On Derandomizing Algorithms that Err Extremely Rarely

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

Avi Wigderson
School of Mathematics
Institute for Advanced Study
Princeton, NJ 08540, USA.
avi@ias.edu

September 19, 2016

Abstract

Does derandomization of probabilistic algorithms become easier when the number of "bad" random inputs is extremely small?

In relation to the above question, we put forward the following *quantified derandomization challenge*: For a class of circuits $\mathcal{C}$ (e.g., $\mathcal{P}$/poly or $\mathcal{AC}^0, \mathcal{AC}^0[2]$) and a bounding function $B : \mathbb{N} \to \mathbb{N}$ (e.g., $B(n) = n^{\log n}$ or $B(n) = \exp(n^{0.99})$)), given an n-input circuit C from $\mathcal{C}$ that evaluates to 1 on all but at most $B(n)$ of its inputs, find (in deterministic polynomial-time) an input x such that $C(x) = 1$. Indeed, the *standard* derandomization challenge for the class $\mathcal{C}$ corresponds to the case of $B(n) = 2^n/2$ (or to $B(n) = 2^n/3$ for the two-sided version case). Our main results regarding the new *quantified* challenge are:

1. Solving the *quantified* derandomization challenge for the class $\mathcal{AC}^0$ and every sub-exponential bounding function (e.g., $B(n) = \exp(n^{0.999})$).
2. Showing that solving the *quantified* derandomization challenge for the class $\mathcal{AC}^0[2]$ and any sub-exponential bounding function (e.g., $B(n) = \exp(n^{0.001})$), implies solving the *standard* derandomization challenge for the class $\mathcal{AC}^0[2]$ (i.e., for $B(n) = 2^n/2$).

Analogous results are obtained also for stronger (Black-box) forms of efficient derandomization like hitting-set generators.

We also obtain results for other classes of computational devices including log-space algorithms and Arithmetic circuits. For the latter we present a deterministic polynomial-time hitting set generator for the class of n-variate polynomials of degree d over GF(2) that evaluate to 0 on at most an $O(2^{-d})$ fraction of their inputs.

In general, the quantified derandomization problem raises a variety of seemingly unexplored questions about many randomized complexity classes, and may offer a tractable approach to unconditional derandomization for some of them.

Keywords: Derandomization, approximate counting, pseudorandom generators, Hastad's switching lemma, $\mathcal{AC}^0$, $\mathcal{AC}^0[2]$, log-space, $\mathcal{MA}$ and $\mathcal{AM}$.

Contents

1 Introduction

The challenge of derandomizing various complexity classes and algorithms has fascinated the theory of computation community ever since Yao's [39] (conditional) subexponential-time derandomization of $\mathcal{BPP}$. One branch of research refers to strong computational models and employs complexity assumptions (cf., e.g., [29]), whereas the other branch focuses on unconditional results for relatively weak models of computation (as in the celebrated derandomizations of randomized logarithmic-space [27, 28] and approximate counting for $\mathcal{AC}^0$ [26]). The current work is positioned within the latter branch.

Specifically, the known deterministic algorithms for approximate counting for $\mathcal{AC}^0$ run in quasi-polynomial time. While significant progress has been made recently regarding the derandomization of approximate counting for $\mathcal{AC}^0$ (cf., e.g., [12, 18, 38]), we still do not know of a (deterministic) polynomial-time algorithm that finds a satisfiable assignment when given a CNF that is satisfied by a majority of its assignments. That is, we do not have a "full derandomization" even when the circuit is of depth two.

In light of the above, we propose a seemingly easier computational problem in which one is asked to find a satisfying assignment for a circuit that is satisfied by a vast majority of its assignments (i.e., by almost all assignments). Specifically, for a class of circuits $\mathcal{C}$ such as $\mathcal{AC}^0, \mathcal{TC}^0, \mathcal{NC}$ or even $\mathcal{P}/\mathrm{poly}$, and a function $B : \mathbb{N} \to \mathbb{N}$ such as $B(n) = 2^{\sqrt{n}}$ or $B(n) = n^{\log_2 n}$, provide a (deterministic) polynomial-time algorithm that *when given an n-input circuit $C \in \mathcal{C}$ that is satisfied by all but at most $B(n)$ of its possible inputs, finds an n-bit input that satisfies C.* Indeed, B can be thought of as a bound on the number of bad (or exceptional) inputs, and the standard question of derandomization refers to the case that these bad inputs are merely in minority (i.e., $B(n) = 2^{n-1}$).[1]

Definition 1.1 (the quantified derandomization problem): *For a class of circuits $\mathcal{C}$ and a function $B : \mathbb{N} \to \mathbb{N}$, the $(\mathcal{C}, B)$-*search problem *is the following promise problem:*
Input: *An n-input circuit $C \in \mathcal{C}$ that evaluates to 1 on all but at most $B(n)$ of its possible inputs;*
Desired output: *An n-bit string on which C evaluates to 1.*

The $(\mathcal{C}, B)$-search problem is easy if B is a fixed polynomial and the deterministic algorithm is allowed running time that exceeds B. However, if we seek a (single) polynomial-time algorithm that may handle any polynomial B (or just a polynomial B that is larger than the running time of the algorithm), then solving the $(\mathcal{P}/\mathrm{poly}, B)$-search problem does not seem so easy (whereas the case of a subexponential B is as hard as the case of $B(n) = 2^{n-1}$; see Theorem 1.4 below).[2] We do not know whether efficient derandomization in this regime (when B is small), implies any circuit lower bounds (as is the case for large B).

As an initial step in the study of the quantified derandomization problem, we focus on several classes of circuits and other computational models, which we detail in the rest of this introduction.

[1] In the introduction we focus on the one-sided error version of the problem, but our results apply also to the two-sided version.

[2] Basically, by using strong error reduction, one may reduce the standard derandomization problem (i.e., with $B(n) = 2^{n-1}$) to one with subexponential B (i.e., with $B(n) = 2^{n^c}$ for any $c > 1$). Sipser was the first to conceive of such a strong error reduction, and named the class $\mathcal{RP}$ with such small $B(n)$ "Strong R" [32]. Such an error reduction has become a reality via the connection to randomness extractors established by Zuckerman [41], and the construction of adequate extractors by Trevisan [37]. Theorem 1.4 asserts that all of this applies to $\mathcal{AC}^0[2]$, and not merely to $\mathcal{P}/\mathrm{poly}$.

Logarithmic space. In order to illustrate the possibilities that emerge in the study the quantified problem (i.e., of derandomization with respect to bounds on the number of bad inputs), we first consider the simple case of (log-space uniform) ordered (read-once) branching program of polynomial width, which correspond to the log-space computations, and quasi-polynomial bounding functions.[3]

Proposition 1.2 (the case of log-space and $B(n) = \exp(\text{poly}\log n)$): *Suppose that S is decidable by a probabilistic log-space algorithm that errs only on at most quasi-polynomial many sequences of the possible random outcomes. Then, S is in $\mathcal{L}$.*

Following the initial posting of this work, we learned that a much stronger result was obtained (but never published) by Mike Saks in the 1990s: See Theorem A.1 (in Appendix A), which obtains a derandomization to the case of $B(n) = 2^{cn}$ for any $c < 1$.

Proof: Let $B(n) = \exp(\text{poly}(\log n)))$ denote an upper bound on the number of erroneous random pads for a generic n-bit input. Then, letting $\ell = \ell(n) = \log_2 B(n)$, we set (say, to zero) all but the first $\ell + 2$ random bits of the algorithm, and obtain a randomized log-space algorithm of polylogarithmic randomness complexity that errs with probability at most $1/4$. Applying the Nisan-Zuckerman pseudorandom generator [30], we are done. ■

Constant-depth circuits ($\mathcal{AC}^0$)**.** Our main positive result resolves the quantified derandomization problem for the case of $\mathcal{AC}^0$ and any sub-exponentially bounded function B (i.e., $B(n) < 2^{n^c}$ for some constant $c < 1$).[4]

Theorem 1.3 (the case of $\mathcal{AC}^0$ and $B(n) = \exp(n^{1-\Omega(1)})$): *Let $\mathcal{AC}^0_{d,p}$ denote the class of depth-d circuits of size at most $p(n)$, where n is the number of inputs to the circuit. For every two constants $c < 1$ and $d \in \mathbb{N}$ and any polynomial p, the $(\mathcal{AC}^0_{d,p}, 2^{n^c})$-search problem can be solved in* (deterministic) *polynomial-time. Moreover, we give a Hitting-Set generator for this class; that is, for every $c < 1$, $d \in \mathbb{N}$ and polynomial p, there exists a* (deterministic) *polynomial-time algorithm that on input 1^n, outputs a set of n-bit strings S_n such that every circuit that satisfies the promise of the $(\mathcal{AC}^0_{d,p}, 2^{n^c})$-search problem evaluates to 1 on some string in S_n. Furthermore, every such circuit evaluates to 1 on at least two-thirds of the strings in S_n.*

The furthermore-clause implies that, when given a constant-depth circuit that evaluates to σ on at least $2^n - 2^{n^c}$ of the possible n-bit assignments, we can decide in (deterministic) polynomial-time whether $\sigma = 1$ or not.

The proof of Theorem 1.3 uses a new switching lemma in which the restrictions are chosen pseudorandomly, using only a logarithmic number of random bits.[5] This switching lemma, presented in Section 3.1, simplifies any depth-two circuit, while leaving a large number of variables undetermined, but it does not necessarily preserve the fraction of satisfying assignments of the original circuit. Hence, this lemma cannot be used for approximate counting in general, but it can

[3] See [31, Sec. 1.1] for a recent review of models of branching program.

[4] Recall that the class $\mathcal{AC}^0$ refers to Boolean circuit over the standard (de-Morgan) basis; that is, each of its gates is either an AND-gate or an OR-gate of unbounded arity, or a NOT-gate.

[5] A weaker result can be obtained by using the deterministic switching lemma of Agrawal *et al.* [1, sec. 4]. This suffices for obtaining the main claim of Theorem 1.3 for *some* (tiny) $c > 0$, which depends on p and d, but not for all $c < 1$. Also, this alternative does not establish the furthermore-clause (of Theorem 1.3), since the switching lemma of Agrawal *et al.* [1] uses the input circuit in an essential way.

be used for our application as long as the number of undetermined variables is greater than the logarithm of the number of assignments that do not satisfy the original circuit.

Constant-depth circuits with parity-gates **($\mathcal{AC}^0[2]$).** We observe that an analogous result for $\mathcal{AC}^0[2]$ (i.e., extending Theorem 1.3 to "$\mathcal{AC}^0$ circuits with parity gates") would imply a polynomial-time hitting set generator for $\mathcal{AC}^0[2]$ itself. In fact a stronger result holds (where "$\forall c<1$" is replaced by "$\exists c>0$"):

Theorem 1.4 (the case of $\mathcal{AC}^0[2]$ and $B(n)=\exp(n^{\Omega(1)})$): *Let $\mathcal{AC}^0_{d,p}[2]$ denote the class of depth-d circuits with parity of size at most $p(n)$. Suppose that for every constant d and polynomial p there exists a constant $c>0$ such that the $(\mathcal{AC}^0_{d,p}[2],2^{n^c})$-search problem can be solved in* (deterministic) *polynomial-time. Then, there exists a* (deterministic) *polynomial-time algorithm that finds a satisfying assignment to any $\mathcal{AC}^0[2]$ circuit that is satisfied by a majority of its inputs.*

Theorem 1.4 generalizes to any class of circuits that can compute a randomness extractor with parameters as those of Trevisan's [37] (and can compute approximate majority as well as branch to polynomially many computations). The argument uses the connection between randomness extraction and error reduction outlined by Zuckerman [41]. For details, see Section 4.

Two frontiers. The two parameters of the quantified derandomization problem (i.e., a class of circuits $\mathcal{C}$ and a bounding function B) suggest two frontiers in which one may push the positive result (of Theorem 1.3) forward. The first frontier aims at larger bounding functions; that is, functions B of the form $\exp(n^{1-o(1)})$. Following the initial posting of this work, we were able to show that extending Theorem 1.3 to $B(n)=\exp(n/\mathrm{poly}(\log n))$ would yield a (deterministic) polynomial-time algorithm for approximate counting $\mathcal{AC}^0$; see Theorem 3.4.

The second frontier aims at classes larger than $\mathcal{AC}^0$. In particular, note that Theorem 1.4 is not applicable to bounding functions B of the form $B(n)=\exp(n^{o(1)})$. Hence, we may try to extend Theorem 1.3 to $\mathcal{AC}^0[2]$ coupled with such functions B. A very minimal step is suggested next:

Open Problem 1.5 ($\mathcal{AC}^0[2]$ and B that is larger than the solver's running time): *For any polynomial p and $d>2$, present a* (deterministic) *p-time algorithm that solves the $(\mathcal{AC}^0_{d,n^2}[2],p^2)$-search problem.*[6]

It is conceivable that the above challenge can be solved without providing a hitting set generator for the class $\mathcal{AC}^0[2]$. The same holds for depth-three $\mathcal{AC}^0[2]$ circuits and bounding functions B of the form $B(n)=\exp(n^{\Omega(1)})$ since the proof of Theorem 1.4 (even when applied to depth-two circuits) yields circuits of depth at least five (see Remark 4.4). In Section 6 we present partial results regarding $\mathcal{AC}^0[2]$, which led us to consider also the arithmetic setting.

