Electronic Colloquium on Computational Complexity, Report No. 120 (2021)

ECCC

How to Find Water in the Ocean

A Survey on Quantified Derandomization

Roei Tell *

August 17, 2021

Abstract

The focus of this survey is the question of quantified derandomization, which was introduced by Goldreich and Wigderson (2014): Does derandomization of probabilistic algorithms become easier if we only want to derandomize algorithms that err with extremely small probability? How small does this probability need to be in order for the problem's complexity to be affected?

This question opens the door to studying natural relaxed versions of the derandomization problem, and allows us to construct algorithms that are more efficient than in the general case as well as to make gradual progress towards solving the general case. In the survey I describe the body of knowledge accumulated since the question's introduction, focusing on the following directions and results:

1. **Hardness vs "quantified" randomness:** Assuming sufficiently strong circuit lower bounds, we can derandomize probabilistic algorithms that err extremely rarely while incurring *essentially no time overhead*.
2. For general probabilistic polynomial-time algorithms, **improving on the brute-force algorithm for quantified derandomization implies breakthrough circuit lower bounds**, and this statement holds for *any given probability of error*.
3. Unconditional **algorithms for quantified derandomization of low-depth circuits and formulas,** as well as *near-matching reductions* of the general derandomization problem to quantified derandomization for such models.
4. **Arithmetic quantified derandomization**, and in particular constructions of hitting-set generators for polynomials that vanish extremely rarely.
5. **Limitations of certain black-box techniques** in quantified derandomization, as well as a tight connection between black-box quantified derandomization and the classic notion of **pseudoentropy**.

Most of the results in the survey are from known works, but several results are either new or are strengthenings of known results. The survey also offers a host of concrete challenges and open questions surrounding quantified derandomization.

*Massachusetts Institute of Technology, Cambridge, MA. Email: roei.tell@gmail.com

ISSN 1433-8092

Contents

1 Introduction

> *Does derandomization of probabilistic algorithms become easier when the number of "bad" random inputs is extremely small? (Goldreich and Wigderson [GW14].)*

The context for this survey is the question of derandomization: Can we simulate randomness in a deterministic and efficient way? More accurately, we ask *which types* of randomized algorithms can be simulated in a deterministic way, and what is the precise cost of simulation. The main focus in this study is on simulating randomized algorithms that solve *decision problems*, which is the $\mathcal{BPP}$ vs $\mathcal{P}$ question.[1] As we can expect of one the main questions in complexity theory, progress on it has been challenging, and we know that essentially any progress on this question is closely related to progress on other central questions in complexity theory.

The textbook definition of *probabilistically solving a decision problem* $L \subseteq \{0,1\}^*$, which underlies the definition of $\mathcal{BPP}$, considers a randomized algorithm to be successful if it errs with probability at most $1/3$ on every fixed input; that is, the fraction of random strings that cause the algorithm to err is at most $1/3$.

This survey is concerned with the seemingly innocent choice of error bound $1/3$. Going back to the original definition of $\mathcal{BPP}$ in [Gil74], the class was defined with an unspecified error bound that can be any constant smaller than $1/2$, such as .49. On the other hand, when we present this topic to non-expert audiences, we sometimes choose a miniscule constant such as 10^{-10} for dramatic effect. Of course, both formulations are essentially equivalent, since we can apply *error reduction* to efficiently reduce the error from $1/2 - n^{-O(1)}$ to $2^{-\mathrm{poly}(n)}$ with only a polynomial runtime overhead.

Therefore, a common sentiment is that the precise choice of error bound doesn't really matter, as long as it is noticeably smaller than $1/2$. [2] But is this sentiment accurate even when we take a sub-constant error bound very close to zero, focusing on algorithms that only err extremely rarely? It turns out that in this setting, *the precise choice of error bound matters a lot*. In fact, the problem is so sensitive to this choice that even tiny changes in the error bound mark the difference between settings in which efficient derandomization is known, and settings in which showing even mild derandomization would yield dramatic consequences in complexity theory.

1.1 The general question

Let's start with a trivial extreme point: If we define a probabilistic algorithm to be successful only if it never errs – that is, we set the error bound in the definition of $\mathcal{BPP}$ to be zero – then we just defined *deterministic* computation in a cumbersome way;

[1] As usual, this focus is taken merely for simplicity, and there is an efficient search-to-decision reduction in this setting (i.e., search problems that can be efficiently solved by probabilistic algorithms, and for which solutions can be efficiently verified, reduce to *promise-$\mathcal{BPP}$*; see [Gol11, Theorem 3.5]).

[2] Allowing error that is arbitrarily close to $1/2$ is a different story. Such a choice is less natural (since we are defining a negligible improvement over a random coin toss as "successfully solving a problem") and yields the very large complexity class $\mathcal{PP}$ (recall that $\mathcal{P}^{\mathcal{PP}} = \mathcal{P}^{\#\mathcal{P}} \supseteq \mathcal{PH}$, using [Tod91]).

needless to say, derandomization becomes trivial in this case. But what if we allow the randomized algorithm to err on *just a single random string*, out of the exponentially many possible choices for random strings? What if we allow it to err on polynomially many strings? Where is the threshold at which the derandomization problem stops being trivial, and what happens beyond this threshold?

Several years ago Goldreich and Wigderson [GW14] asked these questions in a broad and methodical way, leading to a fruitful study of what they called quantified derandomization: This is the question of *derandomizing algorithms that err extremely rarely*, where "extremely rarely" here refers to the number of random strings that cause the probabilistic algorithm to err. As they mention in their work, an early form of this question was already considered long ago by Sipser [Sip86], who considered the class "strong $\mathcal{R}$" of problems solvable with extremely small one-sided error.

Let us define the notion of probabilistically solving a decision problem with error bound B, where the parameter B will quantify the number of exceptional random strings (i.e., random strings that cause the algorithm to err). We will measure B as a function of the number of random coins (rather than of the input length), since we are interested in comparing the number of exceptional random strings to the total number of choices for a random string. For simplicity of presentation, let us assume for the moment that the number of random coins equals the running time. (We will get rid of this simplifying assumption later on in Section 2.3.)

