of many examples where "flipping a coin" facilitates algorithm design. From a complexity theoretic point of view, however, the merits of randomization remain unclear. Can we always eliminate the use of random bits without substantially increasing the need for other resources?

Blum and Micali [BM84] and Yao [Yao82] gave a partial answer to this question. They realized that if some of the seemingly hard algorithmic problems really are computationally intractable then, in certain settings, randomness cannot help much. Nisan and Wigderson [NW94] established a range of hardness versus randomness trade-offs. They showed how to use any language in exponential time that is nonuniformly hard in an average-case sense to construct a pseudo-random generator that fools circuits of polynomial size. They obtained nontrivial derandomizations of polynomial-time randomized decision algorithms under average-case hardness assumptions and even deterministic polynomial-time simulations under the strongest of their hypotheses. Babai, Fortnow, Nisan, and Wigderson [BFNW93] and Impagliazzo and Wigderson [IW97] relaxed the hardness condition from average-case to worst-case. As a corollary, they showed how to simulate every polynomial-time randomized decision algorithm deterministically in subexponential time for infinitely many input lengths unless the polynomial-time hierarchy collapses [BFNW93].

The authors of [NW94], [BFNW93], and [IW97] used their techniques to derandomize traditional models of randomized computation, most notably BPP. There is no reason, however, to restrict ourselves to these models. We show how to apply the techniques used to derandomize BPP to more general models of randomized computation. The key observation we make is that the reductions proved in [NW94], [BFNW93], and [IW97] are "black-box", i.e., they relativize. More specifically, the above papers show how to transform any small circuit that distinguishes the output of a certain pseudo-random generator from the uniform distribution into a small circuit that computes a function used to build the pseudo-random generator. We observe that these reductions work for *any* nonuniform model of computation that satisfies certain closure properties, and in particular for oracle circuits given any fixed oracle. Thus, in order to build a pseudo-random generator that looks random to any small B-oracle circuit, we need only assume the existence of a function that cannot be computed by small B-oracle circuits.

The above observations allow us to apply the classical hardness versus randomness results to various settings, in particular to the nondeterministic setting of Arthur-Merlin games [Bab85]. The class AM of languages with bounded round Arthur-Merlin games forms a randomized extension of NP. The most notable problem in AM not known to be in NP is Graph Nonisomorphism [GMW91, GS89]. Derandomizing AM requires security against nondeterministic adversaries. Rudich [Rud97] pointed out that pseudo-random generators in the traditional cryptographic setting where an adversary has more resources than the generator cannot hope to have this property because the adversary can always guess the seed and check. From a derandomization perspective, however, the Nisan-Wigderson type of pseudo-random generator does not suffer from this drawback, as adversaries in this setting do not have the resources to run the generator — even if they correctly guess the seed.

We give evidence that AM coincides with NP. More specifically, we show that the existence of an exponential-time decidable language with high worst-case nonuniform SAT-oracle complexity implies nontrivial derandomizations of AM. The trade-offs are presented in Table 1, where we use C^B to denote circuit complexity given access to oracle B. See Section 2 for precise definitions. The parameter ℓ in Table 1 can be any space constructible function, the interesting range lying between logarithmic and subpolynomial, e.g., polylogarithmic.

hardness assumption:	derandomization consequence:
$\exists\, f \in \mathrm{NEXP} \cap \mathrm{coNEXP} : C_f^{\mathrm{SAT}}(n) \in n^{\omega(1)}$	$\mathrm{AM} \subseteq \cap_{\epsilon>0}\mathrm{NTIME}[2^{n^\epsilon}]$
$\exists\, f \in \mathrm{NEXP} \cap \mathrm{coNEXP} : C_f^{\mathrm{SAT}}(\ell(n)) \in \Omega(n)$	$\mathrm{AM} \subseteq \cup_{c>0}\mathrm{NTIME}[2^{(\ell(n^c))^{O(1)}}]$
$\exists\, f \in \mathrm{NE} \cap \mathrm{coNE} : C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$	$\mathrm{AM} = \mathrm{NP}$

Table 1: Hardness versus randomness trade-offs for AM.

If the hardness condition on the left-hand side of Table 1 holds for infinitely many input lengths, then the corresponding derandomization on the right-hand side works for infinitely many input lengths. We refer to the *weak* version of Table 1 when we assume the above hardness conditions only hold for infinitely many input lengths. As in [NW94], [BFNW93], and [IW97], we typically state our theorems assuming the hardness conditions are true for every input length. Both interpretations hold for all of our results.

We can view the assumptions in Table 1 as statements concerning the relationships among computation, nonuniformity, and nondeterminism. For example, the third entry in the table states that if nonuniformity and nondeterminism cannot significantly speed up computation then we can derandomize AM. We point out that if the hardness assumption in the first row of the weak version of Table 1 fails, then the polynomial-time hierarchy collapses to the third level.

Arvind and Köbler [AK97] obtained similar results to those in Table 1 using nondeterministic circuits, but needed average-case hardness assumptions instead of worst-case. As opposed to oracle circuits with access to SAT, nondeterministic circuits do not seem to have the closure properties that allow us to relax the hardness hypothesis from average-case to worst-case. Bridging this gap is crucial. It lets us conclude from the weak version of Table 1 that every language in AM, and graph nonisomorphism in particular, has subexponential size proofs for infinitely many input lengths unless the polynomial-time hierarchy collapses.

Our simulations of AM are a special case of an "all-purpose" derandomization tool which applies to any randomized process for which we can efficiently check the successfulness of a given random bit sequence. We formally define the notion of a success predicate in Section 4. If we can decide the success predicate of a randomized process with polynomial size B-oracle circuits, then the hardness assumption on the left-hand side of Table 2 provides a pseudo-random generator G with the characteristics on the right-hand side of Table 2 for derandomizing the process. The symbol A in Table 2 represents an arbitrary class of oracles.

hardness assumption:	complexity of G:	seed length:
$\exists\, f \in \mathrm{EXP}^A : C_f^B(n) \in n^{\omega(1)}$	EXP^A	$O(n^\epsilon)$ for every $\epsilon > 0$
$\exists\, f \in \mathrm{EXP}^A : C_f^B(\ell(n)) \in \Omega(n)$	EXP^A	$O((\ell(n^c))^6)$ for some $c > 0$
$\exists\, f \in \mathrm{E}^A : C_f^B(n) \in 2^{\Omega(n)}$	E^A	$O(\log n)$

Table 2: Overview of pseudo-random generator constructions.

To illustrate the power of our generalization, we apply our technique to the following fundamental constructions from different areas of theoretical computer science.

- The Valiant-Vazirani random hashing procedure which prunes the number of satisfying assignments of a Boolean formula to one [VV86].
- Exact learning of circuits using equivalence queries and access to an NP oracle [BCG+96].

- The construction of matrices with high rigidity [Val77].
- The construction of polynomial-size universal traversal sequences [AKL+79].

Elaborating on the first application, given a Boolean formula ϕ, we can construct in subexponential time a collection of polynomial-size formulas with the Valiant-Vazirani property with respect to ϕ unless the polynomial-time hierarchy collapses. If there exists a language in E with nonuniform SAT-oracle complexity $2^{\Omega(n)}$ we achieve a polynomial time deterministic procedure. It follows that, under the same hypothesis, we can find in polynomial time a satisfying assignment for a Boolean formula given nonadaptive access to SAT as opposed to the standard adaptive method of binary search. We obtain derandomization results of a similar kind for our three other examples. See Section 5 for the precise statements.