The arithmetic setting. Suppose that f is an n-variate polynomial of degree d over GF(2). If f evaluates to 0 on less than a 2^{-d} fraction of its domain, then f must be identically 1 and finding an input on which it evaluates to 1 is trivial. *But what happens beyond this threshold of triviality?* Specifically, for which functions $b:\mathbb{N}\to\mathbb{N}$ can we find deterministically and efficiently an input on which f evaluates to 1 when it is guaranteed that $\Pr_x[f(x)=0]\leq b(n)\cdot 2^{-d}$? We prove that this is possible when b is any constant.

[6] That is, consider n^2-sized circuits (of depth d) with parity that evaluate to 1 on all but at most $p(n)^2$ of their possible n-bit inputs. For starters, consider either the special case in which all parity gates are at the bottom layer (cf. Remark 4.3 as well as Case 2 in Section 6) or the special case of $d=3$ (see Case 4 in Section 6).

Theorem 1.6 (polynomials with $b(n) = O(1)$): *For every constant c, there exists a deterministic* poly(n)*-time algorithm that outputs a set of n-bit strings S_n such that for every d and every n-variate polynomial f of degree d over* GF(2) *that evaluates to 0 on at most a $c \cdot 2^{-d}$ fraction of its domain* (i.e., $\Pr_x[f(x)=0] \leq c \cdot 2^{-d}$)*, there exists $x \in S_n$ such that $f(x) = 1$.*

As stated above, the case of $c < 1$ is trivial, since in this case the polynomial must be identically 1. Theorem 1.6 is proved by using a refinement of Lemma 4 in Viola [36], which refers to "fooling polynomials that have a large bias" (see Section 5).

The probabilistic proof systems $\mathcal{MA}$ and $\mathcal{AM}$. The quantified derandomization problem (discussed above) has an interesting analogous also in the case of probabilistic proof systems. Specifically, consider an MA or an AM proof system and assume that the number of bad random coins is extremely small (as above). *Can the corresponding set be placed in $\mathcal{NP}$?* Restricting our attention to systems in which the residual decision can be computed by an $\mathcal{AC}^0$ circuit, we show that the MA-version of the problem is in $\mathcal{NP}$, while the AM-version allows to place all $\mathcal{AM}$ in $\mathcal{NP}$. This dichotomy is indeed analogous to the dichotomy that exists between Theorem 1.3 and Theorem 1.4, and indeed the results regarding these proof system are proved by reductions to the latter theorems (see Section 7).

One-sided versus two-sided error versions. Most of the above discussion refer to the one-sided error version of the derandomization problem (as in Definition 1.1); nevertheless, Proposition 1.2 and the furthermore clause of Theorem 1.3 refer to the two-sided version in which one is given a circuit with $B(n) < 2^n/2$ exceptional inputs and needs to find an input that evaluates to the majority value. Moreover, we observe that a known transformation of hitting-set generators (which are black-box derandomizers for the one-sided error version) into derandomizers of the corresponding two-sided error classes is applicable in the conrtext of the quantified derandomization challenge. Specifically, as captured by Theorem 2.1, the transformation of Goldreich, Vadhan, and Wigderson [16] only increases the depth of the circuit (for the one-sided version) by two units (i.e., adding an unbounded `and`-gate and some negations) and only increases the value of the bounding function by a factor of n.

A key convention. As we have done so far, unless stated differently, we shall always let n denote the number of inputs to the given circuit.

2 Preliminaries

This work refers explicitly and implicitly to several different types of pseudorandom generators. Indeed, pseudorandomness is a general notion (or a theme) with many different incarnations that differ by (1) the class of tests (or distinguishers) fooled by the generator, (2) the complexity of the generator itself, and (3) the amount of stretch [14, Chap. 8]. In particular, we shall use standard tools such as limited-independence generators [11, 5] and small biased generators [25], and will refer to hitting set generators. In addition, we shall present and use a generalization of a known result of [16], which originally refers to the derandomization of $\mathcal{BPP}$ via a hitting set generator.

2.1 Standard tools

A t-wise independent generator of n-long sequences (over a set Σ) is an deterministic algorithm G that on input a random (seed) $s \in \{0,1\}^k$ outputs an n-long sequence $G(s)$ such that for every

$\leq i_1 < \cdots < i_t \leq n$ and every $\sigma_1, ..., \sigma_t \in \Sigma$ it holds that

$$\Pr\left[(\forall j \in [t]\, G(s)_{i_j} = \sigma_j\right] = |\Sigma|^{-t},$$

where $G(s)_i$ denotes the i^{th} element in $G(s)$. Such efficient generators of seed length $k = t \cdot \log_2 n$ can be constructed for any $|\Sigma| \leq n$ that is a power of two [5].

An ϵ-biased generator over $\{0,1\}^n$ is an deterministic algorithm G that on input a random (seed) $s \in \{0,1\}^k$ outputs an n-long bit string $G(s)$ such that for every non-empty set $I \subseteq [n]$ it holds that

$$\left|\mathrm{E}\left[(-1)^{\sum_{i\in I} G(s)_i}\right]\right| = \left|\Pr\left[\sum_{i\in I} G(s)_i = 0\right] - \Pr\left[\sum_{i\in I} G(s)_i = 1\right]\right| \leq \epsilon.$$

Such efficient generators of seed length $k = O(\log(n/\epsilon))$ can be constructed for any n (see, e.g., [25, 6]). We use the fact that ϵ-biased distributions are *ϵ-close in max-norm* to the uniform distribution (over $\{0,1\}^n$); that is, for every $\sigma \in \{0,1\}^n$ it holds that $\Pr[G(s) = \sigma] = 2^{-n} \pm \epsilon$ (see [6, Apdx] or [15, Sec. 1]). Indeed, for every $I \subseteq [n]$ and every $\sigma \in \{0,1\}^{|I|}$, it holds that $\Pr[G(s)_I = \sigma] = 2^{-|I|} \pm \epsilon$, where $G(s)_I$ denotes the projection of $G(s)$ on the bit positions in I.

2.2 Hitting Set Generators and two-sided error classes

Recall that most results stated in the introduction refer to the one-sided version as in Definition 1.1; however, as stated there, our results extend to the two-sided version in which one is given a circuit with $B(n) < 2^n/2$ exceptional inputs and needs to find an input that evaluates to the majority value. The standard derandomization challenge uses $B(n) = 2^n/3$, whereas the quantified version may allow any $B(n) < 2^n/2$. Theorem 2.1 provides some justification for our focus on the one-sided version.

Some of our results (e.g., the furthermore clause of Theorem 1.3) refer to the notion of a *hitting set generator*, but we apply this notion also to non-standard classes of circuits. Indeed, usually the notion of a hitting set generator is applied to a class of circuits of certain complexity (e.g., $\mathcal{P}$/poly or $\mathcal{AC}^0$) and is interpreted as referring only to circuits that evaluate to 1 with probability at least 1/2. Here we consider hitting set generators for classes of circuits of certain complexity that evaluate to 1 on at least $2^n - B(n)$ of the n-bit long inputs, for arbitrary functions B (rather than only for $B(n) = 2^{n-1}$). That is, a hitting set generator for such a class of circuits is a deterministic algorithm that on input 1^n outputs a set of n-bit strings such that for every n-input circuit C in the class the set contains a string on which C evaluates to 1. Using this terminology, we seize the opportunity to state a result that is implicit in [16].

Theorem 2.1 (Derandomization of two-sided error problems via a Hitting Set Generator): *Let $\mathcal{C}$ be a class of circuits that is closed under taking unbounded conjunctions and disjunctions* (i.e., closed under $\mathcal{AC}^0$). *Suppose that there exists a* (deterministic) *polynomial-time hitting set generator for the class of $\mathcal{C}$-circuits that evaluate to 1 on all but at most $B(n)$ of their possible n-bit assignments. Then, there exists a* (deterministic) *polynomial-time algorithm for deciding the majority value of a given $\mathcal{C}$-circuit that evaluate to the majority value on all but at most $B(n)/n$ of its possible n-bit assignments.*

The following proof sketch assumes familiarity with the proof of [16]; we only outline the additions requires for the proof of [16] in order to derive Theorem 2.1.

Proof Sketch: Loosely speaking, given a circuit C (as in the hypothesis), the derandomization procedure presented in [16] invokes a hitting set generator for a class of circuits that are n times larger than C and have a number of exceptional inputs that is n times larger than the number of exceptional inputs in C. Specifically, given a circuit $C : \{0,1\}^n \to \{0,1\}$ and a hitting set $S \subset \{0,1\}^n$ of size $m > n$ (obtained from the hitting set generator), the procedure evaluates C on m^2 inputs (derived from pairs in the set S) and runs a specific poly(m)-time algorithm (which finds a small dominating set in an auxiliary graph). The analysis of this procedure refers to circuits of the form $C_{\overline{y}}$, for every $\overline{y} = (y_1, ..., y_n) \in S^n$, such that $C_{(y_1,...,y_n)}(z) = \wedge_{i\in[n]}(C(y_i \oplus z) = \sigma)$, where σ is the majority value of C (which, by our hypothesis, is attained on $2^n - B(n)/n$ inputs). Our main observation here is that $C_{(y_1,...,y_n)}$ assumes the value σ on at least $2^n - B(n)$ inputs, whereas $C_{(y_1,...,y_n)}$ is obtained by taking the conjunction of n values computed by C (of $\neg C$). Hence, by our hypothesis, the circuit $C_{(y_1,...,y_n)}$ is in $\mathcal{C}$. ■

3 The class $\mathcal{AC}^0$: Proof of Theorem 1.3 and beyond

We start with a brief warm-up, which may be skipped. Next, we state and prove a result on pseudorandom restrictions, which is based on a new switching lemma and immediately implies Theorem 1.3.

A Warm-up: Hitting CNFs. The following result (which is implicit in [19]) demonstrates that for the quantified derandomization problem one can improve over the standard derandomization problem. Actually, when focusing on CNFs, the following result is stronger than Theorem 1.3.

Proposition 3.1 (CNFs and $B(n) = 2^n/\text{poly}(n)$): *Let ψ be an m-clause CNF over n variables that evaluates to 1 on at least a $1-\rho$ fraction of the possible n-bit strings. Let S_n be an ρ-biased sample space over $\{0,1\}^n$. Then, $\Pr_{s\in S_n}[\psi(s)=1] \geq 1 - 2\rho m$.*

Hence, if $\rho < 1/2m$, then a satisfying assignment for ψ can be found in polynomial time by scanning all sequences in a poly(n/ρ)-time constructible ρ-biased sample space. This establishes a result analogous to Theorem 1.3, but only for CNFs (and in this case even for $B(n) = 2^n/6p(n)$).[7]

Proof: By using the hypothesis, it follows that a uniformly chosen assignment (in $\{0,1\}^n$) satisfies each individual clause (of ψ) with probability at least $1-\rho$. Hence, the probability that a uniformly selected string in S_n does not satisfy such a clause is at most 2ρ, since an ϵ-biased assignment to the variables of a t-long clause hits the *unique* unsatisfying assignment with probability at most $2^{-t} + \epsilon$, whereas $2^{-t} \leq \rho$ (by the hypothesis). Applying a union bound, it follows that $\Pr_{s\in S}[\psi(s)=0] \leq 2\rho m$. ■

Our pivot: The effect of some pseudorandom restrictions. Turning to the general case of constant-depth circuits, we first show how to efficiently construct a sample space of (pseudorandom) restrictions such that each restriction leaves sufficiently many variables undetermined, while any $\mathcal{AC}^0$-circuit is simplified to a CNF of constant size by almost all restrictions. Hence, for any $\mathcal{AC}^0$-circuit, with very high probability, there is a significant gap between (1) the number of variables that are undetermined by the pseudorandom restrictions and (2) the number of variables

[7] The main claim of Theorem 1.3 is obvious for DNFs, but for the furthermore-clause (i.e., a hitting set generator for such DNFs) we seem to need techniques such as in Section 3.1.

that influence the corresponding restricted circuit. As noted in the introduction, although these pseudorandom restrictions do not necessarily preserve the fraction of satisfying assignments of the original circuit, this gap suffices for our application.

Recall that restrictions to n-variable Boolean functions are represented by n-long sequences over $\{0,1,*\}$ such that the i^{th} entry in the sequence indicates whether the i^{th} variable is assigned a value (in $\{0,1\}$) or is left undetermined (indicated by the symbol $*$).

Theorem 3.2 (pseudorandom restrictions with a gap between undetermined variables and influential variables): *For every two constants $c < 1$ and $d \in \mathbb{N}$ and any two polynomial p and q, there exists a constant κ and a $\mathrm{poly}(n)$-time algorithm of $O(\log n)$ randomness complexity that produces restrictions on n variables such that the following conditions hold:*

1. *The number of undetermined variables in each restriction is at least $2n^c$.*
2. *For any n-input circuit of depth d and size at most $p(n)$, with probability at least $1 - 1/q(n)$, the corresponding restricted circuit is a CNF of size at most κ.*

Theorem 1.3 follows easily from Theorem 3.2, because (with high probability) the number of variables that are undetermined but do not influence the restricted circuit is at least $2n^c - \kappa > n^c = \log_2 B(n)$, where $B(n)$ is the bound in the hypothesis of Theorem 1.3. In such a case, the restricted circuit must be the constant 1, since otherwise the number of inputs that evaluate to 0 exceeds $B(n)$ (in contradiction to the said hypothesis). Note that this argument is insensitive to the fact that the gap is between $2n^c$ and a constant (i.e., κ); all that matters is that the difference exceeds $n^c = \log_2 B(n)$.