Definition 1.1 (probabilistically solving a decision problem with error bound B)**.** *For $B\colon \mathbb{N} \to \mathbb{N}$, we say that $\Pi = (\mathsf{Y}, \mathsf{N}) \subseteq \{0,1\}^* \times \{0,1\}^*$ is in $pr\mathcal{BPTIME}_B[T]$ if there exists a randomized algorithm that gets input $x \in \{0,1\}^*$, runs in time $T = T(|x|)$, and:*

1. *If $x \in \mathsf{Y}$, the algorithm* accepts *given all but at most $B(T)$ choices of random strings.*
2. *If $x \in \mathsf{N}$, the algorithm* rejects *given all but at most $B(T)$ choices of random strings.*

I stress again that that $B(T)$ is the *absolute number* of exceptional random strings in Definition 1.1, rather than their fraction. Thus, and since we assumed (for now) that the number of random coins equals the running time T, the error probability of the algorithm in Definition 1.1 is $B(T)/2^T$. Indeed, the standard definition of $pr\mathcal{BPTIME}[T]$ is the special case obtained by using $B(T) = 2^T/3$.

Trying to derandomize only algorithms that err extremely rarely makes the challenge potentially easier; that is, Definition 1.1 opens the door for a *relaxation of the classical derandomization problem*. However, this relaxation entirely hinges on the choice of function B: For small values of B (e.g., for $B(T)) = O(1)$) the corresponding derandomization problem is easy, since we can just use the brute-force deterministic simulation that runs the original algorithm using $2B(T) + 1$ fixed choices of a random string; whereas for larger values of B (e.g., for $B(T) = \Omega(2^T)$) the derandomization problem is as challenging as the original and general derandomization problem.

1.2 The role of error-reduction

As mentioned above, we can efficiently *reduce the error* of a probabilistic algorithm. The naive way to do so is to repeat an algorithm that has error $1/3$ for k times and output the majority decision, which reduces its error to $2^{-\Omega(k)}$. This naive method reduces B only mildly as a function of the number of random coins, and using more sophisticated tools we can reduce B to be (say) subexponential in the number of random coins at a relatively low computational cost (see Section 3.3 for details).[3] This means that, in high-level, *general derandomization reduces to quantified derandomization* with relatively small values of B and with a corresponding computational overhead.

The point is that, in contrast to a common mistaken intuition, this does not trivialize the question of quantified derandomization, but rather (to the contrary) highlights its importance. Specifically, this suggests a natural approach to solve the general derandomization problem: First reduce general derandomization to quantified derandomization (e.g., by error-reduction), and then solve the corresponding quantified derandomization problem. Indeed, when taking this approach what we are actually asking is whether we can *reduce general derandomization to a target setting of quantified derandomization that we can efficiently solve*. This calls for developing efficient algorithms for quantified derandomization, as well as efficient approaches for error-reduction.[4] We will see both types of results in this survey.

1.3 Additional motivation

Derandomizing algorithms that err extremely rarely is, in my view, a natural problem that is inherently interesting, and therefore it does not need additional external motivations. (Indeed, recall that the problem was considered as early as 1986 [Sip86].) For example, one may ask what is the precise time complexity of derandomizing algorithms that err extremely rarely, or which assumptions are sufficient and necessary in order to do so (as we will see, both questions have recently been studied).

Nevertheless, let me mention two additional motivations for studying quantified derandomization, where both of them view this question as a *stepping-stone towards solving the general derandomization problem*. The first additional motivation is that, as explained in Section 1.2, a natural approach to solve the general derandomization problem is to reduce it to quantified derandomization and then solve the latter.

The second additional motivation is more generic: Since quantified derandomization is a relaxation of general derandomization, we a-priori hope that studying the former will shed light on the latter and pave the way for gradual progress towards

[3] To be more precise, let us measure B as a function of the number of random coins R. Naive error-reduction only yields $B(R) = 2^{(1-o(1))\cdot R}$, since repeating an algorithm with $r = \omega(1)$ coins for k times yields an algorithm with $R = k \cdot r$ coins and error probability $2^{-\Omega(k)} = 2^{-\Omega(R/r)}$.

[4] In general, applying standard black-box techniques for error-reduction and then the brute-force algorithm for quantified derandomization can never yield a non-trivial algorithm for general derandomization (see Appendix A). Thus, when using this approach, we need either a better-than-brute-force algorithm for quantified derandomization, or a non-standard technique for error-reduction.

solving it. It turns out that this generic motivation materialized in a fruitful way in the case of quantified derandomization: The *results* that we will see are surprising, rely on new techniques, and point both at specific technical challenges that create bottlenecks and at connections between quantified derandomization and well-known questions in complexity theory (e.g., circuit lower bounds and pseudoentropy).

Lastly, as pointed out by Avi Wigderson, the study of quantified derandomization led to constructions of important pseudorandom objects. For example, Sipser's [Sip86] original work was one of the driving forces behind the study of explicit randomness extractors (see, e.g., [CW89, Acknowledgements]). Analogously, the recent introduction of quantified derandomization in [GW14] led to constructions of pseudorandom restriction algorithms for weak circuit classes, and to constructions of extractors that are computable in weak circuit classes (see, e.g., Appendices B and C, respectively).

1.4 Organization

An overview of the results that are included in this survey is presented in Section 2. After stating preliminary definitions in Section 3, the subsequent Sections 4, 5, 6, 7, 8, and 9 expand on each of the subsections of Section 2, respectively, elaborating on the high-level results with more technical details and explanations. A reader interested in open problems in quantified derandomization will find numerous ones in Section 10.

Appendix A expands on Footnote (4) above. Appendices B and C describe technical constructions that underlie some of the results described in Section 2. Finally, Appendix D surveys two additional settings for quantified derandomization that have been explored relatively less so far.

2 An overview: What do we know?

This is the main section of this survey, and it presents a high-level overview of the results that are included in the text. Some results will be stated a bit concisely or informally, and in the subsequent technical sections I'll state them formally and with additional explanations, details, and sometimes with proofs or proof sketches.