Regarding universal traversal sequences, we obtain polynomial-time constructions under the assumption that there exists a language in E with nonuniform complexity $2^{\Omega(n)}$. We also show that if there is a language in linear space that requires circuits of size $2^{\Omega(n)}$, then BPL = L, where BPL denotes the languages recognizable in randomized logspace with bounded two-sided error. This answers a question raised by Clementi, Rolim, and Trevisan [CRT98]. As a corollary, under the same hypothesis, we can generate universal traversal sequences in logspace.

1.1 Organization

Section 2 introduces our notation. In Section 3 we generalize the techniques used to derandomize BPP and establish how to use them in the Arthur-Merlin setting. Section 4 defines a broad class of randomized algorithms and shows how our approach allows us to reduce the randomness of any algorithm that fits within this class. In Section 5 we apply this framework to the four examples mentioned above. We give some conclusions in Section 6.

2 Notation

Most of our complexity theoretic notation is standard. We refer the reader to the textbooks by Balcázar, Díaz and Gabarró [BDG95, BDG90], and by Papadimitriou [Pap94].

An *oracle circuit* D is a circuit with AND, OR and NOT gates as well as oracle gates, which compute membership of the string formed by their input bits to some unspecified oracle B. The function D^B the circuit computes depends on the oracle B. For fixed B, we will use the term B-*oracle circuit* to denote an oracle circuit with access to B as an oracle. In this paper, we measure the *size* of a circuit by its number of connections.

Given a Boolean function $f : \{0,1\}^* \to \{0,1\}$ and an oracle B, the *circuit complexity* $C_f^B(n)$ of f at length n relative to B is the smallest integer t such that there is a B-oracle circuit of size t that computes f on inputs of length n. The *hardness* $H_f^B(n)$ of f at length n relative to B is the largest integer t such that for any oracle circuit D of size at most t with n inputs

$$|\Pr_x[D^B(x) = f(x)] - \frac{1}{2}| \leqslant \frac{1}{t},$$

where x is uniformly distributed over $\{0,1\}^n$.

A *pseudo-random generator* G is a sequence of functions $(G_n)_n$ such that G_n maps $\{0,1\}^{s(n)}$ to $\{0,1\}^n$ for some function $s : \mathbb{N} \to \mathbb{N}$ with $s(n) < n$. The function s is called the *seed length* of G. We say that G is *computable in* $\mathcal{C}$ if the problem of deciding the i-th bit of $G_n(\sigma)$ given $\langle n, \sigma, i\rangle$

belongs to $\mathcal{C}$. Given an oracle B, G is said to be *secure against* B if for almost all n and for any oracle circuit D of size at most n

$$|\Pr_{\rho}[D^B(\rho)=1] - \Pr_{\sigma}[D^B(G_n(\sigma))=1]| \leqslant \frac{1}{n}, \tag{1}$$

where ρ is uniformly distributed over $\{0,1\}^n$ and σ over $\{0,1\}^{s(n)}$.

For any function $t:\mathbb{N}\to\mathbb{N}$, AM-TIME$[t(n)]$ represents the class of languages L for which there exists a deterministic Turing machine M that runs in time $O(t(n))$ on inputs of the form $\langle x,y,z\rangle$ where $x\in\{0,1\}^n$ and $y,z\in\{0,1\}^{t(n)}$, such that for any input x,

$$x\in L \quad\Rightarrow\quad \Pr_{|y|=t(n)}[\exists z\in\{0,1\}^{t(n)} : M(\langle x,y,z\rangle)=1] \geqslant \frac{2}{3} \tag{2}$$

$$x\notin L \quad\Rightarrow\quad \Pr_{|y|=t(n)}[\exists z\in\{0,1\}^{t(n)} : M(\langle x,y,z\rangle)=1] \leqslant \frac{1}{3}, \tag{3}$$

where $n=|x|$ and the probabilities are with respect to the uniform distribution. AM denotes $\cup_{c>0}$AM-TIME$[n^c]$.

EXP $=\cup_{c>0}$DTIME$[2^{n^c}]$ and E $=\cup_{c>0}$DTIME$[2^{cn}]$. Similarly, NEXP $=\cup_{c>0}$NTIME$[2^{n^c}]$ and NE $=\cup_{c>0}$NTIME$[2^{cn}]$.

For any complexity class $\mathcal{C}$, the class i.o.-$\mathcal{C}$ consists of all languages L for which there is a language $L'\in\mathcal{C}$ such that $L\cap\{0,1\}^n = L'\cap\{0,1\}^n$ for infinitely many lengths n.

For any function $s:\mathbb{N}\to\mathbb{N}$, $\Omega(s)$ denotes the class of all functions $t:\mathbb{N}\to\mathbb{N}$ for which there exists a constant $\epsilon>0$ such that $t(n)\geqslant\epsilon\cdot s(n)$ for almost all n. $\omega(s)$ is the class of t such that for any $c>0$ and almost all n, $t(n)\geqslant c\cdot s(n)$.

3 Derandomizing Arthur-Merlin Games

In this section, we develop methods for derandomizing Arthur-Merlin games and give evidence that the class AM of languages with bounded round Arthur-Merlin games is not much larger than NP and may even coincide with it.

As is customary in the area of derandomization, our approach will be to construct pseudo-random generators with appropriate security properties. The following lemma states that to derandomize AM $=\cup_{c>0}$AM-TIME$[n^c]$, the pseudo-random generator need only be secure against SAT. See the Appendix for a proof.

Lemma 3.1 *Let $s:\mathbb{N}\to\mathbb{N}$ be a space constructible function, and $t,\tau:\mathbb{N}\to\mathbb{N}$ be time constructible functions. If there is a pseudo-random generator G computable in* NTIME$[\tau(n)]\cap$coNTIME$[\tau(n)]$ *and with seed length s that is secure against* SAT, *then*

$$\text{AM-TIME}[t(n)] \subseteq \text{NTIME}[2^{s(t'(n))}\cdot\tau(s(t'(n)))\cdot t(n)], \tag{4}$$

where $t'(n)\in O(t(n)\log^2 t(n))$.

In order to build such a pseudo-random generator, we will extend the work of [NW94], [BFNW93] and [IW97] to the nondeterministic setting of Arthur-Merlin games. The main construction in these papers is a reduction from a circuit that distinguishes the output of a pseudo-random generator based on f from the uniform distribution to a circuit capable of computing f. We argue that this construction works for any nonuniform model of computation satisfying certain closure properties, and in particular for B-oracle circuits for any fixed oracle B. In this way, we obtain pseudo-random generators secure against B from functions which B-oracle circuits cannot compute.