Theorem 3.2 is proved by $d-2$ sequential applications of a corresponding switching lemma (see Lemma 3.1 below) and some additional work. Without the latter work, we would have obtained a weaker result in which the size of the restricted circuit is smaller than n^c. As stated above, this would have sufficed for deriving Theorem 1.3.

Theorem 3.2 is incomparable to other known results regarding pseudorandom restrictions. In particular, the restriction procedure of Ajtai and Wigderson [4], which is the first that uses pseudorandom (rather than random) restrictions, uses randomness $n^{\Omega(1)}$ but the pseudorandomly restricted circuits approximately preserve the acceptance probability of the original circuits. On the other extreme, the restriction procedure of Agrawal *et al.* [1] is deterministic (and also approximately preserves the said probability), but it uses the circuit in an essential way and keeps undetermined a smaller number of variables (i.e., n^c for a small constant $c > 0$ that depends on the circuit's size and depth).

3.1 A Switching Lemma with Logarithmic Randomness

The following switching lemma simplifies any depth-two circuit, while leaving a large number of variables alive, but it does not necessarily preserve the fraction of satisfying assignments of the original circuit. Again, this suffices for our application.

Lemma 3.3 (a switching lemma): *For any three constants $\alpha \in (0,1)$ and $\beta, \gamma > 0$, there exists a randomized polynomial-time algorithm of logarithmic randomness complexity that on input $(1^n, 1^m)$ such that $n < m$ outputs an element of $\{0,1,*\}^n$* (i.e., a restriction) *such that for any m-clause CNF over n variables, with probability at least $1 - m \cdot n^{-\gamma}$ over the choice of the restriction, the following two conditions hold:*

1. *The number of undetermined variables under this restriction is* $\Theta(n^{1-\alpha})$.

2. *The restricted function has a DNF of size* $O(n^{\beta}\log n)$.

The same holds when we consider all m-term DNFs and the possibility of computing the restricted function by a CNF.

Indeed, the lemma is meaningful only for $m < n^{\gamma}$. The constants hidden in the O- and Θ-notation depend on γ.

Proof: Let $\epsilon = 1/6n^{\gamma}$. We generate the final pseudorandom restriction in two stages. In the *first stage*, we use a use a pseudorandom restriction that lets each variable remain undetermined with probability $p_1 = n^{-\alpha}$ such that, with probability at least $1 - 3m\epsilon$, it holds that (1) the number of undetermined variables is $\Theta(n^{1-\alpha})$, and (2) the resulting simplified CNF has only clauses of constant length, where the constant upper bound is denoted c_1. This pseudorandom restriction is implemented by combining a pseudorandom generator of (c_1+1)-wise independent n-long sequences over $[1/p_1]$ (or rather over $[2^{\lceil \log_2(1/p_1)\rceil}]$) with an ϵ-biased sample space generator (for $\{0,1\}^n$).

In the *second stage*, we use a pseudorandom restriction that lets each variable remains undetermined with probability $p_2 = 1/10c_1c_2$ (where $c_2 \geq 1$ is a new constant)[8] such that, with probability at least $1 - 2\epsilon$, it holds that (1) the number of undetermined variables is $\Theta(n^{1-\alpha})$, and (2) the resulting function has a DNF of size $\widetilde{O}(n^{\beta})$. This pseudorandom restriction is implemented by using a poly(ϵ)-biased sample space generator, and its analysis relies on the switching lemma of Hastad [20]. Specifically, we shall show that for the current setting of parameters (i.e., for a constant probability p_2 of a variable remaining undetermined), it suffices to use a pseudorandom restriction that is "almost log-wise" independent (in max-norm).

(Before turning to a detailed description, we clarify that when dealing with CNF (resp., DNF) formulae, we shall assume that all clauses (resp., terms) are non-trivial; that is, no clause (resp., term) contains both a variable and its negation as literals. When we speak of simplifying a CNF after hitting it with a restriction, we refer to the process in which literals that are assigned the value 0 are omitted from any clause in which they occur, whereas a literal assigned the value 1 causes the omission of any clause in which they occur. If all clauses are omitted from a CNF, then the resulting CNF is set to 1 (and is viewed as "empty").)

We first detail the first stage. Recall that $p_1 = n^{-\alpha}$ and consider a pseudorandom restriction $\rho \in \{0,1,*\}^n$ that is generated by combining a (c_1+1)-wise independent sequence $\sigma \in [1/p_1]^n$ with an ϵ-biased sequence $\tau \in \{0,1\}^n$ as follows. For every $i \in [n]$, if $\sigma_i = 1$ then we set $\rho_i = *$, and otherwise (i.e., $\sigma_i \in \{2, ..., 1/p_1\}$) we set $\rho_i = \tau_i$. We now analyze the effect of this restriction on the clauses of the CNF, differentiating clauses of length at most $t = n^{\alpha/2}$ from longer clauses.

- Let C be a clause of length at most t. Then, the probability that more than c_1 variables of C remain undetermined by a restriction generated as above is at most

$$\binom{t}{c_1+1} \cdot p_1^{c_1+1} < (tp_1)^{c_1+1} = n^{-(c_1+1)\alpha/2}, \tag{1}$$

  where the LHS of Eq. (1) relies on the (c_1+1)-wise independence of the sequence σ. The RHS of Eq. (1) can be made smaller than $n^{-\gamma}/6 = \epsilon$ by a suitable choice of c_1 (i.e., $c_1 > 2\gamma/\alpha$).

[8] Again, we shall actually use $p_2 = 2^{-\lceil \log_2(10c_1c_2)\rceil}$.

- Let C be a clause of length at least t. We shall show that in this case, with probability at least $1-2\epsilon$, this clause is omitted from the simplified CNF (because it is set to 1 by the restriction).

  First note that the probability that less than $t/2$ variables of C are determined by a restriction generated as above is at most $t^{-c_1/3}$, since each variable is determine with probability $1-p_1 > 2/3$. Here we rely on the (c_1+1)-wise independence of the sequence σ and apply an c_1^{th} moment tail inequality.[9] But in such a case, with probability at least $1-2^{-t/2}-\epsilon > 1-2\epsilon$, the determined variables are assigned values that satisfy this clause (regardless of the undetermined variables). The latter assertion holds because the said event does not happen if and only if all determined literals are assigned 0, which happens with probability at most $2^{-t/2}$ if the assignment is chosen at random and thus happens with probability at most $2^{-t/2}+\epsilon$ when the assignment is ϵ-biased (because ϵ upper bounds the max-norm of the distance between a random assignment and an ϵ-biased assignment).[10]

Hence, the probability that some clause in the simplified CNF contains more that c_1 undetermined variables is at most $2m\epsilon$. Also note that, with probability at least $1-2\epsilon$, the number of undetermined variables is $n' = \Theta(p_1 n) = \Theta(n^{1-\alpha}) = o(n)$. Thus, the first stage is completed successfully with probability at least $1-2m\epsilon-2\epsilon > 1-3m\epsilon$.

We now turn to detail the analysis of the second stage. Recall that in the second stage, we use an ϵ'-biased sequence over $\{0,1\}^n$ in order to further restrict the remaining variables such that each variable remains undetermined with probability $p_2 = 1/10c_1c_2$, where $c_2 \geq 1$ is a suitable constant (e.g., $c_2 = 2^{\gamma/\beta+1}$) and $\epsilon' = \text{poly}(\epsilon)$. Specifically, let ψ_1 denote the *simplified CNF that results from the first stage*, and recall that with high probability ψ_1 has $n' = o(n)$ variables (or else we halt the restriction process). Now, we parse the ϵ'-biased sequence into blocks of length $\lceil \log_2(2/p_2) \rceil$, and use the i^{th} block for the i^{th} variable in ψ_1; e.g., the i^{th} variable remains undetermined if the i^{th} block is "monochromatic" (i.e., either all-zeros or all-ones) and is otherwise assigned the value of the first bit in the block.

To analyze the effect of this pseudorandom restriction, we follow a standard presentation (cf., e.g., [9, 38]) of Hastad's proof of the Switching Lemma [20], but select the random restriction "on the fly" (rather than selecting it up-front). This is done in order to keep track of the actual use of randomness in the restriction process, which in turn allows showing that log-wise independent pseudorandom restrictions has approximately the same effect as random restrictions, and ditto with respect to "almost (in max-norm) log-wise independent" pseudorandom restrictions.[11] Accordingly, Hastad's proof constructs a decision tree for $\psi = \psi_1$ using the following recursive procedure:

1. If the current CNF ψ is empty, then the procedure returns a decision tree consisting of a single vertex (a leaf) labeled 1 (i.e., a terminal).

2. Otherwise, let C be an arbitrary clause in ψ (e.g., the first one), and let V denote the variables appearing in C. Select a random restriction $\rho_V : V \to \{0,1,*\}$ for the variables of C such that each variable is undetermined with probability p_2 and is assigned a random Boolean value otherwise, where these $|V|$ choices are independent of one another. Let C' denote the clause resulting from C by applying this restriction.

[9] The actual expression is $\exp(c_1 \log c_1)/t^{-c_1/2}$.

[10] For $1-2^{-t/2}-\epsilon > 1-2\epsilon$, we use $2^{-t/2} < \epsilon$, which holds by the setting of t and ϵ (i.e., $t = n^{\Omega(1)}$ and $\epsilon = \text{poly}(n)$).

[11] The argument is somewhat reminiscent of [13].

3. If $C' \equiv 0$, then the procedure returns a decision tree consisting of a single vertex (a leaf) labeled 0.

4. If $C' \equiv 1$, then the procedure makes a recursive call on the residual CNF ψ' and returns the answer it gets, where ψ' is the CNF that results from ψ by restricting it with ρ_V (and simplifying). (In particular, this means omitting the clause C from ψ, applying the restriction ρ_V to the other clauses of ψ, and simplifying the resulting CNF.)

5. Otherwise (i.e., C' is undetermined), the procedure considers all possible assignments to the undetermined variables of C'. Denoting the set of these undetermined variables by V' (i.e., $V' = \{v \in V : \rho_V(v) = *\}$), for each assignment $\sigma : V' \to \{0,1\}$, we consider two sub-cases:

   (a) If σ satisfies C', then the procedure makes a recursive call on the resulting ψ', obtaining the decision tree T_σ, where ψ' is the CNF that results from ψ by restricting it with ρ_V and then by σ (i.e., variable $v \in V$ is assigned $\rho_V(v)$ if $\rho_V(v) \in \{0,1\}$ and $\sigma(v)$ otherwise). Again, ψ' is simplified, which in particular means omitting the clause C.

   (b) Otherwise (i.e., σ does not satisfy C'), the procedure sets T_σ to be a decision tree consisting of a single vertex (a leaf) labeled 0.

   The procedure forms a depth-$|V'|$ decision tree with internal vertices labeled by V' (e.g., all vertices in the i^{th} level are labeled by the i^{th} variable in V'), and attaches the decision tree T_σ to the leaf that corresponds to the path σ. Finally, the procedure returns the resulting decision tree (which combines the $2^{|V'|}$ aforementioned trees, i.e., the T_σ's).

In terms of our notation, Hastad [20] proved that the probability that the depth of the decision tree returned by the procedure exceeds D is upper bounded by $(5p_2c_1)^D = (1/2c_2)^D$. Hence, for $D = \beta \log_2 n$ and sufficiently large constant c_2 (i.e., $c_2 > 2^{(\gamma/\beta)-1}$), we have $(1/2c_2)^D < n^{-\gamma}/6$.

The above description refers to steps that are performed based on a random restriction $\rho : [n] \to \{0,1,*\}$ that is selected on the fly such that the values of ρ on different $i \in [n]$ are independent of one another. Now, our main task is to prove that the above assertion (regarding the depth of the final decision tree produced by the restriction procedure) remains valid also if we use an (almost) $O(D)$-wise independent pseudorandom restriction, where the O-notation hides dependence on p_2 (and c_1). Towards this end, we upper bound the depth of the (tree of) recursive calls performed by the foregoing restriction procedure, while noting that each recursive call consumes a constant number of random bits (i.e., at most $c_1\lceil \log_2(2/p_2) \rceil$ random bits).

The main observation is that in any non-trivial recursive call (i.e., one invoked on a non-empty formula), the depth of the constructed decision tree increases with constant probability (while the depth of the decision tree never decreases). Specifically, an increase in depth occurs when the random restriction selected in the current call does not determine the chosen clause C (i.e., the resulting clause C' is not a constant). This happens with probability that exceeds $\delta \stackrel{\text{def}}{=} (1-(1-p_2)^{c_1}) \cdot (1/2)^{c_1} \approx 2^{-c_1}/10c_2$, where the first factor accounts for the probability that not all variables of C are determined and the second factor account for the probability that none of the determined variables is set to a value that satisfies the clause. Hence, the probability that a (positioned) path in the recursion tree has depth greater than D' is exponentially vanishing in $\delta D' - D$. Let us detail the argument so to assist the verification that it can be applied when the choices are taken from an almost $O(D')$-wise independent sample space.