2.1 Non-uniform quantified derandomization with no overhead

As in any derandomization problem, a natural starting point is asking what we can do *non-explicitly*; that is, by asking for an unconditional non-uniform quantified derandomization. Recall that for general derandomization, Adleman's theorem [Adl78] asserts that $prBPTIME[T] \subset prSIZE[n \cdot \tilde{O}(T)]$; hence, the known non-uniform general derandomization incurs a multplicative size overhead proportional to the input size n. However, for quantified derandomization this can be improved:

Theorem 2.1 (non-uniform quantified derandomization)**.** *For any $T(n) \geq n$ and any $B(T) = 2^{(1-\epsilon)\cdot T}$ where $\epsilon > 0$ is a constant we have that $prBPTIME_B[T] \subset prSIZE[\tilde{O}(T)]$.*

Proof. As in Adleman's proof, given a probabilistic T-time algorithm, we first reduce its error below 2^{-n} and then non-uniformly hard-wire a fixed choice of a random string that yields a correct decision on all 2^n inputs. The difference here is that the initial error probability isn't $1/3$ but rather $B(T)/2^T = 2^{-\epsilon \cdot T} \leq 2^{-\epsilon \cdot n}$, so the error reduction step will incur less time overhead. Specifically, after only constantly many repetitions the error decreases below 2^{-n} while incurring only a constant multiplicative runtime overhead,[5] at which point we can hard-wire a fixed random string. ■

Thus, when derandomizing algorithms that err on at most $B(T) = 2^{(1-\Omega(1)) \cdot T}$ random strings we can *avoid the multiplicative derandomization overhead of n* that occurs for general derandomization. This difference is particularly striking in the case of probabilistic linear-time algorithms, where general derandomization can be done by circuits of quadratic size, but quantified derandomization with $B(T) = 2^{(1-\Omega(1)) \cdot T}$ can be done by circuits of near-linear size. The meaning of this result is that already for "slightly non-trivial" values of B, the relaxation of considering quantified derandomization might allow for more efficient derandomization algorithms.

2.2 Hardness vs quantified randomness

Can we get *uniform* quantified derandomization with almost no time overhead under reasonable assumptions? The high-level answer is yes: The classical "hardness versus randomness" framework extends to the setting of very fast quantified derandomization, allowing to get uniform derandomization with parameters that are close to the ones in Theorem 2.1 assuming strong circuit lower bounds.

Specifically, the results below deduce quantified derandomization that avoids a multiplicative overhead of n, at the cost of assuming lower bounds for $\mathcal{SVN}$ circuits rather than for standard circuits as in classical hardness to randomness results. Recall that $\mathcal{SVN}$ circuits are the non-uniform analogue of $\mathcal{NP} \cap co\mathcal{NP}$ (see Definition 5.1), and therefore a lower bound for such circuits is a stronger hardness assumption than a lower bound for standard circuits. The first result yielding fast quantified derandomization from lower bounds for $\mathcal{SVN}$ circuits was proved by Doron *et al.* [DMO+20]:

Theorem 2.2 (hardness to quantified randomness with a near-quadratic overhead [DMO+20])**.** *For any $\epsilon > 0$, assume that there exists $L \in \mathcal{DTIME}[2^n]$ that cannot be computed by $\mathcal{SVN}$ circuits of size $2^{(1-\epsilon) \cdot n}$ for all sufficiently large $n \in \mathbb{N}$. Then, for $B(T) = 2^{T^{1-O(\epsilon)}}$ we have*

$$pr\mathcal{BPTIME}_B[T] \subseteq pr\mathcal{DTIME}[T^{2+O(\epsilon)}] \,.$$

The near-quadratic time overhead in Theorem 2.2 can be separated into two different overheads: A near-quadratic "preprocessing" overhead, which can be computed once per input length n and yields information that will be used to derandomize all

[5] Denoting $p = 2^{-\epsilon \cdot n}$, when we repeat the algorithm $k = O_\epsilon(1)$ times, the probability that it errs at least $k/3$ times is at most $\sum_{i=k/3}^{k} \binom{k}{i} p^i (1-p)^{k-i} < k \cdot 2^k \cdot p^{k/3} < 2^{-n}$.

T-time algorithms on inputs of length n, and a smaller *near-linear overhead* that is paid later on when derandomizing each particular algorithm (see Section 5 for details).

Chen and the current author [CT21b] showed that a stronger hypothesis yields a stronger conclusion: Assuming that the entire truth-table of the hard problem above can be printed "in a batch" in time $2^{(1+\epsilon)\cdot n}$ (rather than paying time 2^n for each of the 2^n inputs), we deduce quantified derandomization with a near-linear overhead:

Theorem 2.3 (hardness to quantified randomness with a near-linear overhead [CT21b])**.** *For any $\epsilon > 0$, assume that there exists $L \subseteq \{0,1\}^*$ such that for every $n \in \mathbb{N}$ the truth-table of L can be printed in time $2^{(1+\epsilon)\cdot n}$, but L cannot be computed by $\mathcal{SVN}$ circuits of size $2^{(1-\epsilon/4)\cdot n}$. Then, for $B(T) = 2^{T^{1-\epsilon}}$ we have that*

$$prBPTIME_B[T] \subseteq prDTIME[T^{1+O(\epsilon)}] .$$

The time overhead in Theorem 2.3 almost matches the non-uniform derandomization in Theorem 2.1 (a more refined and parameterized bound appears in Theorem 5.3). However, the derandomizations both in Theorem 2.2 and in Theorem 2.3 hold only for $B(T) = 2^{T^{1-\epsilon}}$ rather than for $B(T) = 2^{(1-\epsilon)\cdot T}$ as in Theorem 2.1.