3.1 Subexponential Nondeterministic Simulations for AM

We begin with a generalized version of [NW94]:

Theorem 3.2 *Let $\mathcal{A}$ be a class of oracles, B an oracle, and $s : \mathbb{N} \to \mathbb{N}$ a space constructible function. For any Boolean function $g \in \mathrm{EXP}^{\mathcal{A}}$, there is a pseudo-random generator G computable in $\mathrm{EXP}^{\mathcal{A}}$ which is secure against B and has seed length s provided*

$$H_g^B(\sqrt{s(n)}) \geqslant n^2.$$

Proof Sketch
The constructions of Nisan and Wigderson [NW94] work for any nonuniform model which is closed under precomputation and complementation. In particular, they carry through for oracle circuits. □

The generalization of the main reduction implicit in [BFNW93] can be stated as follows:

Theorem 3.3 *Let $\mathcal{A}$ be a class of oracles and B an oracle. For any Boolean function $f \in \mathrm{EXP}^{\mathcal{A}}$, there is a Boolean function $g \in \mathrm{EXP}^{\mathcal{A}}$ such that*

$$H_g^B(n) \in \Omega\left(\sqrt[3]{C_f^B(\sqrt[3]{n})/n^2}\right).$$

Proof Sketch
The transformation in [BFNW93] only needs closure of the model of computation under majority, complementation, and certain arithmetic field operations. Since oracle circuits have these closure properties, we obtain the desired result. □

Combining Theorems 3.2 and 3.3 yields our main tool for obtaining subexponential simulations.

Theorem 3.4 *Let $\mathcal{A}$ be a class of oracles, B an oracle, and $s : \mathbb{N} \to \mathbb{N}$ a space constructible function. For any Boolean function $f \in \mathrm{EXP}^{\mathcal{A}}$, there is a pseudo-random generator G computable in $\mathrm{EXP}^{\mathcal{A}}$ which is secure against B and has seed length s provided*

$$C_f^B\left(\sqrt[6]{s(n)}\right) \in \omega(n^9).$$

In this section, we will apply Theorem 3.4 with $\mathcal{A} = \mathrm{NP} \cap \mathrm{coNP}$ and $B = \mathrm{SAT}$. Note that $\mathrm{EXP}^{\mathrm{NP}\cap\mathrm{coNP}} = \mathrm{NEXP} \cap \mathrm{coNEXP}$. Together with Lemma 3.1, Theorem 3.4 yields the following results:

Theorem 3.5 *If there is a Boolean function $f \in \mathrm{NEXP} \cap \mathrm{coNEXP}$ such that $C_f^{\mathrm{SAT}}(n) \in n^{\omega(1)}$, then*

$$\mathrm{AM} \subseteq \cap_{\epsilon>0}\mathrm{NTIME}[2^{n^\epsilon}].$$

Theorem 3.6 *If there is a Boolean function $f \in \mathrm{NEXP} \cap \mathrm{coNEXP}$, and a space constructible function $\ell : \mathbb{N} \to \mathbb{N}$ such that*

$$C_f^{\mathrm{SAT}}(\ell(n)) \in \Omega(n),$$

then there is a constant d such that

$$\mathrm{AM} \subseteq \cup_{c>0}\mathrm{NTIME}[2^{(\ell(n^c))^d}].$$

We can rephrase the weak version of Theorem 3.5 as:

Theorem 3.7 *If* $\mathrm{NEXP} \cap \mathrm{coNEXP} \not\subseteq \mathrm{P}^{\mathrm{NP}}/\mathrm{poly}$*, then* $\mathrm{AM} \subseteq \cap_{\epsilon>0}\text{i.o.-NTIME}[2^{n^\epsilon}]$.

So, if the conclusion of Theorem 3.7 fails to hold, then $\mathrm{NEXP} \cap \mathrm{coNEXP} \subseteq \mathrm{P}^{\mathrm{NP}}/\mathrm{poly}$, which implies that $\mathrm{EXP} = \Sigma_3^{\mathrm{p}} \cap \Pi_3^{\mathrm{p}}$. Therefore, we obtain:

Theorem 3.8 *If the polynomial-time hierarchy does not collapse, then every language in* AM*, and graph nonisomorphism in particular, has subexponential size proofs for infinitely many lengths.*

Theorem 3.6 yields a range of hardness versus randomness trade-offs for the various choices of the parameter ℓ. Since $C_f^B(n) \in O(2^n/n)$ always holds, the hypothesis of Theorem 3.6 cannot be met for ℓ sublogarithmic. On the other hand, the conclusion becomes trivial in the case where ℓ is polynomial. Therefore, the interesting range for ℓ lies between logarithmic and subpolynomial. For example, for ℓ polylogarithmic, Theorem 3.6 reads:

Theorem 3.9 *If there is a Boolean function* $f \in \mathrm{NEXP} \cap \mathrm{coNEXP}$ *such that* $C_f^{\mathrm{SAT}}(n) \in \Omega(2^{n^\epsilon})$ *for some* $\epsilon > 0$*, then every language in* AM *has quasipolynomial size proofs.*

3.2 Polynomial Nondeterministic Simulations for AM

Theorem 3.6 is not powerful enough to yield a complete derandomization — even if ℓ is logarithmic, we only obtain quasipolynomial simulations. However, we can strengthen Theorem 3.6 for logarithmic ℓ along the lines of [NW94], [Imp95], and [IW97].

We first generalize the theorem of [NW94] used to give conditions for equating BPP and P:

Theorem 3.10 *Let* A *be a class of oracles and* B *an oracle. If there is a Boolean function* $g \in \mathrm{E}^A$ *with* $H_g^B(n) \in 2^{\Omega(n)}$*, then there is a pseudo-random generator* G *computable in* E^A *which is secure against* B *and has seed length* $O(\log n)$*.*

Impagliazzo and Wigderson's construction [IW97], building upon work by Impagliazzo [Imp95], also carries through for oracle circuits:

Theorem 3.11 *Let* A *be a class of oracles and* B *an oracle. For any Boolean function* $f \in \mathrm{E}^A$ *such that* $C_f^B(n) \in 2^{\Omega(n)}$*, there is a Boolean function* $g \in \mathrm{E}^A$ *such that* $H_g^B(n) \in 2^{\Omega(n)}$*.*

Now, by combining Theorems 3.10 and 3.11, we can state a general derandomization theorem:

Theorem 3.12 *Let* A *be a class of oracles and* B *an oracle. If there is Boolean function* $f \in \mathrm{E}^A$ *such that* $C_f^B(n) \in 2^{\Omega(n)}$*, then there is a pseudo-random generator* G *computable in* E^A *which is secure against* B *and has seed length* $O(\log n)$*.*

When we apply Theorem 3.6 for $A = \mathrm{NP} \cap \mathrm{coNP}$ and $B = \mathrm{SAT}$, Lemma 3.1 achieves complete derandomizations of Arthur-Merlin games, noting that $\mathrm{E}^{\mathrm{NP}\cap\mathrm{coNP}} = \mathrm{NE} \cap \mathrm{coNE}$.

Theorem 3.13 *If there is a Boolean function* $f \in \mathrm{NE} \cap \mathrm{coNE}$ *such that* $C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$*, then* $\mathrm{AM} = \mathrm{NP}$*. In particular, the same hypothesis implies that graph nonisomorphism has polynomial size proofs.*

We point out that, under the hypothesis of Theorem 3.13, we give an explicit certificate for graph nonisomorphism in the proof of Lemma 3.1 in the Appendix.