We stress that we are discussing two different trees: One is the *decision tree* constructed by the (randomized) recursive procedure, and the other is the *tree of recursion calls*. Both trees are random variables that depend on random choices made at the various recursive calls. The tree of recursive calls branches (only) in Step 5, when the restriction leaves the current clause undetermined (i.e., C' is not constant), which is also the only case in which the depth of the constructed decision tree increases. The branches in the tree of recursive calls correspond to sub-paths in the decision tree constructed by the procedure; indeed, the structure of the decision tree is "isomorphic" to the structure of the tree of recursive calls (in the sense that when the procedure constructs a partial tree decision on variables V' it branches to $2^{|V'|}$ calls that correspond to the leaves of this partial tree). In contrast, in Step 4 a single recursive call is made, whereas in the other cases no recursive calls are made at all. (Hence the number of different positional paths in the recursion tree equals the number of different positional paths in the decision tree.) We are concerned with establishing that, with high probability, the tree of recursive calls has depth at most D', while assuming that the depth of the constructed decision tree is at most D. Recalling that the latter event happens with very high probability, it follows that with very high probability both events occur. (Things get complicated because we wish to establish both probabilistic statements with respect to almost $O(D')$-wise independent pseudorandom restrictions.)[12]

Recall that the internal nodes of recursion tree have varying out-degree, where these degrees are powers of two (i.e., 1, 2, 4,...). However, for simplicity of the analysis, we replace each 2^d-star in this tree by a binary sub-tree of depth d (i.e., the depth 1 subtree of degree 2^d rooted at v is replaced by a binary tree of depth d rooted at v such that the original children of v are made leafs of the latter binary tree). The rest of the analysis refers to the resulting tree, in which each internal vertex has out-degree either one or two. We first consider the probability that this recursion tree has depth at most D' and the decision tree has depth at most D, when using a random restriction.

Fixing any potential positional path in the recursion tree (i.e., choice of branches for Step 5), the probability that the depth of recursion along this path exceeds D', assuming that the depth of the final decision tree returned by the recursion (along this path) is at most D, is at most $\binom{D'}{D} \cdot (1-\delta)^{D'-D}$, where this union bound refers to all possible cases in which at least $D'-D$ of the D' recursive calls fail to increase the depth of the constructed decision tree. The latter expression is upper bounded by $2^{H_2(D/D')\cdot D'} \cdot \exp(-\delta D'/2)$, where H_2 denotes the binary entropy function. Using $D' = \Omega(\delta \cdot (D + \gamma \log_2 n))$ and $H_2(\eta) = O(\eta \log(1/\eta))$, we upper bound $2^{H_2(D/D')\cdot D'} \cdot \exp(-\delta D'/2)$ by $2^{-D} \cdot n^{-\gamma}/4$. Applying a union bound (over all possible positional paths)[13], it follows that, with probability at least $1 - n^{-\gamma}/4$, the recursion tree has depth at most D' (given that the decision tree constructed by it has depth at most D). Hence, with probability at least $1 - n^{-\gamma}/2$, the recursion tree has depth at most D' and the decision tree constructed by it has depth at most D.

The above analysis refers to the case that we use a random restriction, but we observe that a similar probability bound holds when we use an $O(D')$-wise independent pseudorandom restriction.

[12] The problem is that Hastad's analysis of the depth of the decision tree refers to a random restriction, and it can be extended to an almost $O(D')$-wise independent pseudorandom restrictions provided that the recursion tree has depth at most D'. On the other hand, when analyzing the depth of the recursion tree we condition on the decision tree having depth at most D. This apparent vicious cycle is "broken" by considering only paths of length at most D' from the root of the recursion tree and arguing about events that correspond to this sub-tree; that is, we consider the partial decision tree constructed by this partial recursion tree and the case that this partial recursion tree is actually complete.

[13] Recall that there are at most 2^D positional paths in the recursive tree (which is "isomorphic" to the decision tree), provided that the decision tree has depth at most D.

In this case, we break the event in which the recursive procedure generates a decision tree of depth exceeding D into events that correspond to the paths in the decision tree generated by each path in the tree of recursive calls. Clearly, the probability that the path in the decision tree has length exceeding D is upper bounded by the probability that the entire procedure generates a decision tree of depth exceeding D. The key observation is that *the events that correspond to a fixed positional path (of length at most D') in the tree of recursive calls only depend on $c_1\lceil\log_2(20c_1c_2)\rceil\cdot D' = O(D')$ random variables*. Hence, the probability that recursion did not stop at depth at most D' along this path (given that the these recursive calls generate a path in the decision tree that has length at most D) is at most $2^{-D}\cdot n^{-\gamma}/4$ also when we use an $O(D')$-wise independent restriction. Likewise, when using an $O(D')$-wise independent restriction, the probability that the (first D') recursive calls (corresponding to this path) generated a path in the decision tree that has length greater than D is at most $(1/2c_2)^D$. (That is, we consider the path in the decision tree that is generated by at most D' recursive calls along the corresponding recursion path, regardless whether this path stops after at most D' calls or not.) Using an ϵ'-bias restriction instead of the $O(D')$-wise independent only adds an error term of $\exp(O(D'))\cdot\epsilon'$. Applying a union bound (over all possible positional paths)[14], it follows that, with probability at least $1-n^{-\gamma}/4-(1/c_2)^D-\exp(D+O(D'))\cdot\epsilon'$, the recursive tree does not exceed depth D' and the decision tree constructed by it does not exceed depth D.

Setting $c_1 = 2\gamma/\alpha$ and $c_2 = 2^{\gamma/\beta+1}$ (and using $\epsilon' = \exp(O(D'))$ and $D < D' = O(D)$), it follows that, with probability at least $1-n^{-\gamma}/2$, the decision tree built based on ϵ'-biased choices (rather than on totally random choices) has depth at most D, and so it can be computed by a DNF of size at most $D\cdot 2^D = O(n^\beta\log n)$.

We conclude that, with probability at least $1-m\cdot n^{-\gamma}$, the pseudorandom restriction generated by the combination of the two stages satisfies the following two conditions: (1) the number of surviving variables is $\Theta(n^{1-\alpha})$, and (2) the resulting function has a DNF of size $O(n^\beta\log n)$. The randomness complexity of the restriction procedure is $O(\log(n/\epsilon')) = O(D'+\log n) = O(\log n)$. ■

3.2 Proof of Theorem 3.2

Given a depth parameter d, we first apply the Switching Lemma (i.e., Lemma 3.3) for $d-2$ times, where in each iterations the depth decreases by one unit. This way we can obtain a weaker version of Theorem 3.2, in which the size of the restricted circuit is n^c rather than $O(1)$. The stronger result is obtained by applying Lemma 3.3 another one and a half times (where the half refers to Stage 1 in the pseudorandom restriction used in the proof of Lemma 3.3), and inferring that the further restricted circuit can be computed both by an $O(1)$-CNF and an $O(1)$-DNF, which implies that it can be computed by a circuit of constant size. Details follow.

For any constant $d \geq 2$, we consider a generic depth-d circuit (with a top AND-gate) and size at most $p(n)$, and proceed in $d-2$ iterations. In each iteration we apply the Switching Lemma (i.e., Lemma 3.3) to the two bottom levels of the current circuit, obtaining a circuit that is (possibly) slightly larger but is one level less deep (since the switching lemma allows us to merge two layers

[14] Formally, we consider positional paths in the recursion tree such that each path (or recursive calls) is truncated either when its length exceeds D' or when it generates a path of length exceeding D in the corresponding decision tree. The path of recursive calls is actually trunacted either due to more than D' recursive calls (while the path in the generated decision tree has length at most D) or due to generating a path of length exceeding D in the decision tree (where this path is generated by the first D' recursicve calls).

(i.e., the next-to-bottom layer with the one above it)). Specifically, we set $\alpha = (1-c)/d$ and $\gamma = O(d \log_n p(n))$ (and set β arbitrarily, e.g., $\beta = 0.9$). The setting of α guarantees that after $d-2$ iterations we will be left with at least $\Omega(n^{1-(d-2)\alpha}) > 2n^c$ undetermined variables, whereas the setting of γ guarantees that the accumulated error probability is sufficiently small (even if we transform the original depth-d circuit into a depth-d formula of fan-in $p(n)$ before starting the switching process).[15] Hence, after i iterations, we obtain a formula of depth $d-i$ and size $p(n)^d \cdot n^i$. (Actually, at the last iteration we may select a smaller $\beta > 0$, and so obtain a CNF of size n^β.)

At this point, we apply Stage 1 of the pseudorandom restriction used in the proof of Lemma 3.3, and obtain a CNF in which each clause is of constant size. Applying Stage 2 of the lemma, obtaining a DNF, and applying Stage 1 of the lemma to it, we obtain a DNF in which each term is of constant size. Hence, for some constant κ, the corresponding function can be computed both by a κ-CNF and a κ-DNF. It follows that this function can be computed by a decision tree of depth κ^2 (see [22, Sec. 14.2]), which implies that it can be computed by a CNF of size $\exp(\kappa^2)$.

Overall, the amount of randomness used in the process is $O(d \log n)$, and the theorem follows. (Indeed, this sample space may contain a small (polynomial) fraction of pseudorandom restrictions that determine too many variables, but these restrictions can be replaced by any other restriction that determines fewer variables (e.g., the restriction that leaves all variables undetermined).) ■

Digest. For every constant $c < 1$ and $d, e \in \mathbb{N}$, the above proof of Theorem 3.2 yields a hitting set generator for the class of depth-d circuits of size at most n^e that evaluate to 1 on at least $2^n - 2^{n^c}$ of their inputs. The hitting set generator consists of generating d sample spaces of pseudorandom restrictions (as in the proof of Lemma 3.3) and assigning the remaining undetermined variables arbitrarily (say, setting all to 1), while relying on the fact that in this case the restricted circuit always outputs 1. Recall that each of the pseudorandom restrictions is generated by combining a constant-wise independent sample space and two small biased sample spaces. Hence, in total, d constant-wise independent sample spaces and $2d$ small biased sample spaces are used, and their results are combined to form poly(n)-sized sample space over $\{0,1\}^n$. More specifically, the sample spaces that correspond to the d applications of Lemma 3.3 generate d sample spaces over $\{0,1,*\}^n$, denoted $S_1, ..., S_d$. The resulting hitting set corresponds to a Cartesian product of these d sample spaces such that for every $s_1 \in S_1$,..., $s_d \in S_d$, the hitting set contains the n-bit string s such that for each $i \in [n]$ the i^{th} bit of s equals the first Boolean value in the sequence $(s_{1,i}, ..., \sigma_{d,i}, 1)$, where $s_{j,i}$ is the i^{th} element of $s_j \in \{0,1,*\}^n$.

3.3 The case of $B(n) = \exp(n/\text{poly}(\log n))$

In the initial posting of this work, we posed the challenge of extending Theorem 1.3 to $B(n) = 2^{0.01n}$, while admitting that it may be the case that such an extension (or just a stronger one – say, to a level of $B(n) = 2^{0.99n}$) would yield a (deterministic) polynomial-time algorithm for approximate counting $\mathcal{AC}^0$. Here, we show that the latter is indeed the case. In fact, a stronger result holds.

Theorem 3.4 (the case of $\mathcal{AC}^0$ and $B(n) = \exp(n/\text{poly}(\log n))$):[16] *Let $\mathcal{AC}^0_{d,p}$ denote the class of depth-d circuits of size at most $p(n)$. Suppose that there exists a constant $\epsilon > 0$ such that for every*

[15] Such a transformation facilitates the iterative process of applying the switching lemma and collapsing two adjacent levels that use the same type of gates.

[16] The current version corrects inaccuracies in the statement of the result made in 2014. These inaccuracies were pointed out by Roei Tell.

sufficiently large constant d and every polynomial p the $(\mathcal{AC}^0_{d,p}, 2^{n/(\log n)^{(1-\epsilon)\cdot d}})$-search problem can be solved in (deterministic) *polynomial-time. Then, there exists* (deterministic) *polynomial-time algorithms that find a satisfying assignment to any $\mathcal{AC}^0$ circuit that is satisfied by a majority of its inputs.*

Proof: We reduce the $(\mathcal{AC}^0, 2^{n-1})$-search problem to an $(\mathcal{AC}^0, 2^{n/\mathrm{poly}(\log n)})$-search problem as follows. The reduction proceeds in three steps: In the first step we drastically reduce the length of the input (from n to $\mathrm{poly}(\log n)$), while preserving the fraction of bad inputs by using a suitable pseudorandom generator [26, 29]. In the second step we drastically reduce the relative number of bad inputs, by using a suitable extractor [37]. And finally, we derive the desired $\mathcal{AC}^0$ circuit.

We start with an $\mathcal{AC}^0_{d,p}$ circuit $C : \{0,1\}^n \to \{0,1\}$ so that $\Pr_x[C(x) = 1] \geq 1/2$. Setting $m = (\log n)^{2d+6}$ and using a suitable pseudorandom generator $G : \{0,1\}^m \to \{0,1\}^n$, which is computable by $\mathrm{poly}(n)$-circuits of depth $d+4$, we obtain a circuit $C'(s) = C(G(s))$ that is computable by $\mathrm{poly}(n)$-circuits of depth $d' = 2d+4$ so that $\Pr_s[C'(s) = 1] > 1/4$.

Next, for any constant $c > 2$, setting $t = m^c = (\log n)^{2c(d+3)}$, we employ error reduction using a $(m^2, 0.1)$-extractor $E : \{0,1\}^t \times \{0,1\}^{O(\log n)} \to \{0,1\}^m$ that is computable by a $\mathrm{poly}(n)$-size circuit of depth $2c \cdot (d+3)$ (e.g., Trevisan's [37]).[17] We obtain a $\mathrm{poly}(n)$-size circuit $C'' : \{0,1\}^t \to \{0,1\}$ of depth $d'' = d' + 2c \cdot (d+3) < (2c+2) \cdot (d+3)$ such that $|\{r \in \{0,1\}^t : C''(r) \neq 1\}| \leq 2^{m^2} = 2^{t^{2/c}}$. Specifically, $C''(r) = \vee_i C'(s_i)$, where $s_i = E(r,i)$ for $i \in \{0,1\}^{O(\log n)}$.