Black-box quantified derandomization necessitates lower bounds for $\mathcal{E}$. The quantified derandomization algorithms underlying Theorems 2.2 and 2.3 are proved via the standard black-box approach of constructing PRGs.[6] The following result asserts that such black-box algorithms *necessitate* circuit lower bounds for $\mathcal{E}$. Let me state a nice special case of this assertion, deferring the general statement to the technical section:

Theorem 2.4 (black-box quantified derandomization implies circuit lower bounds)**.** *For any $\epsilon > 0$ and $s = s(n)$, assume that there exists a hitting-set generator for n-bit circuits of size $s(n)$ that reject at most $B(n) = 2^{(1-\epsilon)\cdot n}$ exceptional inputs, whose seed length is $\ell(n) \leq (1-\epsilon)\cdot n$ and that is computable in time $2^{O(\ell)}$. Then, $\mathcal{E} \not\subset \mathcal{SIZE}[s]$.*

The seed length in Theorem 2.4 is just shy of the trivial one, since there is a trivial HSG with seed length $\log(B(n)) + 1$.[7] In contrast, the PRGs underlying Theorems 2.2 and 2.3 have a much shorter seed, of length $\approx \epsilon \cdot \log(n)$. The meaning is that, on the one hand, essentially any non-trivial black-box quantified derandomization necessitates showing that $\mathcal{E}$ is hard for standard circuits; but on the other hand, we only know how to get black-box quantified derandomization that has *almost no time overhead* under the assumption that $\mathcal{E}$ is hard for $\mathcal{SVN}$ circuits. (see Open Problem 1).

[6] These PRGs only fool extremely biased distinguishers, and are thus suitable for quantified derandomization rather than for general derandomization; see Section 3.2 for definitions and details.

[7] The general form of Theorem 2.4 holds whenever the seed length of the PRG is at most $\log(B(n))$, and trades off the seed length for the computational complexity of the hard function (see Theorem 5.4). In contrast to the analogous result for general HSGs, we do not know how to trade off the seed length for the *size* of the circuits for which we deduce lower bounds (see Remark 5.5).

2.3 Quantified derandomization of specific circuit classes

What can we do unconditionally, without relying on hardness assumptions? For general probabilistic algorithms, we currently do not have any quantified derandomization algorithm that outperforms the naive brute-force algorithm (i.e., the algorithm the enumerates over $2B(T)+1$ fixed choices for a random string). In fact, essentially any improvement over the brute-force algorithm for general circuits would yield breakthrough circuit lower bounds, such as $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$; see Theorem 2.6 below.

However, for restricted classes of probabilistic algorithms we have a variety of nontrivial algorithms for quantified derandomization, which significantly outperform the brute-force algorithm. Moreover, we also know that improving the known algorithms would yield new lower bounds for corresponding classes of circuits. To go into more detail, I first define a complete problem for $prBPTIME_B$, then I state the result that improving the brute-force algorithm for general circuits implies circuit lower bounds, and then I'll describe the known results for several restricted classes of algorithms.

2.3.1 A complete problem for quantified derandomization

Recall the standard complete problem for $prBPP$, which is called the Circuit Acceptance Probability Problem (CAPP): In CAPP we are given as input a Boolean circuit that has n input bits and a single output bit, and we need to decide whether the circuit accepts all but at most $2^n/3$ of its input strings or rejects all but at most $2^n/3$ of its input strings. The complete problem for $prBPTIME_B$ generalizes CAPP using the parameter B that quantifies the number of exceptional inputs.

Definition 2.5 (quantified derandomization)**.** *The* Quantified Derandomization problem with error bound B (QD_B, *in short) is the following promise problem:*

1. *The set of "yes" instances* $\mathsf{Y} \subseteq \{0,1\}^*$ *consists of descriptions of* n*-bit circuits that* accept all but $B(n)$ *of their input strings.*
2. *The set of "no" instances* $\mathsf{N} \subseteq \{0,1\}^*$ *consists of descriptions of* n*-bit circuits that* reject all but $B(n)$ *of their input strings.*

When the given circuit is also promised to belong to a certain restricted class of circuits denoted by $\mathcal{C}$*, we denote the problem by* $\mathsf{QD}_B[\mathcal{C}]$.

Indeed, CAPP is the special case of QD_B with $B(n) = 2^n/3$, and for any B it holds that QD_B is complete for $prBPTIME_B$. In fact, at this point we can also remove the simplifying assumption that the number of random coins equals the running time: Consider derandomizing algorithms that get n-bit inputs, run in time $T = T(n) \geq n$, use $R = R(n)$ random coins, and have at most $B(R)$ exceptional inputs; this problem reduces to solving QD_B for R-bit circuits of size $\tilde{O}(T)$ (see Theorem 3.2).[8]

[8]Similarly to the convention when defining CAPP, in Definition 2.5 we denote the number of input bits to the circuit by n. This is a non-obvious choice, since the total input length to QD_B is not n but rather the description length of the circuit, which may be larger than n.

2.3.2 Beating the brute-force algorithm for general circuits implies circuit lower bounds

Let us begin with the setting of solving QD_B for general circuits (i.e., circuits with gates of fan-in two over the De Morgan basis, with no restriction on depth or on fan-out). For context, recall that no better-than-brute-force algorithms for CAPP of general circuits are known, and that the celebrated result of Williams [Wil13] asserts that solving CAPP for n-bit circuits of polynomial size, even just in "slightly non-trivial" time $2^n/n^{\omega(1)}$, already implies that $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$.

Williams' result can be interpreted as saying that for $B = B(n) = 2^n/3$, solving QD_B faster than the brute force algorithm that enumerates over $2B+1 = \Theta(2^n)$ inputs implies that $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$. Specifically, the algorithm is required to run in time $2^n/n^{\omega(1)} = B \cdot (\log B)^{-\omega(1)}$. The following result asserts that *the result of [Wil13] generalizes to all possible values of* $B(n)$; that is, if there *exists* some $B = B(n)$ such that we can solve QD_B in time $B \cdot (\log B)^{-\omega(1)}$, then $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$.