Finally, we note that for derandomizing AM it is actually sufficient to construct efficient pseudo-random generators that are secure against SAT-oracle circuits with *parallel* access to the oracle. All theorems in this section also hold for oracle circuits with such restricted access.

4 A General Framework for Derandomization

In the previous section, we showed that an Arthur-Merlin protocol can be viewed as a SAT-oracle distinguisher for a pseudo-random generator, and if a Boolean function f exists with sufficient hardness against SAT-oracle circuits we can construct a pseudo-random generator based on f that will look random to our Arthur-Merlin protocol. Still, we have only applied our results to randomized *decision* algorithms. In this section, we show how to relax this condition and obtain hardness versus randomness trade-offs for a broader class of randomized processes. Under a sufficient hardness condition depending upon the particular randomized algorithm, we are able to reduce the algorithm's randomness to a logarithmic factor and, in some cases, provide a complete derandomization. Weaker hardness conditions yield partial derandomizations.

We first describe the notion of a randomized process to which our approach applies. We formalize a process that uses $r(n)$ random bits on inputs of length n as a pair (F, π), where

- F is a function that takes a string x and a string ρ of length $r(|x|)$, and outputs the outcome of the process on input x using ρ as the random bit sequence.
- π is a predicate with the same domain as F that indicates whether the process succeeds on input x using ρ, i.e., whether ρ is a "good" choice of random bits for the given input x. We call π the success predicate of the randomized process.

In the case of Arthur-Merlin games F is Boolean. More specifically, for the game defined by (2) and (3) in Section 2, $F(x, \rho)$ coincides with the predicate $\exists z \in \{0,1\}^{t(n)} : M(\langle x, \rho, z\rangle) = 1$; the success predicate $\pi(x, \rho)$ equals $F(x, \rho)$ if $x \in L$ and the complement of $F(x, \rho)$ otherwise. In the specific randomized processes we will consider in Section 5, F will be non-Boolean.

What matters for our derandomization results is the complexity of the success predicate π and more precisely the following property.

Definition 4.1 *Let (F, π) be a randomized process using $r(n)$ random bits, and B an oracle. We say that B can* efficiently check *(F, π) if there is a polynomial p such that for any fixed input x of length n, the predicate $\pi_x : \{0,1\}^{r(n)} \to \{0,1\}$ where $\pi_x(\rho) = \pi(x, \rho)$ can be decided by B-oracle circuits of size $p(n + r(n))$.*

Using the success predicate as a distinguisher in the proofs of Theorems 3.4 and 3.12, we obtain our general derandomization tool:

Theorem 4.2 *Let A be a class of oracles and B an oracle. Let (F, π) be a randomized process using a polynomial number of random bits, and suppose that B can efficiently check (F, π). Then the hardness conditions of the left-hand side of Table 2 provide a pseudo-random generator G with complexity and seed length s as specified on the right-hand side of the table such that for some constant $d > 0$ and any input x of length n*

$$|\Pr_{\rho}[\pi(x, \rho) = 1] - \Pr_{\sigma}[\pi(x, G_{n^d}(\sigma)) = 1]| \in o(1).$$

The parameter s in Table 2 can be any space constructible function.

In order to reduce the randomness of a randomized process, we will first analyze the complexity of an oracle B capable of efficiently checking the associated success predicate and then construct a pseudo-random generator secure against B based on a function with presumed hardness against B.

5 More Applications and New Derandomizations

We will now apply the general framework of Section 4 to various fundamental constructions in computational complexity. As customary, we only state our results in terms of the strongest of the assumptions in Table 2, yielding polynomial time deterministic simulations. It should be noted, however, that weaker assumptions can be taken (the weakest being that the polynomial-time hierarchy does not collapse) in order to achieve weaker, but still subexponential, deterministic simulations.

5.1 Valiant-Vazirani

Our first example is the randomized Boolean hashing protocol developed by Valiant and Vazirani [VV86]. They give a method for pruning the satisfying assignments of a Boolean formula to one:

Theorem 5.1 ([VV86]) *There exists a randomized polynomial time algorithm that, on input a Boolean formula ϕ, outputs a list of Boolean formulas such that:*

- *Every satisfying assignment to any of the formulas in the list also satisfies ϕ.*
- *If ϕ is satisfiable, then with high probability at least one of the formulas in the list has exactly one satisfying assignment.*

Let $F(x, \rho)$ denote the list of formulas the Valiant-Vazirani algorithm produces on input x using coin flips specified by ρ. We define the success predicate $\pi(x, \rho)$ to hold unless x is a satisfiable formula and none of the formulas in $F(x, \rho)$ has a unique satisfying assignment. It is clear that (F, π) corresponds to a formalization of the Valiant-Vazirani process and fits within our framework.

Theorem 5.2 *If there is a Boolean function $f \in \mathrm{E}$ such that $C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$, then, given a Boolean formula ϕ, we can generate in polynomial time a list of Boolean formulas such that:*

- *Every satisfying assignment to any of the formulas in the list also satisfies ϕ.*
- *If ϕ is satisfiable, then at least one of the formulas in the list has exactly one satisfying assignment.*

Proof Sketch
Let (F, π) be the formalization as described above. Notice that checking whether a given Boolean formula has at least two satisfying assignments is an NP question. Hence, we can check, in polynomial time, whether a Boolean formula has a unique satisfying assignment using two queries to SAT. It follows that SAT can efficiently check (F, π). Applying theorem 4.2 to (F, π) yields a pseudo-random generator that looks random to the Valiant-Vazirani process. Enumerating over all seeds and collecting all formulas produces the desired list of formulas. $\square$

We now have the following corollary about computing satisfying assignments nonadaptively.

Corollary 5.3 *If there is a Boolean function $f \in \mathrm{E}$ such that $C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$, then, given a satisfiable Boolean formula ϕ, we can find a satisfying assignment for ϕ in polynomial time given non-adaptive oracle access to* SAT.

We also obtain interesting structural observations. Recall that SPP denotes the class of all languages whose characteristic function is a GapP function [FFK94], i.e., the difference of two #P functions. SPP contains both $\oplus$P and PP. We refer the reader to the survey by Fortnow [For97] for background on these counting classes. Theorem 5.2 implies:

Corollary 5.4 *If there is a Boolean function $f \in \mathrm{E}$ such that $C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$, then* NP *is contained in* SPP*.*

Similarly, we can conditionally derandomize the result by Toda and Ogiwara [TO92] that the polynomial-time hierarchy does not add power to GapP in a randomized setting. Applying our techniques to their main lemma yields:

Lemma 5.5 *Let B be any oracle. If there is a Boolean function $f \in \mathrm{E}$ such that $C_f^{\mathrm{SAT}^B}(n) \in 2^{\Omega(n)}$, then $\mathrm{GapP}^{\mathrm{NP}^B}$ is contained in GapP^B.*

This allows us to show:

Theorem 5.6 *For any integer $k \geqslant 1$ the following holds. If there is a Boolean function $f \in \mathrm{E}$ such that $C_f^{\mathrm{TQBF}_k}(n) \in 2^{\Omega(n)}$, then Σ_k^{p} is contained in* SPP*.*

Here TQBF_k denotes the language of all true fully quantified Boolean formulae with at most $k-1$ quantifier alternations.