Finally, we obtain a circuit $C''' : \{0,1\}^n \to \{0,1\}$ of depth $d''' = d'' < (2c+2)(d+3)$ and size $p'''(n) = \mathrm{poly}(n)$, by letting $C'''(r_1, ..., r_{n/t}) = \vee_j C''(r_j)$. Note that the number of bad inputs for this C''' is at most $(2^{t^{2/c}})^{n/t} < 2^{n/t^{1-(2/c)}} = 2^{n/(\log n)^{(2c-4)(d+3)}}$, since $t = (\log n)^{2c(d+3)}$. Hence, finding a satisfying assignment to C reduces to finding a satisfying assignment to C''', which means that the $(\mathcal{AC}^0_{d,p}, 2^{n-1})$-search problem reduces to the $(\mathcal{AC}^0_{d''',p'''}, 2^{n/(\log n)^{(1-\epsilon)d'''}})$-search problem, since $\frac{(2c-4)(d+3)}{d'''} > \frac{2c-4}{2c+2} \geq 1-\epsilon$ for a sufficiently large c (i.e., for $c \geq (3-\epsilon)/\epsilon$). The theorem follows. ∎

4 The class $\mathcal{AC}^0[2]$: Proof of a generalization of Theorem 1.4

The proof of Theorem 1.4 relies on the fact that the corresponding class allows for extremely strong error reduction, reaching a point that the number of bad (n-bit) inputs is at most $B(n) = \exp(n^c)$, for any $c > 0$. The following definition provides sufficient conditions for such an error reduction.

Definition 4.1 (sufficiently strong class): *We say that a class $\mathcal{C}$ of circuits is* sufficiently strong *if it satisfies the following conditions:*

1. *The class $\mathcal{C}$ contains circuits for computing approximate majority; that is, it contains circuits that compute majority correctly on inputs that have at least a 51%-majority in some direction.*[18]

2. *The class $\mathcal{C}$ is closed under polynomially bounded parallelism and sequential composition. That is, if $\mathcal{C}$ contains circuits for computing $F : \{0,1\}^m \to \{0,1\}^\ell$ and $G : [\mathrm{poly}(n)] \times \{0,1\}^n \to$*

[17] Note that Trevisan's extractor is actually better than needed, since we can afford a seed of length $O(\log n) = O(t^{1/2c(d+3)})$.

[18] That is, inputs $x = x_1 \cdots x_n$ such that either $|\{i \in [n] : x_i = 1\}| \geq 0.51 \cdot n$ or $|\{i \in [n] : x_i = 0\}| \geq 0.51 \cdot n$.

$\{0,1\}^m$, *then* $\mathcal{C}$ *contains circuits for computing the composition of* G *with* m *parallel executions of* F (i.e., the mapping $x \mapsto (F(G(1,x)),...,F(G(\mathrm{poly}(|x|),x))$.

3. *For every constant* $\alpha > 0$ *there exists a constant* $\beta > 0$ *such that the class* $\mathcal{C}$ *contains circuits for computing an* $(n^\alpha, 0.1)$*-extractor* $E : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^{n^\beta}$*; that is, an extractor of logarithmic seed length for min-entropy* n^α*, error* (or statistical deviation) 0.1*, and output length* n^β (cf., e.g., [33]).[19]

Moreover, in each of these cases, the desired circuit can be computed in time that is polynomial in its size.[20]

While the class $\mathcal{AC}^0$ is not sufficiently strong (i.e., it cannot compute a randomness extractor with parameters as in Condition 3; cf. [34, Thm. 6.4]), the class $\mathcal{AC}^0[2]$ is sufficiently strong (see Remark 4.3): In particular, it can compute Trevisan's extractor [37], which satisfies Condition 3. Recall that $\mathcal{AC}^0 \subseteq \mathcal{AC}^0[2]$ contains circuits for approximate majority (and that they can be constructed in polynomial-time; cf. [2, 3, 35]).

Theorem 4.2 (sufficiently strong classes and $B(n) = \exp(n^{\Omega(1)})$): *Let* $\mathcal{C}$ *be a sufficiently strong class, and let* $B(n) = 2^{n^c}$ *for some constant* $c > 0$*. Suppose that the* $(\mathcal{C}, B)$*-search problem can be solved in* (deterministic) *polynomial-time. Then, there exists a* (deterministic) *polynomial-time algorithm that finds a satisfying assignment to any circuit in* $\mathcal{C}$ *that is satisfied by a majority of its inputs.*

A weak version of Theorem 1.4 (in which the same $c > 0$ is used for all $\mathcal{AC}^0_{d,p}[2]$'s) follows by the fact that $\mathcal{AC}^0[2]$ is a sufficiently strong class (see Remark 4.3). In order to prove Theorem 1.4 as stated, we observe that the "depth overhead" introduced by the following proof is (a constant that is) independent of $\alpha = c$ (since the circuits computing the extractor are of depth one, and the circuits computing approximate parity are of depth three); ditto for the size overhead. By trivial error reduction, which is possible for the class $\mathcal{C}$, we may assume that we are given circuits that evaluate to 1 on at least two-thirds of their inputs (rather than at least half their inputs).[21]

Proof: Let C be an m-variable circuit in the class $\mathcal{C}$ and suppose that $\Pr_{r\in\{0,1\}^m}[C(r)=1] > 2/3$. For a constant $c > 0$ as in the theorem's hypothesis, set $\alpha = c$, and let $\beta > 0$ be as is guaranteed for α (in the definition of a sufficiently strong class). Now, let $n = m^{1/\beta}$, and consider an n-variable circuit C' that, on input $x \in \{0,1\}^n$, computes the (approximate) majority vote among the values $C(E(x,s))$ for all $s \in \{0,1\}^{O(\log n)}$, where E is the extractor guaranteed in the definition of a sufficiently strong class. Indeed, we shall use an approximate majority circuit instead of the majority function. Hence, the circuit C' consists of a bottom layer of circuits that, on input x, compute $y_s \leftarrow E(x,s)$ for each $s \in \{0,1\}^{O(\log |x|)}$, an intermediate level that computes $z_s \leftarrow C(y_s)$ (for each s), and a top level that computes an approximate majority of the z_s's. The circuit C' can be constructed in polynomial-time, since the bottom and top levels can be so constructed (per the hypothesis regarding the class).

[19] In fact, it suffices to require that for every $s \in \{0,1\}^{O(\log n)}$, the class $\mathcal{C}$ contains circuits for computing the residual function $E(\cdot, s)$. However, combined with Condition 2, this weaker condition implies the stronger Condition 3.

[20] Of course, in the case of Condition 2, the desired circuit (for the composition of F and G) is computed efficiently when given circuits for G and F.

[21] Alternatively, we can adapt the argument below and use an approximate threshold circuit that accepts inputs that have at least a fraction of 49% ones and rejects inputs for which the fraction is lower than 47%.

Turning to the analysis, we note that there are less than $2^{n^\alpha} = 2^{n^c}$ strings $x \in \{0,1\}^n$ such that $\Pr_{s\in\{0,1\}^{O(\log n)}}[C(E(x,s)=1] < 0.51$, because otherwise taking a uniform distribution over the set of bad x's yields a distribution X of min-entropy at least n^α such that the statistical difference between $E(X, U_{O(\log n)})$ and U_m is at least $(2/3) - 0.51 > 0.1$ (where U_ℓ denotes the uniform distribution over $\{0,1\}^\ell$). It follows that there are at most 2^{n^c} strings $x \in \{0,1\}^n$ such that $C'(x) = 0$, and by applying the algorithm in the theorem's hypothesis we find an x such that $C'(x) = 1$. In this case (i.e., $C'(x) = 1$), it holds that $\Pr_{s\in\{0,1\}^{O(\log n)}}[C(E(x,s)=1] > 0.49$, and by using this x and trying all $s \in \{0,1\}^{O(\log |x|)}$ we find a string $E(x,s)$ on which C evaluates to 1. The claim of the theorem follows. ■

Remark 4.3 (the case of $\mathcal{AC}^0[2]$): *In the case of $\mathcal{AC}^0[2]$, we can use Trevisan's extractor [37] in the role of the extractor postulated in Condition 3 of Definition 4.1. Recall that the computation of Trevisan's extractor requires a construction of "weak designs" and an adequate error correcting code, and the computation of bits in the encoding w.r.t the latter. The constructions themselves can be performed in polynomial-time, whereas the code itself is linear and thus bits in the encoding can be computed by parity gates. In fact, for any $s \in \{0,1\}^{O(\log n)}$, each bit in the extracted output $E(x,s)$ is a linear combination of the bits of x, where the combination itself is determined by s* (according to the aforementioned design)*. Hence, in this case, the bottom level consists of computing partial sums* (mod 2) *of the bits of x, where these partial sums correspond to bits in a suitable codeword* (and that the corresponding partial subsets can be computed in polynomial-time)*.*

Remark 4.4 (Remark 4.3 applied to $\mathcal{AC}^0$): *Applying the above construction to a $\mathcal{AC}^0$-circuit of depth d, we obtain a circuit of depth $d+3$ with XOR-gates at the bottom and $d+2$ layers of AND/OR-gates. The latter $d+2$ layer result from combining the original depth-d circuit with a depth-three circuit computing approximate majority [2, 35].*

A black-box version of Theorem 4.2. As stated, Theorem 4.2 refers to non-black-box algorithms that get a circuit (which is guaranteed to have a certain number of satisfying assignments) and output a satisfying assignment for it (i.e., an assignment that satisfies this circuit). However, the above proof supports also a black-box version, which is analogous to the furthermore claim of Theorem 1.3.

Theorem 4.5 (simplified version):[22] *Let $\mathcal{C}$ be a sufficiently strong class, and suppose that there exists a constant $c > 0$ and a* (deterministic) *polynomial-time algorithm that on input 1^n outputs a set of n-bit strings S_n such that every circuit C that satisfies the* (input) *condition of Theorem 4.2 evaluates to 1 on some string in S_n. Then, there exists a* (deterministic) *polynomial-time algorithm that on input 1^n outputs a set of n-bit strings S'_n such that every circuit $C \in \mathcal{C}$ that is satisfied by the majority of the assignments in $\{0,1\}^n$ is satisfied by some string in S'_n.*

We note that the hitting set generator (for the class $\mathcal{C}$) that is guaranteed by the conclusion of Theorem 4.5 yields a (deterministic) polynomial-time approximate counter (with $2^n/\mathrm{poly}(n)$ additive deviation) for the class $\mathcal{C}$. This can be shown by combining the following two observations:

[22] More generally, we may assume a hitting set generator that is given some parameters of the circuit (e.g., its size and depth, as in the furthermore clause of Theorem 1.3). In such a case, the conclusion will also refer to such hitting set generators (i.e., they will have to be given the same parameters).

1. For a sufficiently strong class $\mathcal{C}$, approximate counting for $\mathcal{C}$ reduces to distinguishing circuits (in $\mathcal{C}$) that are satisfied by at least a $1-\exp(-\sqrt{n})$ fraction of their inputs from circuits (in $\mathcal{C}$) that are satisfied by at most a $\exp(-\sqrt{n})$ fraction of their inputs.

2. For any class $\mathcal{C}$ that is closed under taking unbounded conjunctions and disjunctions (i.e., closed under $\mathcal{AC}^0$), a hitting set generator implies a distinguisher as in the prior item (see Theorem 2.1).

5 The class of GF(2) Polynomials: Proof of Theorem 1.6

Let us start by restating the theorem, while explicitly referring to the notion of a hitting set generator.

Theorem 5.1 (Theorem 1.6, restated): *For every constant c, there exists a* poly(n)*-time hitting set generator for the class of n-variate polynomials p over* GF(2) *that evaluate to 0 on at most a $c \cdot 2^{-\deg(p)}$ fraction of their inputs, where* $\deg(p)$ *denotes the degree of p.*

Note that the case of $c < 1$ is trivial, since in this case the polynomial must be identically 1. We stress that the hitting set applies to all degrees. Theorem 5.1 is proved by using a refinement of Lemma 4 in Viola [36], which refers to "fooling polynomials that have a large bias". We define the bias of a function $f : \{0,1\}^n \to \{0,1\}$ as the absolute value of the expectation of $(-1)^{f(r)}$ when r is uniformly distributed in $\{0,1\}^n$. We say that a distribution W ϵ-fools f if it holds that $|\mathrm{E}[(-1)^{f(W)}] - \mathrm{E}[(-1)^{f(U)}]| \leq \epsilon$, where U denotes the uniform distribution over $\{0,1\}^n$. Indeed, if f is unbiased (i.e., has bias zero, as when f is a non-constant linear function) and W ϵ-fools it, then it holds that $|\mathrm{E}[(-1)^{f(W)}]| \leq \epsilon$ (and if W ϵ-fools all (non-constant) linear functions, then it is ϵ-biased).