Theorem 2.6 (beating brute-force quantified derandomization implies circuit lower bounds; informal, see Theorem 6.1)**.** *Suppose that for some* $B(n) < 2^n$ *there exists a deterministic algorithm that solves* QD_B *for* n*-bit circuits of size* $\text{poly}(n)$ *in time* $B(n) \cdot (\log(B(n)))^{-\omega(1)}$*. Then,* $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$*.*

The meaning of Theorem 2.6 is that essentially any improvement over the brute-force algorithm for QD_B of general circuits, for any value of $B = B(n)$, would imply breakthrough circuit lower bounds.[9] Indeed, in this result the better-than-brute-force algorithm does not necessarily need to be a black-box algorithm as in Theorem 2.4, but the implied lower bound is not as strong as in Theorem 2.4 (i.e., the lower bound is for $\mathcal{NEXP}$ rather than for $\mathcal{E}$). In addition, taking a broader perspective than quantified derandomization per-se, Theorem 2.6 can be viewed as shedding additional light on the challenge in circuit-analysis that was pointed out in [Wil13].

As in other results using the algorithmic method for circuit lower bounds, the hypothesis of Theorem 2.6 can be relaxed, and only require a non-deterministic algorithm that accepts circuits that accept all the inputs, and rejects circuits that reject all but at most $B(n)$ of their inputs (see Theorem 6.1). On the other hand, using Murray and Williams' [MW18] extension of [Wil13], a stronger hypothesis in Theorem 2.6 yields a stronger lower bound: If the algorithm solves QD_B for circuits of polynomial size with $B(n) = \text{poly}(n)$ exceptional inputs in time $B(n)^{1-\epsilon}$ (e.g., the given circuit is of size n^2, the number of exceptional inputs is n^{100}, and the required running time is $n^{.99}$), then $\mathcal{NP}$ does not have circuits of size n^k, for every $k \in \mathbb{N}$ (see Theorem 6.2).

[9] One caveat is that the brute-force algorithm for QD_B of polynomial-sized circuits runs in time $\text{poly}(n) \cdot B(n)$, whereas the hypothesized running time in Theorem 2.6 is $B(n)/\ell$ rather than $\text{poly}(n) \cdot B(n)/\ell$, where $\ell = \log(B(n))^{\omega(1)}$. For large values of $B(n) \geq 2^{n^{\Omega(1)}}$ as in [Wil13] the difference is immaterial, since $\ell = n^{\omega(1)}$; but for $B(n) = 2^{n^{o(1)}}$ this leaves a small gap between the brute-force algorithm and an algorithm that would imply breakthrough circuit lower bounds.

2.3.3 Remarkably tight results for low-depth circuits and formulas

We now turn our attention to restricted classes of probabilistic algorithms. One of the most striking phenomena in the study of quantified derandomization has been *extraordinarily tight threshold results* for such restricted classes. Loosely speaking, we consider algorithms whose decision procedure on any fixed input as a function of the random coins can be decided by "weak" circuits (e.g., by constant-depth circuits, or by polynomial-sized formulas). Intuitively, denoting the latter class of circuits by $\mathcal{C}$, we will refer to two settings of parameters:

- **Tractable values** of $\mathcal{C}$ and B: There exists, unconditionally, an efficient algorithm that solves the corresponding quantified derandomization problem.
- **Threshold values** of $\mathcal{C}$ and B: If there exists an efficient algorithm that solves the corresponding quantified derandomization problem, then we can solve the general derandomization problem for a related class in a better-than-brute-force way, and deduce breakthrough results (e.g., new circuit lower bounds).

The notion of "efficient algorithm" above typically refers to polynomial time, but we will also allow (say) small quasipolynomial time. The crucial point is that the notion of efficiency in threshold values will not be stricter than the notion of efficiency in tractable values. Jumping ahead, the main takeaway from the results below is:

> **Main takeaway:** For several well-studied circuit classes, *the gap between tractable values and threshold values is tiny*, typically consisting only of low-order terms in specific parameters. A possible explanation for this recurrent phenomenon is suggested in Section 2.4.

The main difference between the case of general circuits from Section 2.3.2 and the case of restricted circuit classes presented next is that *we have many non-trivial algorithms for* QD_B *of restricted classes*. That is, for general circuits any better-than-brute-force algorithm would imply breakthrough circuit lower bounds, whereas for restricted classes we have algorithms that (significantly) outperform the better-than-brute-force algorithm, and yet essentially any improvement in the parameters of these currently known algorithms would imply breakthrough circuit lower bounds.

Bird's eye. The results below focus on three of the most well-studied "weak" classes in circuit complexity: The class $\mathcal{AC}^0$, the class $\mathcal{TC}^0$, and the class of De Morgan formulas (definitions appear below). The high-level picture for each of these classes is similar: Lower bounds for circuits of bounded size against $\mathcal{P}$ have been known for decades, and proving lower bounds for larger circuits (even against $\mathcal{E}^{\mathcal{NP}}$) is a major open problem; known algorithms for CAPP of circuits from the class have parameters that imply lower bounds for circuits of the same size as in the known (long-standing) lower bounds; and constructing better algorithms for CAPP of circuits from this class would imply lower bounds for larger circuits, yielding the sought breakthrough.

Circuit Class		Tractable Value	Threshold Value	
$\mathcal{AC}^0_d$	*#gates*	$n^{O(1)}$	$n^{O(1)}$	[GW14; CL16; Tel19]
	$B(n)$	$2^{n/\log^{d-2}(n)}$	$2^{n/\log^{d-14}(n)}$	
LTF_d	*#wires*	$n^{1+60^{-d}}$	$n^{1+1.61^{-d}}$	[Tel18; CT19]
	$B(n)$	$2^{n^{1-1.61^{-d}}}$	$2^{n^{1-1.61^{-d}}}$	
$\mathsf{PTF}_{d,O(1)}$	*#wires*	$n^{1+2^{-20d}}$	(same as LTF_d)	[KL18]
	$B(n)$	$2^{n^{.99}}$		
FORMULAS	*#leaves*	$n^{2-2\epsilon-o(1)}$	$n^{2-\epsilon+\Omega(1)}$	[CJW20]
	$B(n)$	2^{n^ϵ}	2^{n^ϵ}	

Figure 1: Tractable values vs threshold values for quantified derandomization of $\mathcal{AC}^0_d$, LTF_d, $\mathsf{PTF}_{d,O(1)}$, and FORMULAS (respectively denoting depth-d circuits, depth-d LTF circuits, depth-d PTF circuits of constant degree, and probabilistic De Morgan formulas). The running times of the algorithms are omitted for simplicity; they are either polynomial or quasipolynomial, and the required running times for threshold values are larger than the running times of the algorithms yielding the tractable values.