Proof Sketch
Corollary 5.4 relativizes and, under the given assumption, implies that the characteristic function of any language in Σ_k^{p} is contained in $\mathrm{GapP}^{\mathrm{TQBF}_{k-1}}$. The successive application of Lemma 5.5 with $B = \mathrm{TQBF}_{k-2}$, $B = \mathrm{TQBF}_{k-3}$, $\ldots$, $B = \mathrm{TQBF}_1$, and $B = \varnothing$ yields under the given assumption that $\mathrm{GapP}^{\mathrm{TQBF}_k}$ is contained in GapP. $\square$

Consequently, we have:

Corollary 5.7 *Let B be any oracle hard for the polynomial-time hierarchy under polynomial-time Turing reductions. If there is a Boolean function $f \in \mathrm{E}$ such that $C_f^{B}(n) \in 2^{\Omega(n)}$, then the polynomial-time hierarchy is contained in* SPP*.*

Corollary 5.7 is strongly related to Toda's Theorem [Tod91] that the polynomial-time hierarchy lies in $\mathrm{BP} \cdot \oplus\mathrm{P}$. Applying our derandomization technique directly to Toda's Theorem yields:

Theorem 5.8 *If there is a Boolean function $f \in \mathrm{E}$ such that $C_f^{\oplus\mathrm{SAT}}(n) \in 2^{\Omega(n)}$, then the polynomial-time hierarchy is contained in $\oplus\mathrm{P}$.*

5.2 Learning Circuits

Learning theory represents another area where we can apply our techniques. We will focus on exact concept learning with equivalence queries [Ang88, Lit88], in which the learner presents hypotheses to a teacher, who then tells the learner whether the hypothesis agrees with the concept in question. If it does, the learner has succeeded; otherwise the teacher provides a counterexample and the learner continues.

A major open problem is whether we can efficiently learn Boolean circuits in this model. Bshouty et al. [BCG+96] showed that we can, provided we have access to an NP oracle and to a source of randomness. We give evidence that we may be able to dispense with the latter one.

Theorem 5.9 *If there is a Boolean function $f \in \mathrm{NE} \cap \mathrm{coNE}$ such that $C_f^{\mathrm{TQBF}_2}(n) \in 2^{\Omega(n)}$, then we can perform the following task in deterministic polynomial time given access to an* NP *oracle: exactly learn Boolean circuits of size t using equivalence queries with hypotheses that are circuits of size $O(tn + n \log n)$.*

As before, TQBF_2 denotes the language of all true fully quantified Boolean formulae with at most one quantifier alternation.

Proof Sketch
We apply Theorem 4.2 to a randomized process (F, π) underlying the algorithm of Bshouty et al. [BCG+96]. We define F in the obvious way. The specification of the success predicate π is somewhat more involved than in the examples we have seen so far. We only want to consider a random seed as "good" if the learner, when using this seed, finds an equivalent circuit no matter how the teacher picks the counterexamples. Therefore, we define $\pi(x, \rho)$ to indicate whether for every choice of candidate counterexamples, either one of them fails to be a valid counterexample on input x and random seed ρ, or else the learner ends up with an equivalent circuit to x. It follows from the construction of Bshouty et al. that π is a Π_2^{p} predicate, and their analysis shows that for any input x, $\pi(x, \rho)$ holds with high probability.

Using the derandomization provided by Theorem 4.2, we end up with a polynomial number of candidate circuits at least one of which is equivalent to x. So, we just present each of these to the teacher and will succeed. □

Bshouty et al. used their learning theory result to improve the known collapse of the polynomial-time hierarchy in case NP would have polynomial size circuits: They showed that the latter implies that the polynomial-time hierarchy is contained in $\mathrm{ZPP}^{\mathrm{NP}}$. Along the same lines, we obtain:

Corollary 5.10 *If* NP *has polynomial size circuits and there is a Boolean function $f \in \mathrm{NE} \cap \mathrm{coNE}$ such that $C_f(n) \in 2^{\Omega(n)}$, then the polynomial-time hierarchy is contained in* P^{NP}.

5.3 Rigid Matrices

Several researchers have studied the problem of finding explicit constructions of combinatorial objects that have been proven to exist non-constructively (by using the probabilistic method, for example). In many cases, an explicit construction of some combinatorial object yields an interesting complexity theoretic result. One of the notable examples of this is the problem of matrix rigidity. The rigidity of a matrix M over a ring S, denoted $R_M^S(r)$, is the minimum number of entries of M that must be changed to reduce its rank to r or below (an entry can be changed to any element of S). Valiant [Val77] proved that an explicit construction of an infinite family of highly rigid matrices yields a circuit lower bound:

Theorem 5.11 ([Val77]) *Let $\epsilon, \delta > 0$ be constants. For any positive integer n, let M_n be an $n \times n$ matrix over a ring S_n. If $R_{M_n}^{S_n}(\epsilon n) \geqslant n^{1+\delta}$ for infinitely many values of n, then the linear transformations defined by the family M_n cannot be computed by linear size, log-depth circuits consisting of gates computing binary linear operators on S_n.*

Valiant also proved that almost all matrices over an infinite field have rigidity $(n-r)^2$ and almost all matrices over a fixed finite field have rigidity $\Omega((n-r)^2/\log n)$. The best known explicit constructions achieve rigidity $\Omega(n^2/r)$ over infinite fields [PV91, Raz] and $\Omega(\frac{n^2}{r}\log(\frac{n}{r}))$ [Fri90] over finite fields. These are not sufficient to obtain circuit lower bounds using Theorem 5.11. Under a hardness assumption, our derandomization technique will give an explicit construction to which Theorem 5.11 applies: a family of matrices M_n over $S_n = \mathbb{Z}_{p(n)}[x]$ such that $R_{M_n}^{S_n}(r) \in \Omega((n-r)^2/\log n)$, where $p(n)$ is polynomially bounded.

We will need to use the following lemma which follows directly from [Val77]:

Lemma 5.12 ([Val77]) *Let S be a ring with at least 2 elements, and n a positive integer. All but at most a $\frac{1}{n}$ fraction of the $n \times n$ matrices M over S satisfy*

$$R_M^S(r) \geqslant \frac{(n-r)^2 - 2n - 2\log n}{1 + 2\log n}. \tag{5}$$

We can use our technique to achieve the nonconstructive rigidity bounds by noticing that there exist polynomial-sized SAT-oracle circuits which can check if a matrix is rigid.

Theorem 5.13 *If there exists a Boolean function $f \in \mathrm{E}$ such that $C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$, then given integers $n \geqslant 1$ and $p \geqslant 2$, we can construct in time polynomial in $n + \log p$ a list of $n \times n$ matrices M over $S = \mathbb{Z}_p$ most of which satisfy (5).*

Proof Sketch
Let ρ be a string of length $n^2 \cdot \lceil \log p \rceil$. We will view ρ as the concatenation of n^2 blocks of $\lceil \log p \rceil$ bits each. Let $F(\langle 1^n, p \rangle, \rho)$ denote the matrix whose ij-th entry is the $(n(i-1)+j)$-th block of ρ interpreted as a number in binary and taken modulo p. We define the predicate $\pi(\langle 1^n, p \rangle, \rho)$ to be true if $M = F(\langle 1^n, p \rangle, \rho)$ satisfies (5). Note that the latter is a coNP predicate: If (5) is violated for some r, we can guess modified values for fewer than the right-hand side of (5) many entries of M and verify that the rank of the modified matrix over S is at most r. So, we can decide π by one query to an oracle for SAT. Moreover, Lemma 5.12 states that for most sequences ρ, the predicate holds. By applying theorem 4.2 to (F, π) we obtain a pseudo-random generator which on most seeds outputs a matrix with the required rigidity property. Enumerating over all seeds gives us the desired list of matrices. □

Now we need to combine this list of matrices into a single matrix with similarly high rigidity. We refer to Appendix B for a proof of the following construction.