Lemma 5.2 ([36, Lem. 4], refined): *Let p be a degree $d+1$ polynomial over* GF(2) *with bias at least $1-\delta \geq 1/2$ and suppose that W ϵ-fools every degree d polynomial that has bias at least $1-2\delta$. Then, W $(\epsilon/(1-\delta))$-fools p.*

Proof: Going through Viola's proof (see details in Appendix B), note that it defined polynomials $p'_z(x) = p(x+z) + p(x)$ and relies on the hypothesis that W ϵ-fools each of them. As noted by Viola, each p'_z has degree at most d (since the degree $d+1$ terms cancel out). We note that the bias of each p'_z is at least $1-2\delta$:

$$\begin{aligned}
\left|\mathrm{E}\left[(-1)^{p'_z(U)}\right]\right| &= \left|\mathrm{E}\left[(-1)^{p(U+z)+p(U)}\right]\right| \\
&= |1 - 2 \cdot \Pr\left[p(U+z) + p(U) = 1\right]| \\
&= |1 - 2 \cdot \Pr[p(U+z) \neq p(U)]| \\
&\geq 1 - 4 \cdot \Pr[p(U) = b_{\min}]
\end{aligned}$$

where $b_{\min} \in \{0,1\}$ is such that $\Pr[p(U)=b_{\min}] \leq 1/2$, and the inequality uses $\Pr[p(U+z) \neq p(U)] \leq \Pr[P(U+z)=b_{\min} \vee p(U)=b_{\min}]$ (which in turn is upper bounded by $\Pr[P(U+z)=b_{\min}]+\Pr[p(U)=b_{\min}] = 2\cdot\Pr[p(U)=b_{\min}]$) as well as $\Pr[p(U)=b_{\min}] = (1-|\mathrm{E}[(-1)^{p(U)}]|)/2 \leq \delta/2 \leq 1/4$. Combining $|\mathrm{E}[(-1)^{p'_z(U)}]| \geq 1 - 4 \cdot \Pr[p(U) = b_{\min}]$ with $\Pr[p(U)=b_{\min}] \leq \delta/2$, we get $|\mathrm{E}[(-1)^{p'_z(U)}]| \geq 1-2\delta$, and the lemma follows. ■

Proof of Theorem 5.1. For $c' = 2 + \lceil \log_2 c \rceil$ and any $\epsilon > 0$, let X be a distribution over n-bit long strings that ϵ-fools n-variate polynomials of degree c', and let p be a polynomial as in the hypothesis. Hence, if p has degree d, then it has bias at least $1 - 2c \cdot 2^{-d}$. We shall show that X 2ϵ-fools p. This will be done by iteratively applying Lemma 5.2, starting with the hypothesis that X ϵ-fools all degree c' polynomials (and, in particular, all degree c' polynomials that have bias at least $1 - 2c \cdot 2^{-c'} > 1/2$).

Recall that Lemma 5.2 asserts that *if a distribution ϵ'-fools all degree d polynomials of bias at least $1 - 2\delta$, then it $\epsilon'/(1-\delta)$-fools all degree $d+1$ polynomials of bias at least $1 - \delta$.* We start by setting $\epsilon_{c'} = \epsilon$, and using the hypothesis that X $\epsilon_{c'}$-fools all degree c' polynomials that have bias at least $1 - 2c \cdot 2^{-c'}$. For $i = c', ..., d-1$, we infer (by Lemma 5.2)[23] that X ϵ_{i+1}-fools all degree $i+1$ polynomials that have bias at least $1 - c \cdot 2^{-i} = 1 - 2c \cdot 2^{-(i+1)}$, where $\epsilon_{i+1} = \epsilon_i/(1 - c \cdot 2^{-i})$. Hence, X ϵ_d-fools all degree d polynomials that have bias at least $1 - 2c \cdot 2^{-d}$, where $\epsilon_d = \epsilon / \prod_{i=c'}^{d-1}(1 - c \cdot 2^{-i})$, which is at most $\epsilon/(1 - c\sum_{i=c'}^{d-1} 2^{-i}) < \epsilon/(1 - 2c \cdot 2^{-c'})$. Using $c' = 2 + \lceil \log_2 c \rceil \geq 2 + \log_2 c$, we get $\epsilon_d < 2\epsilon$, and infer that X 2ϵ-fools polynomials of degree d that have bias at least $1 - 2c \cdot 2^{-d}$, which in particular means that X 2ϵ-fools p. Now, setting $\epsilon = 1/3$ and using a pseudorandom generator that 1/3-fools all polynomials of degree $2 + \lceil \log_2 c \rceil$, we are done. ■

6 Partial derandomization results regarding $\mathcal{AC}^0[2]$

Below are some partial derandomization results regarding the class $\mathcal{AC}^0[2]$ (and various bounding functions B). The case analysis refers to the levels at which XOR gates appear, and we may assume (w.l.o.g.) that they do not appear in consecutive levels. Note that we cover all possible cases *only* for depth-two $\mathcal{AC}^0[2]$ circuits, which are quite easy to handle anyhow. Except for Case 1, which refers to any constant depth, all other cases are confined to depth-three.

Case 1: Only the top gate is an XOR gate. In this case the circuit is an XOR of $\mathcal{AC}^0$ circuits, denoted $C_1, ..., C_m$, and we will show that the algorithm used in the proof of Theorem 1.3 will do.

We first apply the switching lemma (i.e., Lemma 3.3) to simplify the C_i's. At the last iteration, when the resulting C_i's are already of depth two, we write the result (which is a decision tree of logarithmic depth) as a sum of polynomially many products (each of logarithmic length). So we get a sum of sums of products, which is just a sum of products. Lastly, we apply Stage 1 of the pseudorandom restriction (used in the proof of Lemma 3.3) and obtain a sum of products that are each of constant length (i.e., length at most c_1). Hence, for any subexponential bounding function B (e.g., $B(n) = \exp(n^{0.99})$), the resulting circuit must be the constant 1, or else it evaluates to 0 with probability at least 2^{-c_1} (in violation of our bound on the number of inputs that make the original circuit evaluate to 0). Alternatively, we hit the resulting polynomial of degree c_1 by a pseudorandom generator that fools all such polynomials (cf. [36]).[24]

Case 2: Depth-three circuits with XOR-gates only at the bottom. For simplicity, we consider only circuits in which all bottom gates are XOR-gates. In this case the circuit is a CNF/DNF of XOR-gates. In case the circuit is a CNF of parity gates we proceed as in the case of pure CNFs (and hence it suffices to assume that the circuit accepts with probability at least

[23] Here we use the fact that X was already established as ϵ_i-fooling all degree i polynomials that have bias at least $1 - 2c \cdot 2^{-i}$.

[24] But in this case, we use a slightly different algorithm than the one used in the proof of Theorem 1.3.

$1 - 1/3m$, where m is the fan-in of the top AND-gate). Specifically, we infer that each of the m sub-circuits (i.e., the ORs-of-XORs) must evaluate to 1 with probability at least $1 - 1/3m$, under the uniform distribution. Now, when we feed such a sub-circuit with an $1/6m$-biased distribution, it evaluates to 0 with probability at most $1/2m$ (since this event is a conjunction of linear conditions on the small biased distribution, whereas this conjunction is satisfied with probability at most $1/3m$ under the uniform distribution).[25] Applying a union bound, we infer that when fed with an $1/6m$-biased distribution, the entire circuit evaluates to 1 with probability at least $1/2$.

In case the circuit is a DNF of parity gates, it suffices to assume that the circuit accepts with probability at least $1/3$. In this case, there must be a sub-circuit that evaluates to 1 with probability at least $1/3m$ (under the uniform distribution), and when using an $1/6m$-biased sample space this sub-circuit evaluates to 1 with probability at least $1/6m$ (under the small biased distribution).

Note that, by Remark 4.4, extending Case 2 (i.e., XOR-gates only at the bottom) to circuits of depth five even just for some subexponential bounding function B (e.g., $B(n) = \exp(n^{0.01})$) would yield a hitting set generator for CNFs.

Case 3: Depth-three circuits with XOR-gates only in the middle. For simplicity, we consider only circuits in which all intermediate gates are XOR-gates. In this case, the circuit is an AND/OR of XORs of AND/OR-gates, where w.l.o.g. the lowest level is of AND-gates.

In case the top gate is an AND-gate, we just need to hit each sub-circuit (i.e., an XOR-of-ANDs) with high probability, and this is done by reduction to Case 1. Actually, we need to apply the same random choices in each of the sub-circuits (to which we apply Case 1).[26]

In case the top gate is an OR-gate, it is tempting to say that it suffices to hit some sub-circuit, and so we may select an arbitrary one that is satisfied with probability at least $1/2m$ and focus on it. But in this case we cannot apply Case 1. Instead, we first apply Stage 1 of the pseudorandom restriction used in the proof of Lemma 3.3, and obtain a circuit of the form OR-XOR-AND but now each AND has constant length (i.e., length at most c_1). We use the hypothesis that the original circuit had few inputs that evaluate to 0 in order to infer that the reduced circuit evaluates to 0 on less than half of its possible inputs. Now, we infer that one of the XOR-AND sub-circuits is satisfied with probability at least $1/2m$, and focus on it. Noting that it computes a polynomial of degree c_1, and applying a pseudorandom generator that fools all such polynomials, we are done.

Case 4: Circuits that are a XOR of AND/OR-gates of XOR-gates. Indeed, we may assume w.l.o.g. that it is an XOR-of-AND-of-XORs. We note that each product corresponds to an Affine subspace, and so we can replace it by a product over a basis of this subspace. Now, by omitting products that refer to subspaces of dimension greater than $k + \log_2 m$ we only introduce an error of 2^{-k}, which we can afford as long as $k \leq n - \log_2 B(n)$. The fact that we lose an additive term of $\log_2 m$ (no matter which k we pick) is a problem, since otherwise we could have applied Theorem 5.1.

Specifically, suppose that, for some k, omitting all products that refer to subspaces of dimension greater than $k + O(1)$ only introduces an error of $O(2^{-k})$. Then, the resulting circuit corresponds to a polynomial of degree $d = k + O(1)$ that evaluates to 1 on at least $1 - (1 + O(1)) \cdot 2^{-k} = 1 - O(2^{-d})$ fraction of its domain. Invoking Theorem 5.1 we would have been done. Unfortunately, the above assumption cannot be justified, and so the current case is left open (and seems the actual obstacle towards completing the treatment of depth-there circuits with parity).

[25] Note that applying a full-rank linear transformation to an ϵ-biased distribution yields an ϵ-biased distribution.

[26] An alternative description can be obtained by adapting the treatment of the case of a top OR-gate (see next).

7 The probabilistic proof systems $\mathcal{MA}$ and $\mathcal{AM}$

In this work we focus on the two most restricted forms of interactive proof systems (introduced in full generality in [17]): (1) MA-proof systems, which are randomized non-interactive proof systems (indeed the true randomized version of $\mathcal{NP}$), and (2) AM-proof systems, which are randomized and interactive proof systems in which the prover sends a single message in response to a random query of the verifier. The corresponding classes of sets that are acceptable by such proof systems were defined in [8], and we review these definition below, while considering a few variants.

Before doing so, we note that the new quantitative framework and a couple of simple observations lead to interesting new problems about these important classes. We also note that while our current results are obtained by reduction or analogy to Theorems 1.3 and 1.4, we do not use nondeterminism in order to assist in the actual derandomization process. The potential of this possibility is demonstrated in the observation that in the context of AM-proof systems one may assume, w.l.o.g, that the final verifier decision is computed by a CNF (see proof of Theorem 7.4).

Definitions and simple observations. Since we wish to maintain n as the amount of randomness used, we denote the input length by k, and let $n = n(k)$ and $m = m(k)$ denote the amount of randomness used by the verifier and the length of the prover's message, respectively. We stress that, here too, the upper bound on the number of exceptional random choices, denoted B, is a function of the number of random choices (i.e., n); that is, $B(n)$ is always a fraction of 2^n. The one-sided and two-sided error versions of a class are subscripted by 1 and 2, respectively.

Definition 7.1 ($\mathcal{MA}$ and $\mathcal{MA}^0$): *A set S is in $\mathcal{MA}_2$ if there exists a deterministic polynomial-time verification procedure V and two polynomials $n, m : \mathbb{N} \to \mathbb{N}$ such that the following two conditions hold:*

Completeness: *For every $x \in S$ there exists $w \in \{0,1\}^{m(|x|)}$ such that*

$$\Pr_{r \in \{0,1\}^{n(|x|)}}[V(x,w,r)=1] \geq 2/3.$$

Soundness: *For every $x \notin S$ and every $w \in \{0,1\}^{m(|x|)}$ it holds that*

$$\Pr_{r \in \{0,1\}^{n(|x|)}}[V(x,w,r)=1] \leq 1/3.$$

If the completeness condition holds with probability 1, then S is in $\mathcal{MA}_1$. The corresponding classes $\mathcal{MA}_2^0$ and $\mathcal{MA}_1^0$ are defined by requiring that the residual decision predicate $V_x(\cdot,\cdot) = V(x,\cdot,\cdot)$ can be computed by $\mathcal{AC}^0$ circuits.

Recall that any MA-proof system with two-sided error probability can be transformed into an MA-proof system with one-sided error probability (cf. [24] or [14, Exer. 9.8]).[27] *This transformation preserves the complexity of verification* (w.r.t $\mathcal{AC}^0$) *and increases the soundness error by at most a factor of n.*

Definition 7.2 ($\mathcal{AM}$ and $\mathcal{AM}^0$): *A set S is in $\mathcal{AM}_2$ if there exists a deterministic polynomial-time verification procedure V and two polynomials $n, m : \mathbb{N} \to \mathbb{N}$ such that the following two conditions hold:*

[27] The transformation involves prepending the prover's message with a sequence of n adequate n-bit strings ("shifts"), denoted $s_1, ..., s_n$, and the modified verification procedure accepts (on random-pad r) iff for some $i \in [n]$ it holds that $V(x, w, r \oplus s_i) = 1$.