I'll now state the technical results for each of these classes, which are also concisely presented in Figure 1. For each class, I'll state both an unconditional algorithm for QD_B (demonstrating tractable values), and a result asserting that a slightly better algorithm for QD_B would yield an algorithm for CAPP that outperforms known algorithms and implies new lower bounds for the class (demonstrating threshold values). In some cases, the deduced lower bounds are not just a mild improvement over the known ones, but are actually dramatically better than the known ones. More detailed results as well as background and context on each of the classes appear in Section 6.

The class $\mathcal{AC}^0$ of constant-depth circuits. The class $\mathcal{AC}^0$ consists of circuit families of constant depth with gates of unbounded fan-in over the De Morgan basis. The following result from [Tel19] asserts that QD_B for polynomial-sized depth-d circuits can be solved by a polynomial-time computable .01-PRG with seed length $\tilde{O}(\log^3(n))$,

where $B(n) = 2^{n/\mathrm{polylog}(n)}$ and the seed length does not depend on the constant d.[10] That is, for such $B(n)$ the seed length is a *fixed polylogarithm, regardless of the (constant) depth*; in particular, this seed is much shorter than the seed of the known PRG for CAPP of such circuits, whose length is $\tilde{O}(\log^{d+1}(n))$ (see Section 6.2).

Complementing this upper bound, the result also asserts that CAPP for depth-d_0 circuits *efficiently reduces* to QD_B for depth-d circuits (for $d \gg d_0$), where again we have that $B(n) = 2^{n/\mathrm{polylog}(n)}$. (The proof of the foregoing statement relies on a technical construction of Cheng and Li [CL16].) Thus, the only gap between the two results is in the constant polylogarithmic power in the exponent of $B(n) = 2^{n/\mathrm{polylog}(n)}$.

Theorem 2.7 (tight threshold results for quantified derandomization of $\mathcal{AC}^0$; see [CL16; Tel19])**.** *Let $d \geq 2$ be a constant integer. Then:*

1. *For any $k \in \mathbb{N}$ there exists a .01-PRG with seed length $\tilde{O}(\log^3(n))$ that solves QD_B for depth-d circuits of size n^k, where $B(n) = 2^{\Omega(n/\log^{d-2}(n))}$.*
2. *For any $d_0 < d - 11$ it holds that CAPP for depth-d_0 circuits of linear size on n_0 input bits reduces, in deterministic polynomial time, to QD_B for depth-d circuits of polynomial size on $n = O(n_0^{3600})$ input bits, where $B(n) = 2^{n/\log^{d-d_0-11}(n)}$.*

To see the tiny gap between the two parts of Theorem 2.7, let us instantiate the threshold result with concrete parameters that would suffice to yield breakthrough results. Already for $B(n) = 2^{n/\log^{d-14}(n)}$, an algorithm for QD_B with running time as in Item (1) of Theorem 2.7 would yield a breakthrough algorithm for CAPP of $\mathcal{AC}^0$. In addition, for the slightly larger $B(n) = 2^{n/\log^{d-18}(n)}$, such an algorithm would imply breakthrough lower bounds for $\mathcal{AC}^0$: A problem in $\mathcal{E}^{\mathcal{NP}}$ that cannot be decided by $\mathcal{AC}^0$ circuits of depth 5 and size $2^{\Omega(n^{2/7})}$. See Section 6.2 for further details.

The class $\mathcal{TC}^0$ of constant-depth threshold circuits. The class $\mathcal{TC}^0$ consists of constant-depth circuit families with unbounded fan-in majority gates. We consider its natural extension called LTF circuits, in which each gate can compute an arbitrary linear threshold function (i.e., a function of the form $\Phi(x) = \mathrm{sgn}(\ell(x))$, where ℓ is a linear function over the reals). We measure the size of such circuits by their number of *wires*.

Constructing a CAPP algorithm for $\mathcal{TC}^0$ circuits of polynomial size is one of the most prominent current open problems in complexity theory (e.g., it was highlighted as the first open problem in Aaronson's "$\mathcal{P}$ vs $\mathcal{NP}$" survey [Aar16]). In the result below, we consider the fixed value $B(n) = 2^{n^{1-1.61^{-d}}}$, and state two complementary results. First, we state an unconditional algorithm for QD_B of LTF circuits of depth d with $n^{1+60^{-d}}$ wires that runs in almost-polynomial time, from [Tel18]; and we complement this by a result of Chen and the current author [CT19] asserting that CAPP for $\mathcal{TC}^0$ circuits reduces to QD_B of $\mathcal{TC}^0$ circuits of depth d with $n^{1+1.61^{-d}}$ wires. Indeed, ihe only gap between the two is in the precise constant $c > 1$ in the size bound $n^{1+c^{-d}}$.

[10] By "PRG that solves QD_B" we mean that the PRG fools circuits from the class that have at most $B(n)$ exceptional inputs, and thus enumerating over its seeds yields an algorithm for QD_B of this class.

Theorem 2.8 (tight threshold results for quantified derandomization of $\mathcal{TC}^0$; see [Tel18; CT19]). *Let $d \geq 2$ be a constant integer and let $B(n) = 2^{n^{1-1.61^{-d}}}$. Then:*

1. *There exists a deterministic algorithm that solves QD_B for depth-d LTF circuits with $n^{1+60^{-d}}$ wires in time $n^{O(\log\log^2(n))}$.*
2. *For any $d_0 \leq d - 7$ it holds that CAPP for depth-d_0 LTF circuits of linear size on n_0 input bits reduces, in deterministic polynomial time, to QD_B for depth-d LTF circuits on $n = n_0^{O_d(1)}$ input bits with $n^{1+1.61^{-d}}$ wires.*

To see the tightness of Theorem 2.8, note the following implication of the threshold result: If there exists an algorithm for QD_B as in Item (1) of Theorem 2.8 that can handle circuits with $n^{1+1.61^{-d}}$ wires (rather than $n^{1+60^{-d}}$ wires), then $\mathcal{NEXP}$ is hard for $\mathcal{TC}^0$ circuits of *arbitrary polynomial size*! In fact, this breakthrough implication would be obtained even from an algorithm that runs in time $2^{n^{o(1)}}$. See Section 6.3 for details.