Lemma 5.14 *Given $n \times n$ matrices $M_0, M_1, \ldots, M_k$ over $\mathbb{Z}_p$ where p is a prime larger than k, we can construct in time polynomial in $n + k + \log p$ an $n \times n$ matrix N over $\mathbb{Z}_p[x]$ such that*

$$R_N^{\mathbb{Z}_p[x]}(r) \geqslant \max_{0 \leqslant i \leqslant k} R_{M_i}^{\mathbb{Z}_p}(r), \tag{6}$$

where the entries of N are polynomials of degree at most k.

Given this construction of rigid matrices we can conclude the following new relationship among circuit lower bounds:

Theorem 5.15 *If there exists a function $f \in \mathrm{E}$ such that $C_f^{\mathrm{SAT}}(n) \in 2^{\Omega(n)}$ then there exists a polynomially bounded function $p(n)$ and a polynomial-time computable family of matrices M_n where M_n is an $n \times n$ matrix over $\mathbb{Z}_{p(n)}[x]$ such that the linear transformations defined by the family M_n cannot be computed by log-depth linear size circuits which have special gates that can compute binary linear operators over $\mathbb{Z}_{p(n)}[x]$.*

Proof Sketch
For any polynomial time computable function $p(n)$, Theorem 5.13 allows us to efficiently compute a list of $(n + \log p(n))^c$ matrices M over $\mathbb{Z}_{p(n)}$ for some constant c, most of which are rigid. Provided $p(n)$ is a prime satisfying

$$p(n) \geqslant (n + \log p(n))^c, \tag{7}$$

Lemma 5.14 efficiently combines them into a single matrix N over $\mathbb{Z}_{p(n)}[x]$ which satisfies a rigidity condition sufficient for Theorem 5.11. The smallest prime value for $p(n)$ that satisfies (7) is polynomially bounded in n, and we can compute it in time polynomial in n. □

5.4 Universal Traversal Sequences

Universal traversal sequences, introduced by Cook, form another example where explicit constructions have important complexity theoretic implications. A universal traversal sequence for size n is a sequence σ of labels from $\{1, 2, \ldots, n-1\}$ such that for any undirected connected graph G with n vertices in which the incident edges at every vertex have been assigned distinct labels from $\{1, 2, \ldots, n-1\}$, the following process always visits every vertex of G: Pick an arbitrary start vertex, and in subsequent steps, go along the edge with the label matching the next symbol of σ; in case of no match, stay put during that step and continue with the next symbol of σ.

If we can construct universal traversal sequences in logspace, then we can solve undirected graph connectivity in logspace, and symmetric logspace equals logspace [LP82]. However, we do not know how to generate universal traversal sequences in logspace or even in polynomial time. Aleliunas et al. [AKL+79] showed that most sequences of length $O(n^3)$ over $\{1, 2, \ldots, n-1\}$ are universal traversal sequences for size n, but as of now the best explicit construction, due to Nisan [Nis92], yields universal traversal sequences of length $n^{O(\log n)}$. We give evidence supporting the belief that explicit universal traversal sequences of polynomial size can be generated efficiently.

A straightforward application of our technique would yield a polynomial-time construction under the assumption that E requires exponential size SAT-oracle circuits. Since being a universal traversal sequence is a coNP predicate, Theorem 4.2 applied to the Aleliunas et al. process and oracle $B = \mathrm{SAT}$ would efficiently generate under that hypothesis a collection of sequences most of which are universal traversal sequences. Concatenating all of them would yield the desired universal traversal sequence of polynomial size. However, we can do better and dispense with the oracle B.

Theorem 5.16 *If there is a Boolean function $f \in E$ such that $C_f(n) \in 2^{\Omega(n)}$, then we can construct universal traversal sequences in polynomial time.*

Proof Sketch
Let G encode a graph with n vertices with edge labels as above. For any sequence σ over $\{1, 2, \ldots, n-1\}$, let $\tau(G, \sigma)$ indicate whether for every vertex v of G, the walk in G starting from v and dictated by σ visits every vertex of G. Let $F(G, \rho)$ denote the sequence of length $c \cdot n^3$ over $\{1, 2, \ldots, n-1\}$ (where c is some sufficiently large constant) as specified by the successive bits of ρ. Let $\pi(G, \rho)$ equal $\tau(G, F(G, \rho))$. Note that π is a P predicate, so $B = \varnothing$ can efficiently check the randomized process (F, π). Since every universal traversal sequence σ satisfies $\tau(G, \sigma)$, the result of Aleliunas et al. [AKL+79] shows that for any graph G, $\pi(G, \rho)$ holds for most ρ. Therefore, Theorem 4.2 allows us to generate in polynomial time a collection of sequences σ most of which satisfy $\tau(G, \sigma)$. Their concatenation forms a single sequence σ' satisfying $\tau(G, \sigma')$. Since the σ's are independent of G, so is their concatenation σ'. Hence, we have constructed a sequence σ' which satisfies $\tau(G, \sigma')$ for every edge labeled graph G with n vertices, i.e., we have found a universal traversal sequence σ'. □

Under the stronger assumption that linear space requires exponential size circuits, we can actually construct universal traversal sequences in logspace. In fact, that assumption allows us to build logspace computable pseudo-random generators for logspace, and hence to derandomize BPL, the class of languages accepted by logspace randomized Turing machines with bounded two-sided error.

Theorem 5.17 *If there is a Boolean function $f \in \mathrm{DSPACE}[n]$ such that $C_f(n) \in 2^{\Omega(n)}$, then* $\mathrm{BPL} = \mathrm{L}$.

We refer to Appendix C for a proof. Along the lines of Babai et al. [BNS92], the pseudo-random generators behind Theorem 5.17 let us conclude:

Corollary 5.18 *If there is a Boolean function $f \in \mathrm{DSPACE}[n]$ such that $C_f(n) \in 2^{\Omega(n)}$, then we can construct universal traversal sequences in logspace.*

6 Conclusions

In this paper, we have demonstrated the power of relativization in the area of derandomization. We gave several striking examples, most notably Arthur-Merlin games, and are convinced that more will follow. In fact, Peter Bro Miltersen [Mil98] recently informed us that one can use these techniques to obtain an alternate perspective on the recent breakthrough extractor constructions of Trevisan [Tre98] and Vadhan [Vad98].

Acknowledgments

We want to express our sincere gratitude to Oded Goldreich, Madhu Sudan, Luca Trevisan, and Salil Vadhan for their heroic encouragement and collaboration. We would also like to thank Rudi Seitz for his contribution to Section 5.4; Theorem 5.16 was proved jointly with him.