Completeness: *For every $x \in S$ it holds that*

$$\Pr_{r\in\{0,1\}^{n(|x|)}}[\exists w\in\{0,1\}^{m(|x|)} \text{ s.t. } V(x,r,w)=1] \geq 2/3.$$

Soundness: *For every $x \notin S$ it holds that*

$$\Pr_{r\in\{0,1\}^{n(|x|)}}[\forall w\in\{0,1\}^{m(|x|)}\ V(x,r,w)=1] \leq 1/3.$$

If the completeness condition holds with probability 1, then S is in $\mathcal{AM}_1$. The corresponding classes $\mathcal{AM}_2^0$ and $\mathcal{AM}_1^0$ are defined by requiring that the residual decision predicate $V_x(\cdot,\cdot) = V(x,\cdot,\cdot)$ can be computed by $\mathcal{AC}^0$ circuits.

Note that AM-proof system with two-sided error probability can be transformed into an AM-proof system with one-sided error probability (cf. [24] or [14, Exer. 9.8]), but this transformation (see Footnote 27) involves turning an MAM-system into an AM-system (see [8] or [14, Apdx. F.2.2.1]), which does *not* preserve the soundness error sufficiently well for our purposes.

Our results. Considering quantified derandomization problems for the classes $\mathcal{MA}^0$ and $\mathcal{AM}^0$, our results present a dichotomy that is analogous to the one Theorem 1.3 and Theorem 1.4: While Theorem 7.3 shows that the $\mathcal{MA}^0$ systems with at most subexponential many exceptional random-pads collapse to $\mathcal{NP}$, Theorem 7.4 shows that an analogous result for $\mathcal{AM}^0$ would imply that $\mathcal{AM} = \mathcal{NP}$. The proof of the latter result uses the observation that $\mathcal{AM} = \mathcal{AM}^0$ (and furthermore that this holds while preserving the number of exceptional random-pads).

Theorem 7.3 (the case of $\mathcal{MA}^0$ and $B(n) = \exp(n^{1-\Omega(1)})$): *Suppose that S is in $\mathcal{MA}_2^0$ by virtue of a proof system that has error probability at most 2^{n^c-n}* (i.e., at most 2^{n^c} exceptional random-pads)*, for any constant $c < 1$. Then, $S \in \mathcal{NP}$.*

This raises the question of whether $\mathcal{MA} = \mathcal{MA}^0$ (and furthermore whether this holds while preserving the number of exceptional random-pads). A positive answer that also maintains a subexponential upper bound on the number of exceptional random-pads would imply $\mathcal{MA} = \mathcal{NP}$, because the exceptional random-pads in an MA-proof system can be reduced to 2^{n^c}, for any constant $c > 0$.

Proof: For any x, consider the residual $\mathcal{AC}^0$ circuit V_x. After the prover sent its message $w \in \{0,1\}^{m(|x|)}$, the verifier derives a circuit $C : \{0,1\}^{n(|x|)} \to \{0,1\}$ such that $C(r) = V_x(\beta, r)$. Applying Theorem 1.3 to the later circuit, the current theorem follows. ■

Theorem 7.4 (the case of $\mathcal{MA}^0$ and $B(n) = \exp(n^{\Omega(1)})$): *Suppose that for any S and any constant $c > 0$ such that S is in $\mathcal{AM}_1^0$ by virtue of a proof system that has error probability at most 2^{n^c-n}, it holds that $S \in \mathcal{NP}$. Then, $\mathcal{AM} = \mathcal{NP}$. Furthermore, the conclusion holds even if the hypothesis holds only for proof systems with a residual verification predicate that is a CNF.*

Proof: For any $S \in \mathcal{AM}$, we may assume w.l.o.g that $S \in \mathcal{AM}_1$. We first reduce the soundness error of the AM-proof system (for S), viewed as a function of its (new) randomness complexity. Specifically, applying the transformation that underlies the proof of Theorem 4.2 to the residual decision predicate, we can obtain an AM-proof system of randomness complexity n and soundness

error at most 2^{n^c-n}, for any $c > 0$ we desire. Indeed, for any $x \notin S$, the analysis distinguishes good (typical) random-pads r for which $\forall w \in \{0,1\}^{m(|x|}\ V(x,r,w)=0$ from bad (exceptional) r's for which this condition does not hold (i.e., $\exists w \in \{0,1\}^{m(|x|}$ s.t. $V(x,r,w)=1$).

We next note that every set in $\mathcal{AM}$ has an AM-proof system with a residual predicate that is a CNF. Furthermore, as shown next, this holds even while preserving the error probability of the proof system (as a function of n). Indeed, starting with the residual predicate $V_x(.,.)$, consider the residual predicate $V'_x(r,w'w) = V_x(w',w) \wedge (w'=r)$, which corresponds to the case that the prover prepends its message with a copy of the verifier's message (i.e., $w'=r$). Applying Cook's reduction to $V_x(w',w)$, while introducing auxiliary variables (which may be determined based only on $w'w$), we obtain the desired CNF. The key observation here is that we did not touch the sample space of the random-pads. ■

8 Discussion

The quantified derandomization challenge put forward in this paper has two parameters: (1) a class of circuits $\mathcal{C}$ (e.g., $\mathcal{AC}^0, \mathcal{AC}^0[2]$ or $\mathcal{P}/\mathrm{poly}$), and (2) a bounding function $B : \mathbb{N} \to \mathbb{N}$ (e.g., $B(n) = n^{\log n}$ or $B(n) = \exp(n^{0.99})$). Each such pair $(\mathcal{C}, B)$ yields a corresponding search problem in which one is given an n-input circuit $C \in \mathcal{C}$ that evaluates to 1 on all but at most $B(n)$ of its inputs, and is asked to find an input on which C evaluates to 1 (see Definition 1.1). The case of $B(n) = 2^{n-1}$ corresponds to the standard derandomization problem (of the one-sided or hitting type), whereas the case of $B(n) = \mathrm{poly}(n)$ is straightforward when allowing running time that is larger than $B(n)$. Hence, the new framework exhibit a spectrum of problems extending from standard derandomization problems to straightforward derandomization problems.

Furthermore, the quantified derandomization framework offers a tractable approach to unconditional derandomization results. This approach suggests making progress along a path that leads from the study of $(\mathcal{C}, B)$-search problems that do not imply unknown results regarding standard derandomization to the study of $(\mathcal{C}, B)$-search problems that do imply such results. We make first steps in this project by providing results for problems of the first type and by identifying problems of the second type.

In particular, our results indicate that, for the class $\mathcal{AC}^0[2]$ (and higher), the interesting but "non-spectacular" range for the function B is between super-polynomial and subexponential (i.e., $B(n) = \exp(n^c)$ for any constant $c \in (0,1)$). Actually, one may consider also a polynomial bounding function B, provided that one looks for algorithms of complexity below B. On the other extreme, recall that the $(\mathcal{AC}^0[2], B)$-search problem for subexponential B is not easier than the case of $B(n) = 2^{n-1}$ (i.e., standard derandomization for $\mathcal{AC}^0[2]$). Furthermore, even for $\mathcal{AC}^0$, the case of $B(n) = \exp(n/\mathrm{poly}(\log n))$ is not easier than $B(n) = 2^{n-1}$ (see Theorem 3.4).

Our main results "separate" $\mathcal{AC}^0$ from $\mathcal{AC}^0[2]$ in the sense that these two classes exhibit a different behavior w.r.t our derandomization challenge: On the one hand, Theorem 1.3 resolves this challenge for the class $\mathcal{AC}^0$ and every subexponential B. On the other hand, Theorem 1.4 asserts that resolving this challenge for the class $\mathcal{AC}^0[2]$ and any subexponential B is not easier than standard approximate counting for $\mathcal{AC}^0[2]$ itself. A similar dichotomy arises in the work of Agrawal *et al.* [1] w.r.t the existence of "Gap Theorems" regarding the power of reductions (i.e., $\mathcal{AC}^0$-reductions "collapse" to projections, whereas $\mathcal{AC}^0[2]$-reductions do not "collapse" to projections).

The first stage in the proof of our switching lemma (Lemma 3.3) bears some similarity to the

switching lemma proved by Ajtai and Wigderson [4], who were the first to use pseudorandom (rather than random) restrictions. While they used n^ϵ-wise independent restrictions, for any $\epsilon > 0$, we are using constant-wise independence. As noted above, we can afford this low amount of independence because (unlike prior studies of restrictions, including [4])[28] we do not care to preserve the acceptance probability of the circuit. We only need to keep alive (as undetermined by the restriction) a sufficient number of variables (i.e., more than $2 + \log_2 B(n)$).

Some of the questions raised by this work. The quantified derandomization problem raises a variety of natural questions. Some of these questions were raised explicitly in the previous sections.

1. *Derandomization implying circuit lower bounds*: For what functions B does a polynomial-time algorithm for the $(\mathcal{P}/\mathrm{poly}, B)$-search problem imply lower bounds? By Kabanets and Impagliazzo [23], this happens if $B(n) > exp(n^{\Omega(1)})$. Does it happen also if B is quasi-polynomial?

2. *What is the complexity of the $(\mathcal{AC}^0, B)$-search problem for $B(n) = \exp(n^{1-o(1)})$)?*

   Recall that for $B(n) = \exp(n^{0.999})$ the problem can be solved in (deterministic) polynomial-time, whereas a solution for the case of $B(n) = \exp(n^{1-O(\log\log n)})$ would imply full derandomization (see Theorem 3.4).

3. *What is the complexity of the $(\mathcal{AC}^0_{3,\mathrm{poly}}[2], B)$-search problem for $B(n) = \exp(n^{\Omega(1)})$) or even for quasi-polynomial B?* Indeed, this question refers to depth-three circuits with parity, and the challenge is extending the result of Theorem 1.3 to this case.

   Meeting this challenge is not known to imply a new full derandomization; partial results regarding this challenge appear in Section 6. The missing part seems to be the case that the circuit is of the form XOR-AND-XOR (see Case 4). This case can be solved if Theorem 1.6 is sufficiently improved (see Question 4)

4. Another subclass of $\mathcal{AC}^0[2]$ that is of interest consists of depth-five circuits *with parity gates only in the bottom*. Denoting this subclass by $\mathcal{C}$ and considering any subexponential B, we know that approximate counting for CNFs reduces to the $(\mathcal{C}, B)$-search problem (see Remark 4.4). On the other hand, the case of depth-three circuits with parity gates only in the bottom is easier than the case of $\mathcal{AC}^0$ (see Case 2 in Section 6). *What about the case of depth four*? Alternatively, what if B is quasi-polynomial?

5. *Can Theorem 1.6 be strengthened*? Try to present a deterministic $\mathrm{poly}(n)$-time algorithm that outputs a set of n-bit strings S_n such that for every d and every n-variate polynomial f

[28] In this sense the work of Trevisan and Xue [38] is a hybrid: They do present a pseudorandom restriction (albeit with polylogarithmic seed length), while only caring about the number of surviving variables (which is almost linear) [38, Lem.7], but in their main application they do care about preserving the acceptance probability of the circuit. So they use the restriction only to select "undetermined" variables, but do not determine the other variables according to the restriction (but rather use a random assignment to these variables as a mental experiment). In other words, they are using the restriction as a two-way partition of the variables (and they actually assign values to the *undetermined* variables according to some small bias probability space). We mention that pseudorandom restriction is generated by a distribution that fool CNFs of size that is larger than the size of the CNF that they hit with the restriction.

of degree d over GF(2) that evaluates to 0 on at most a $n \cdot 2^{-d}$ fraction of its domain, there exists $x \in S_n$ such that $f(x) = 1$.

Recall that Theorem 1.6 only deals with the case that the polynomial evaluates to 0 on an $O(2^{-d})$ fraction of its domain.

6. *Placing $\mathcal{BPP}$ with extremely few bad random inputs in $\mathcal{NP}$.* Suppose you are given a probabilistic polynomial-time algorithm that errs on at most $B(n)$ random inputs, where n denotes (as usual) the number of random inputs. Try to place the corresponding set in $\mathcal{NP}$, when B is quasi-polynomial.

   Recall that the case in which $B(n) = \exp(n^{\Omega(1)})$ is as hard as $\mathcal{BPP}$ itself. On the other hand, recall that $\mathcal{BPL}$ with a quasipolynomial B is in $\mathcal{L}$.

7. In the context of probabilistic proof systems many questions are begging.

   (a) In spirit of Question 1 (i.e., $(\mathcal{AC}^0, \exp(n^{1-o(1)})$-search problem), we ask about the complexity of $\mathcal{MA}^0$ with $B(n) = \exp(n^{1-o(1)})$. The point is using the power of nondeterminism in order to assist us here.

   (b) Can $\mathcal{MA}$ be related to $\mathcal{MA}^0$, possibly while preserving the value of the bounding function B? Recall that $\mathcal{AM} = \mathcal{AM}^0$ while preserving the value of B.

Acknowledgments

We are grateful to Mike Saks for sharing his result (i.e., Theorem A.1) with us and allowing us to include it in this write-up. We are also grateful to Emanuele Viola and David Zuckerman for useful discussions; in particular, our discussions with Emanuele led to Theorem 3.4. We thank Roei Tell for pointing out inaccuracies and unclarities in previous versions. O.G. was partially supported by the Minerva Foundation with funds from the Federal German Ministry for Education and Research.