Kabanets and Lu [KL18] extended the algorithm in Item (1) of Theorem 2.8, by constructing an algorithm for QD_B for the stronger class of low-degree PTF circuits (i.e., circuits in which each gate can compute any function of the form $\Phi(x) = \mathrm{sgn}(p(x))$, where p is a low-degree polynomial over the reals). When the PTF degree is constant, their algorithm runs in quasipolynomial time and can handle circuits of size $n^{1+c^{-d}}$ (for a large constant $c \gg 1$), and $B(n) = 2^{n^{1-\epsilon}}$ exceptional inputs, for a constant $\epsilon = \epsilon_c > 0$.

Theorem 2.9 (quantified derandomization of PTF circuits; see [KL18]). *For any two constants $\Delta, d \in \mathbb{N}$ there exists a deterministic algorithm that solves QD_B for depth-d PTF circuits of degree Δ with $n^{1+c^{-d}}$ wires in time $2^{\log(n)^{O(\Delta^2)}}$, where $B(n) = 2^{n^{0.99}}$ and $c = 2^{20}$.*

Compared to Theorem 2.8, the algorithm in Theorem 2.9 can handle circuits with stronger gates and of similar size (i.e., up to the constant c in the size bound $n^{1+c^{-d}}$). However, the algorithm tolerates fewer exceptional inputs $B(n)$, and has a mildly larger running time (i.e., quasipolynomial instead of almost-polynomial).

The class $\mathsf{FORMULAS}$. Chen, Jin, and Williams [CJW20] studied quantified derandomization of polynomial-sized formulas of fan-in two over the De Morgan basis, where the size is measured as the number of leaves. To make their results as tight as possible, they considered the stronger model of *probabilistic formulas*; thus, we are now concerned with the problem QD_B where the input is a description of a probabilistic formula (see Section 6.4 for details and precise definitions).

For $B(n) = 2^{n^\epsilon}$, where $\epsilon \in (0,1)$ may be any constant, on the one hand they constructed an unconditional polynomial-time algorithm for QD_B of probabilistic formulas of size $n^{2-2\epsilon-o(1)}$; and on the other hand, they showed that CAPP of polynomial-sized formulas reduces to QD_B of probabilistic formulas of size $n^{2-\epsilon+\delta}$, for an arbitrarily small constant $\delta > 0$. Thus, the gap between the two results is only in the size bound, which is $n^{2-2\epsilon-o(1)}$ in the upper bound and $n^{2-\epsilon+\Omega(1)}$ in the reduction.

Theorem 2.10 (tight threshold results for quantified derandomization of De Morgan formulas; see [CJW20]). *Let $\epsilon \in (0,1)$ and let $B(n) = 2^{n^\epsilon}$. Then:*

1. *There is a polynomial-time computable $(4/10)$-PRG with seed length $O(\log(n))$ that solves QD_B for probabilistic formulas of size $n^{2-2\epsilon-o(1)}$.*
2. *For any $\delta > 1$ and $k \in \mathbb{N}$ it holds that $\mathsf{CAPP}_{\frac{1}{2},0}$ for formulas on n_0 input bits of size n_0^k reduces, in deterministic polynomial time, to QD_B for probabilistic formulas of size $n^{2-\epsilon+\delta}$ on $n = (n_0)^{O_{\epsilon,\delta}(1)}$ input bits.*

The threshold result in Theorem 2.10 has the following implication: If there exists an algorithm for QD_B as in Item (1) that can handle probabilistic formulas of size $n^{2-\epsilon+\delta}$ for some $\delta > 0$ (rather than only $n^{2-2\epsilon-o(1)}$), then $\mathcal{NP}$ is hard for formulas of size n^k, for every fixed $k \in \mathbb{N}$. This is a very strong lower bound for formulas, the like of which has been sought since the 1960's. See Section 6.4 for details.

2.3.4 An outlier: Constant-depth circuits with parity gates

The last class that we consider in this section is $\mathcal{AC}^0[\oplus]$, which consists of constant-depth circuits with gates of unbounded fan-in over the basis $\{\mathsf{AND}, \mathsf{OR}, \oplus, \neg\}$. For simplicity, we assume that the input gates are both variables and their negations, and that circuits are layered (i.e., all gates of a given distance from the inputs are of the same type, and feed from gates in the preceding layer).[11]

Analogously to the results in Section 2.3.3, for this class too we have tractable values for the parameters and threshold values for the parameters. However, while these values are close, the currently known gap is not as tiny as in the results in Section 2.3.3. Specifically, both values refer to $B(n) = 2^{n^{\Omega(1)}}$; the known tractable values refer to polynomial-sized circuits of *depth three* that are of any form other than $\oplus \circ \mathsf{AND} \circ \oplus$, or that are of the latter form but satisfy certain structural constraints; whereas the threshold value refers to polynomial-sized circuits of *depth four*.

Theorem 2.11 (quantified derandomization of $\mathcal{AC}^0[\oplus]$ circuits; see [GW14] and [Tel19]). *Let $c \in \mathbb{N}$, let $\epsilon > 0$ be sufficiently small, and let $B(n) = 2^{n^\epsilon}$. Then,*

1. *There exists $\delta > 0$ and a polynomial-time algorithm that solves QD_B for $\mathcal{AC}^0[\oplus]$ circuits of* depth three, *provided that the circuit is either: (a) Of size n^c and* not *of the form $\oplus \circ \mathsf{AND} \circ \oplus$; or (b) Of the form $\oplus \circ \mathsf{AND} \circ \oplus$ and satisfies at least one of the following conditions: It has $O(n)$ gates; it has at most $n^{2-\delta}$ AND-gates and at most $n + n^{\delta/2}$ $\oplus$-gates; it has at most $\frac{1}{5} \cdot n^{1-\delta}$ AND gates and at most $n^{1+\delta}$ $\oplus$-gates.*
2. *The problem $\mathsf{CAPP}_{\frac{1}{2},0}$ for CNF formulas of polynomial size reduces in deterministic polynomial time to QD_B for $\mathcal{AC}^0[\oplus]$ circuits of polynomial size and* depth four.