The first author wishes to thank Dan Spielman and Salil Vadhan for their selfless, patient support on all aspects of this paper. The first author also wishes to thank Henry Cohn, Steven Rudich, and Avi Wigderson for illuminating discussions on this topic.

The second author wishes to thank his adviser Lance Fortnow for the guidance, suggestions, and support he received, and for the statement of Corollary 5.10 in particular. He also wants to thank the following people for stimulating and helpful discussions about issues related to this paper: Eric Allender, Behfar Bastani, Harry Buhrman, Valentine Kabanets, Mike Saks, Avi Wigderson, and Shiyu Zhou. The second author is grateful to Eric Allender, Peter Bro Miltersen, Valentine Kabanets, and Salil Vadhan for the comments he received on an earlier version of the paper.

References

[AK97] V. Arvind and J. Köbler. On pseudorandomness and resource-bounded measure. In *Proceedings 17th Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 1346 of *Lecture Notes in Computer Science*, pages 235–249. Springer-Verlag, 1997.

[AKL+79] R. Aleliunas, R. Karp, R. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 218–223. IEEE, 1979.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[AR98] E. Allender and K. Reinhardt. Isolation, matching, and counting. In *Proceedings of the 13th IEEE Conference on Computational Complexity*, pages 92–100. IEEE, 1998.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, 1985.

[BCG+96] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

[BDG90] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*, volume 22 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1990.

[BDG95] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1995.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BNS92] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45:204–232, 1992.

[CG89] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5:96–106, 1989.

[Coo88] S. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26:269–270, 1988.

[CRT98] A. Clementi, J. Rolim, and L. Trevisan. Recent advances towards proving P = BPP. *Bulletin of the European Association for Theoretical Computer Science*, 64:96–103, February 1998.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[For97] L. Fortnow. Counting complexity. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 81–107. Springer-Verlag, 1997.

[Fri90] J. Friedman. A note on matrix rigidity. Technical Report TR-308-91, Department of Computer Science, Princeton University, 1990.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38:691–729, 1991.

[GS89] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, Greenwich, 1989.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 538–545. IEEE, 1995.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP unless E has sub-exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 220–229. ACM, 1997.

[Lit88] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[LP82] H. Lewis and Ch. Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, 19(2):161–187, 1982.

[Mil98] P. Bro Miltersen. Relativizable pseudorandom generators and extractors. Comment to ECCC Technical Report TR98-055, 1998.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Pap94] Ch. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PV91] P. Pudlák and Z. Vavřín. Computation of rigidity of oerder n^2/r for one simple matrix. *Comment. Math. Univ. Carolinae*, 32:213–218, 1991.

[Raz] A. Razborov. On rigid matrices. Manuscript in Russian.

[Rud97] S. Rudich. Super-bits, demi-bits, and $\tilde{\mathrm{NP}}$/qpoly-natural proofs. In *Proceedings of the 1st International Symposium on Randomization and Approximation Techniques in Computer Science*, volume 1269 of *Lecture Notes in Computer Science*, pages 85–93. Springer-Verlag, 1997.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tre98] L. Trevisan. Constructions of near-optimal extractors using pseudo-random generators. Technical Report TR-98-055, Electronic Colloquium on Computational Complexity, 1998.

[Vad98] S. Vadhan. Extracting all the randomness from a weakly random source. Technical Report TR-98-047, Electronic Colloquium on Computational Complexity, 1998.

[Val77] L. Valiant. Graph-theoretic arguments in low-level complexity. In *Proceedings of the 6th Symposium on Mathematical Foundations of Computer Science*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer-Verlag, 1977.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91. IEEE, 1982.

A Proof of Lemma 3.1

Let L be a language satisfying (2) and (3), and consider

$$L' = \{\langle x, y\rangle \mid y \in \{0,1\}^{t(|x|)} \text{ and } \exists\, z \in \{0,1\}^{t(|x|)} : M(\langle x, y, z\rangle) = 1\}.$$

By Cook's Theorem [Coo88], since $L' \in \mathrm{NTIME}[n]$, there exists a circuit of size $t'(n) \in O(t(n)\log^2 t(n))$ that many-one reduces L' to SAT on inputs $\langle x, y\rangle$ with $x \in \{0,1\}^n$ and $y \in \{0,1\}^{t(n)}$. For any fixed value of $x \in \{0,1\}^n$, let C_x denote the corresponding oracle circuit obtained by hardwiring x. Note that C_x makes a single oracle query and outputs the answer to that query. Since G is a pseudo-random generator secure against SAT, conditions (2) and (3) imply that

$$\Pr_{\sigma}[C_x^{\mathrm{SAT}}(G_{t'(n)}(\sigma)) = 1] \begin{array}{ll} > \frac{1}{2} & \text{if } x \in L \\ < \frac{1}{2} & \text{if } x \notin L, \end{array} \tag{8}$$

where the probability is with respect to the uniform distribution of σ over $\{0,1\}^{s(t'(n))}$.

Since G is computable in $\mathrm{NTIME}[\tau(n)] \cap \mathrm{coNTIME}[\tau(n)]$, there exists a nondeterministic Turing machine N running in time τ that accepts $\{\langle n, \sigma, i, b\rangle \mid i\text{-th bit of } G_n(\sigma) \text{ equals } b\}$. Now, consider the nondeterministic algorithm in Figure 1.

```
counter ← 0
for every σ ∈ {0,1}^{s(t'(n))}
    for  i ← 1, ..., t(n)
        guess ρ_i ∈ {0,1}
        guess a computation path p for N(⟨n, σ, i, ρ_i⟩)
        if N(⟨n, σ, i, ρ_i⟩) rejects along p
          then reject and abort
    guess z ∈ {0,1}^{t(n)}
    counter ← counter + M(⟨x, ρ, z⟩)
if counter > 2^{t'(n)−1} then accept
```

Figure 1: Nondeterministic algorithm for deciding L

Figure 1 describes a nondeterministic Turing machine that runs in time $2^{s(t'(n))} \cdot \tau(s(t'(n))) \cdot t(n)$. The largest possible value of counter at the end of the outer loop over all possible nondeterministic choices in the algorithm of Figure 1, equals $2^{s(t'(n))} \cdot \Pr_\sigma[C_x^{\mathrm{SAT}}(G_{t'(n)}(\sigma)) = 1]$. It follows from (8) that the machine accepts L. This finishes the proof of Lemma 3.1.

B Proof of Lemma 5.14

Let $q_{ij}(x)$ be the polynomial of degree at most k such that $q_{ij}(\ell)$ equals the ij-th entry of M_ℓ, $0 \leqslant \ell \leqslant k$, i.e., q_{ij} interpolates the ij-th entries of all $k+1$ matrices. Note that the q_{ij}'s exist and that each coefficient can be computed in time polynomial in $k + \log p$. Let N be the matrix whose ij-th entry is q_{ij}. We now argue that (6) holds.

Let $m = R_N^{\mathbb{Z}_p[x]}(r)$, and let N' be a matrix obtained by changing m entries of N such that the rank of N' is at most r. Then every $(r+t) \times (r+t)$ minor of N' has determinant 0 for $0 < t \leqslant n - r$.