References

[1] M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich. Reducing the complexity of reductions. *Computational Complexity*, Vol. 10 (2), pages 117–138, 2001. Preliminary version in *29th STOC*, 1997.

[2] M. Ajtai. Σ_1^1-formulae on finite structures. *Ann. Pure Appl. Logic*, Vol. 24 (1), pages 1–48, 1983.

[3] M. Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances in computational complexity theory* (New Brunswick, NJ, 1990), pages 1–20, AMS, 1993.

[4] M. Ajtai and A. Wigderson. Deterministic Simulation of Probabilistic Constant Depth Circuits. In *26th IEEE Symposium on Foundations of Computer Science*, pages 11-19, 1985.

[5] N. Alon, L. Babai and A. Itai. A Fast and Simple Randomized Algorithm for the Maximal Independent Set Problem. *J. of Algorithms*, Vol. 7, pages 567–583, 1986.

[6] N. Alon, O. Goldreich, J. Håstad, R. Peralta. Simple Constructions of Almost k-wise Independent Random Variables. *Journal of Random Structures and Algorithms*, Vol. 3, No. 3, pages 289–304, 1992. Preliminary version in *31st FOCS*, 1990.

[7] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, 2009.

[8] L. Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–429, 1985.

[9] P. Beame. A switching lemma primer. Technical Report UW-CSE-95-07-01, 1994. Available at `http://homes.cs.washington.edu/~beame/publications.html`

[10] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13 (4), pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

[11] B. Chor and O. Goldreich. On the Power of Two–Point Based Sampling. *Jour. of Complexity*, Vol 5, 1989, pages 96–106. Preliminary version dates 1985.

[12] A. De, O. Etesami, L. Trevisan, and M. Tulsiani. Improved Pseudorandom Generators for Depth 2 Circuits. In *14th RANDOM*, pages 504–517, 2010.

[13] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Velickovic. Efficient approximation of product distributions. *Random Structures and Algorithms*, Vol. 13 (1), pages 1–16, 1998.

[14] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[15] O. Goldreich. Three XOR-Lemmas - An Exposition. In *Studies in Complexity and Cryptography*, pages 248–272, 2011. Preliminary version in *ECCC*, TR95-056, 1995.

[16] O. Goldreich, S. Vadhan, and A. Wigderson. Simplified Derandomization of BPP Using a Hitting Set Generator. In *Studies in Complexity and Cryptography*, pages 59–67, 2011. Preliminary version in *ECCC*, TR00-004, 2000.

[17] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985. Earlier versions date to 1982.

[18] P. Gopalan, R. Meka, and O. Reingold. DNF sparsification and a faster deterministic counting algorithm. *Computational Complexity*, Vol. 22 (2), pages 275–310, 2013. Preliminary version in *27th CCC*, 2012.

[19] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. *ECCC*, TR12-123, Sept. 2012.

[20] J. Hastad. Almost Optimal Lower Bounds for Small Depth Circuits. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 143–170, 1989. Extended abstract in *18th STOC*, 1986.

[21] R. Impagliazzo, W. Matthews, and R. Paturi. A satisfiability algorithm for AC0. In *23rd SODA*, pages 961–972, 2012.

[22] S. Jukna. *Boolean Function Complexity: Advances and Frontiers.* Algorithms and Combinatorics, Vol. 27, Springer, 2012.

[23] V. Kabanets and R. Impagliazzo. Derandomizing Polynomial Identity Tests means proving circuit lower bounds. *Computational Complexity*, Vol. 13 (1–2), pages 1–46, 2004.

[24] C. Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, Vol. 17, pages 215–217, 1983.

[25] J. Naor and M. Naor. Small-bias Probability Spaces: Efficient Constructions and Applications. *SIAM Journal on Computing*, Vol 22, 1993, pages 838–856. Preliminary version in *22nd STOC*, 1990.

[26] N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, Vol. 11 (1), pages 63–70, 1991.

[27] N. Nisan. Pseudorandom Generators for Space Bounded Computation. *Combinatorica*, Vol. 12 (4), pages 449–461, 1992. Preliminary version in *22nd STOC*, 1990.

[28] N. Nisan. $\mathcal{RL} \subseteq \mathcal{SC}$. *Computational Complexity*, Vol. 4, pages 1-11, 1994. Preliminary version in *24th STOC*, 1992.

[29] N. Nisan and A. Wigderson. Hardness vs Randomness. *Journal of Computer and System Science*, Vol. 49, No. 2, pages 149–167, 1994. Preliminary version in *29th FOCS*, 1988.

[30] N. Nisan and D. Zuckerman. Randomness is Linear in Space. *Journal of Computer and System Science*, Vol. 52 (1), pages 43–52, 1996. Preliminary version in *25th STOC*, 1993.

[31] O. Reingold, T. Steinke, and S. Vadhan. Pseudorandomness for Regular Branching Programs via Fourier Analysis. In *17th RANDOM*, pages 655–670, 2013.

[32] M. Sipser. Expanders, Randomness, or Time versus Space. *Journal of Computer and System Science*, Vol. 36 (3), pages 379–383, 1988. Preliminary version in *Structure in Complexity Theory Conference*, 1986.

[33] R. Shaltiel. Recent Developments in Explicit Constructions of Extractors. In *Current Trends in Theoretical Computer Science: The Challenge of the New Century*, *Vol 1: Algorithms and Complexity*, World scietific, 2004. (Editors: G. Paun, G. Rozenberg and A. Salomaa.) Preliminary version in *Bulletin of the EATCS 77*, pages 67–95, 2002.

[34] E. Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, Vol. 13 (3-4), pages 147–188, 2005. Preliminary version in *18th CCC*, 2003.

[35] E. Viola. On Approximate Majority and Probabilistic Time. *Computational Complexity*, Vol. 18 (3), pages 337–375, 2009. Preliminary version in *22nd CCC*, 2007.

[36] E. Viola. The Sum of D Small-Bias Generators Fools Polynomials of Degree D. *Computational Complexity*, Vol. 18 (2), pages 209–217, 2009. Preliminary version in *23rd CCC*, 2008.

[37] L. Trevisan. Extractors and Pseudorandom Generators. *Journal of the ACM*, Vol. 48 (4), pages 860–879, 2001. Preliminary version in *31st STOC*, 1999.

[38] L. Trevisan and T. Xue. A Derandomized Switching Lemma and an Improved Derandomization of AC0. *ECCC*, TR12-116, Sept. 2012.

[39] A.C. Yao. Theory and Applications of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[40] A.C. Yao. Separating the Polynomial-Time Hierarchy by Oracles. In *26th IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.

[41] D. Zuckerman. Randomness-Optimal Oblivious Sampling. *Random Structures and Algorithms*, Vol. 11, Nr. 4, December 1997, pages 345–367. Preliminary version in *28th STOC*, pages 286–295, 1996.

Appendix A: Logarithmic space and $B(n) = 2^{0.999n}$

Following the initial posting of this work, we learned that Mike Saks obtained in the 1990s a much stronger result than the one stated in Proposition 1.2. In particular, he obtained the following result (but never published it).

Theorem A.1 (log-space and $B(n) = 2^{cn}$ for every $c < 1$): *Suppose that for some $c < 1$, the set S is decidable by a probabilistic log-space algorithm that, on any n-bit long input, errs only on at most $2^{c \cdot r(n)}$ sequences of the possible $r(n)$-bit random outcomes. Then, S is in $\mathcal{L}$.*

Proof: Consider the (log-space uniform) ordered (read-once) branching program of polynomial in n width that corresponds to the log-space computations, and assume without loss of generality that its length, denoted $r = r(n)$, equals its width.

Let $t = O(\log r)$, where the constant will be determined later. For $i = 0, ..., r/t$, consider the vertices in layer $i \cdot t$ of this branching program, and let v_0 denote the source vertex (which resides in the zero layer). For $i = 1, ..., r/t$, denote by v_i the vertex that is reached with the highest probability when taking a random directed walk of length t from v_{i-1}, and breaking ties arbitrarily. Indeed, v_i is reached by at least $2^t/r$ of the directed paths of length t that start at v_{i-1}. We make the following observations:

1. One can find $v_{r/t}$ in logarithmic space, by iteratively finding v_i when given v_{i-1}, where the key fact is that the search space is the set of strings of length $t = O(\log n)$.

2. The number of directed paths from v_0 to $v_{r/t}$ is at least $M \stackrel{\text{def}}{=} (2^t/r)^{r/t} = 2^{(1-(\log_2 r)/t) \cdot r}$, where the key fact is that the width of the program is r. For $t = (\log_2 r)/(1-c)$, it holds that $M = 2^{cr}$.

Hence, if $B(n) < 2^{c \cdot r}$, then $v_{r/t}$ must encode the correct verdict regarding the input. The theorem follows. ■

Improving over Theorem A.1. It is tempting to think that one can improve over Theorem A.1 by setting $t = \text{poly}(\log n)$ and using the Nisan-Zuckerman pseudorandom generator [30] for finding a popular v_i (i.e., one that is reached by many of the directed paths from v_{i-1}). Unfortunately, the error probability of this generator is too high for this application (e.g., it definitely exceeds $1/n$). than the one stated in Proposition 1.2. This leads to the following open problem.

Open Problem A.2 (log-space and $B(n) = 2^{n-o(n)}$): *Suppose that for some sub-linear function f, the set S is decidable by a probabilistic log-space algorithm that, on any n-bit long input, errs only on at most $2^{r(n)-f(r(n))}$ sequences of the possible $r(n)$-bit random outcomes. Can S be placed in $\mathcal{L}$?*

Appendix B: Self-Contained Proof of Lemma 5.2

Recall that Lemma 5.2 asserts that *if p is a degree $d+1$ polynomial over* GF(2) *with bias at least $1 - \delta \geq 1/2$ and W ϵ-fools every degree d polynomial that has bias at least $1 - 2\delta$, then W $(\epsilon/(1-\delta))$-fools p.*

Proof: The proof follows Viola's proof of [36, Lem. 4], while adding an analysis of the bias of the polynomials that he uses. Let U and U' be two independent random variables, each uniformly distributed in $\{0,1\}^n$. Then

$$\begin{aligned}
&\left|\mathrm{E}[(-1)^{p(W)}] - \mathrm{E}[(-1)^{p(U)}]\right| \cdot \left|\mathrm{E}[(-1)^{p(U')}]\right| && (2)\\
&= \left|\mathrm{E}[(-1)^{p(W)+p(U')}] - \mathrm{E}[(-1)^{p(U)+p(U')}]\right| && (3)\\
&= \left|\mathrm{E}[(-1)^{p(W)+p(W+U')}] - \mathrm{E}[(-1)^{p(U)+p(U+U')}]\right|. && (4)
\end{aligned}$$

For every $z \in \{0,1\}^n$, define $p'_z(x) = p(x+z) + p(x)$, and note that p'_z has degree at most d (since the degree $d+1$ terms cancel out). Then, Eq. (4) can be written as $|\mathrm{E}[(-1)^{p'_{U'}(W)}] - \mathrm{E}[(-1)^{p'_{U'}(U)}]|$, which is upper bounded by $\max_z\{|\mathrm{E}[(-1)^{p'_z(W)}] - \mathrm{E}[(-1)^{p'_z(U)}]|\}$, which in turn is the amount by which W fools p'_z, denoted $\mathtt{fool}(W, p'_z)$. On the other hand, Eq. (2) represents the multiple of the amount by which W fools p, denoted $\mathtt{fool}(W,p)$, and the bias of p, denoted $\mathtt{bias}(p)$. Hence, we get

$$\mathtt{fool}(W,p) \leq \frac{\max_z\{\mathtt{fool}(W,p'_z)\}}{\mathtt{bias}(p)}. \qquad (5)$$

Next, we note that the bias of each p'_z is at least $1-2\delta$:

$$\begin{aligned}
\left|\mathrm{E}\left[(-1)^{p'_z(U)}\right]\right| &= \left|\mathrm{E}\left[(-1)^{p(U+z)+p(U)}\right]\right| \\
&= |1 - 2\cdot \Pr\left[p(U+z)+p(U)=1\right]| \\
&= |1 - 2\cdot \Pr[p(U+z) \neq p(U)]| \\
&\geq 1 - 4\cdot \Pr[p(U) = b_{\min}]
\end{aligned}$$

where $b_{\min} \in \{0,1\}$ is such that $\Pr[p(U) = b_{\min}] \leq 1/2$, and the inequality is proved as follows. First note that $\Pr[p(U+z) \neq p(U)] \leq \Pr[P(U+z) = b_{\min} \vee p(U) = b_{\min}]$, which in turn is upper bounded by $\Pr[P(U+z) = b_{\min}] + \Pr[p(U) = b_{\min}] = 2 \cdot \Pr[p(U) = b_{\min}]$. Using $\Pr[p(U) = b_{\min}] = (1 - |\mathrm{E}[(-1)^{p(U)}]|)/2 \leq \delta/2 \leq 1/4$, we get $|\mathrm{E}[(-1)^{p'_z(U)}]| = 1 - 2\cdot\Pr[p(U+z) \neq p(U)] \geq 1 - 4\cdot\delta/2$.

Having established that each p'_z has degree (at most) d and bias at least $1-2\delta$, and using the hypothesis that W ϵ-fools such polynomials, we get $\max_z\{\mathtt{fool}(W,p'_z)\} \leq \epsilon$. Plugging this (and $\mathtt{bias}(p) \geq 1-\delta$) into Eq. (5), we get $\mathtt{fool}(W,p) \leq \epsilon/(1-\delta)$, and the lemma follows. ∎