Thus the determinant of any $(r+t)\times(r+t)$ minor of N' can be viewed as an identically 0 polynomial in its entries. Let $\phi_a(N')$ be the matrix obtained by substituting $a \in \mathbb{Z}_p$ for x in every entry of N' (recall N' has entries from $\mathbb{Z}_p[x]$). Every $(r+t)\times(r+t)$ minor of $\phi_a(N')$ has determinant 0 for $0 < t \leqslant n-r$. We conclude by noticing that changing m or fewer entries of $\phi_a(N)$ has reduced its rank to a value less than or equal to r. But $\phi_a(N)$ equals M_a for $a \in \{0,1,\ldots,k\}$. Therefore, $R_{M_i}^{\mathbb{Z}_p}(r) \leqslant m$ for any $0 \leqslant i \leqslant k$, i.e., (6) holds. □

C Proof of Theorem 5.17

The proof of Theorem 5.17 follows the outline of the proof of Theorem 3.12 — it is a combination of the extension of Theorems 3.10 and 3.11 to the space bounded setting. The key observations are the following:

- There is a way to make Nisan and Wigderson's pseudo-random generator used in the proof of Theorem 3.10 run in logspace, provided the function g is computable in linear space.
- The polynomial extensions of a Boolean function computable in linear space over fields constructible in linear space, are computable in linear space.
- The pseudo-random generators of Impagliazzo [Imp95] and Impagliazzo and Wigderson [IW97] used to derandomize the XOR Lemma in the proof of Theorem 3.11 are computable in logspace.

The last two bullets are straightforward to check. The crucial ingredient for the first bullet is the space efficient construction of certain designs.

An (ℓ, m) design of size k over a universe Ω is a collection $S_1, S_2, \ldots, S_k$ of subsets of Ω, each of size m, such that for any $1 \leqslant i < j \leqslant k$, the intersection $S_i \cap S_j$ has size at most ℓ.

Nisan and Wigderson [NW94] showed that the greedy approach works for constructing designs with $m = \alpha n$, $\ell = \beta n$, and $k = 2^{\gamma n}$ where $n = |\Omega|$, for α an arbitrary constant in $(0,1)$, and β and γ sufficiently small positive constants depending on α. Their construction runs in time $2^{O(n)}$. We show how to do it in space $O(n)$ using a different approach, namely along the lines of [IW97].

Lemma C.1 *For any constant $\alpha \in (0,1)$, there are constants $\beta, \gamma \in (0,1)$ such that we can generate a $(\beta n, \alpha n)$ design of size $2^{\gamma n}$ over $\{1,2,\ldots,n\}$ in space $O(n)$.*

Eric Allender informed us that Avi Wigderson showed him the same construction. A similar proof to the one we give next appeared in [AR98].

Proof of Lemma C.1
We will show that the following process has a positive probability of generating an (ℓ, m) design of size $k = 2^{\gamma n}$ over $\Omega = \{1,2,\ldots,n\}$ for sufficiently small positive constants β and γ, where $\ell = \beta n$ and $m = \alpha n$: Pick k subsets of Ω of size m in a pairwise independent way such that each of the $\binom{n}{m}$ subsets has about the same probability of being selected.

More precisely, we will do the following. We choose an integer $a \geqslant 0$ such that $2^a \leqslant k^3 \cdot \binom{n}{m} < 2^{a+1}$, pick k numbers $s_i \in \{0,1,\ldots,2^a-1\}$ at random in a pairwise independent way, and set S_i to be the $[(s_i \bmod \binom{n}{m}) + 1]$-st subset of Ω of size m (say using lexicographical order). Known constructions of such sample spaces [CG89] only need $O(a)$ random bits and can generate the samples from the random bits in space $O(a)$. Moreover, checking whether a given sample $S_1, \ldots, S_k$ forms an (ℓ, m) design can be done within the same space bounds. Therefore, we can cycle through all possible random bit sequences, check the corresponding candidate design and output the first valid one. This process runs in space $O(a) = O(n)$, and succeeds provided the probability of picking a valid design this way is positive. The remainder of the proof will argue the latter.

Claim C.2 *Let S, S' be subsets of Ω of size m chosen independently and uniformly at random. Then for some constant c_α,*

$$\Pr\left[\,|S \cap S'| \geqslant 2\alpha^2 n\,\right] \leqslant c_\alpha \cdot n \cdot \exp\left(-\frac{\alpha^4 n}{2}\right).$$

Proof of Claim C.2
First consider a subset S of Ω obtained by doing the following independently for every $i \in \Omega$: Put i in S with probability α. Similarly, and independently, construct S'. Then, by Chernoff's bounds,

$$\Pr\left[\,|S \cap S'| \geqslant 2\alpha^2 n\,\right] \leqslant \exp\left(-\frac{\alpha^4 n}{2}\right).$$

Stirling's formula yields that

$$\Pr[\,|S| = \alpha n\,] \sim \frac{1}{\sqrt{2\pi\alpha(1-\alpha)}} \cdot \frac{1}{\sqrt{n}}.$$

So,

$$\Pr\left[\,|S \cap S'| \geqslant 2\alpha^2 n \;\middle|\; |S| = |S'| = \alpha n\right] \leqslant c_\alpha \cdot n \cdot \exp\left(-\frac{\alpha^4 n}{2}\right)$$

for some constant c_α depending on α. Since the above distribution of S and S' conditioned on $|S| = |S'| = \alpha n$ coincides with the uniform distribution of the statement of the claim, this finishes the proof of the claim. (Claim C.2) □

There is a constant $c < 2k^3$ such that for any $1 \leqslant i \leqslant k$ and any $T \subseteq \Omega$ with $|T| = m$,

$$\frac{c}{2^a} \leqslant \Pr[S_i = T] \leqslant \frac{c+1}{2^a}.$$

Because of the pairwise independence, it follows that for any $1 \leqslant i < j \leqslant k$, the distribution of (S_i, S_j) differs from uniform by at most

$$\begin{aligned}
\binom{n}{m}^2 \cdot \left[(\frac{c+1}{2^a})^2 - (\frac{c}{2^a})^2\right] &= (2c+1)\left(\frac{\binom{n}{m}}{2^a}\right)^2 \\
&< (2c+1)(\frac{2}{k^3})^2 \\
&< \frac{4(4k^3+1)}{k^6} \\
&< \frac{17}{k^3}
\end{aligned}$$

in L_1-norm. Therefore, by Claim C.2,

$$\Pr\left[\,|S_i \cap S_j| \geqslant 2\alpha^2 n\,\right] \leqslant \frac{17}{k^3} + c_\alpha \cdot n \cdot \exp\left(-\frac{\alpha^4 n}{2}\right).$$

Hence, with $\ell = \beta n$ and $\beta = 2\alpha^2$,

$$\Pr[S_1, S_2, \ldots, S_k \text{ is not an } (\ell, m) \text{ design}] \leqslant \binom{k}{2} \cdot \left(\frac{17}{k^3} + c_\alpha \cdot n \cdot \exp\left(-\frac{\alpha^4 n}{2} \right) \right). \tag{9}$$

Since the right-hand side of (9) approaches 0 for $k = 2^{\gamma n}$ with $0 < \gamma < \frac{\alpha^4 \ln e}{4}$, this finishes the proof of the lemma. (Lemma C.1) $\square$