[11] Note that any such circuit can be naturally thought of as computing a polynomial $\mathbb{F}_2^n \to \mathbb{F}_2$; indeed, we can assume that the internal gates are over the arithmetic basis $\{+, \times\} \cup \{1 + g : g \in \{+, \times\}\}$ and that the inputs are $x_1, ..., x_n, x_1 + 1, ..., x_n + 1 \in \mathbb{F}_2$.

To parse the threshold value in Theorem 2.11, recall that despite many decades of research, a polynomial-time algorithm for $\mathsf{CAPP}_{\frac{1}{2},0}$ of polynomial-sized CNF formulas is still not known, and would be a breakthrough. (See [ST17] for the best currently known algorithm, which runs in time $n^{\tilde{O}(\log\log(n))^2}$.) Further details regarding Theorem 2.11 can be found in [GW14, Section 6] and [Tel19, Section 6].

2.4 Natural black-box techniques and their limitations

Results as in Section 2.3.3, which are both very tight and consistent across many circuit classes, seem unlikely to be a mere coincidence. I'll now offer one explanation for this phenomenon (initially suggested in [Tel17]), which focuses on specific natural *black-box techniques* underlying the results above.

Algorithms for QD_B using pseudorandom restrictions. The high-level strategy for all algorithms for QD_B in Section 2.3.3 follows an idea of Goldreich and Wigderson [GW14]. Given an n-bit circuit C with at most $B(n)$ exceptional inputs, let $X \subseteq \{0,1\}^n$ be a subset of the domain of size $|X| > B(n)$ such that C is constant on X. Then, for every $x \in X$ it holds that $C(x)$ is the most common output of C (otherwise C would have $|X| > B(n)$ exceptional inputs). Thus, if we can efficiently find an input guaranteed to be in such a subset X, we can infer the most common output of C.

For example, in the case of $\mathcal{AC}^0$ a natural strategy is to use Håstad's switching lemma [Hås87] to find a *large subcube* on which C is constant. The key challenge, however, is that the restriction procedure underlying the switching lemma is randomized, and we want a *deterministic algorithm* that finds such a subcube. Accordingly, the technical result underlying the algorithm in Theorem 2.7 is a pseudorandom restriction procedure for $\mathcal{AC}^0$, whose key ingredient is a suitable derandomization of Håstad's switching lemma (see Appendix B for details of the technical result).[12]

In similar fashion, the technical results underlying the algorithms in Theorems 2.8, 2.9 and 2.10 are pseudorandom restriction procedures for threshold circuits and for formulas. These technical results are described in Appendix B.

Reductions of CAPP to QD_B using seeded extractors. The high-level strategy for the reductions of CAPP to QD_B is also similar in the different results above: Given a circuit C from the relevant class $\mathcal{C}$ that accepts (or rejects) all but $1/3$ of its inputs, we construct an averaging sampler Samp (equivalently, a seeded extractor; see Section 3.3), and output the circuit $C'(z) = \mathsf{MAJ}\{C(\mathsf{Samp}(z)_s)\}_s$, which has a very small number of exceptional inputs. (This is since Samp samples the event $C^{-1}(1)$ approximately correctly given all but a very small number of input strings z.)

[12] Note that for quantified derandomization we need to pseudorandomly choose both the variables to fix and the values to assign to them. This requirement is stronger than the one in other well-known applications (e.g., when constructing PRGs using the framework of [AW85]), in which only the set of fixed variables is chosen pseudorandomly (and values are uniform).

The crucial point is that the circuit C' is more complicated than the circuit C, since it needs to compute the averaging sampler Samp.[13] Accordingly, the reductions in Theorems 2.7, 2.8 and 2.10 are all based on constructions of averaging samplers (i.e., of seeded extractors) that are computable by small low-depth circuits and formulas. These technical results are described in Appendix C.

The limitation of black-box instantiations of these techniques. In the proofs of the results in Section 2.3.3, the two foregoing techniques are applied in a black-box manner: The restriction procedure for the given circuit C is a pseudorandom distribution $\mathbf{X}$ that does not depend on C, and instead simplifies *every* circuit from the class, with high probability; and the sampler samples *any* subset of $\{0,1\}^n$ approximately correctly, rather than only the subset $C^{-1}(1)$ corresponding to the given circuit C.[14]

The following result asserts that when the two techniques are applied in this black-box manner, the reduction of CAPP to QD_B yields parameters that the unconditional algorithm for quantified derandomization *provably cannot solve*. That is, the combination can only yield reductions of CAPP to $\mathsf{QD}_{B^{\mathtt{thr}}}$ for values of $B^{\mathtt{thr}}$ that are *necessarily larger* than the values of $B^{\mathtt{trac}}$ that the algorithm for $\mathsf{QD}_{B^{\mathtt{trac}}}$ can handle.

Theorem 2.12 (a limitation of two black-box techniques in quantified derandomization; informal, see Theorem 7.3 and [Tel17])**.** *Let $\mathcal{C}$ be a circuit class. Assume that there exists a reduction of CAPP for $\mathcal{C}$-circuits to $\mathsf{QD}_{B^{\mathtt{thr}}}$ that only uses seeded extractors computable in $\mathcal{C}$. Also assume that there exists an algorithm for $\mathsf{QD}_{B^{\mathtt{trac}}}$ of $\mathcal{C}$-circuits that only uses a distribution $\mathbf{X}$ over sets X of size $|X| > B^{\mathtt{trac}}$ that simplifies every $C \in \mathcal{C}$ to a constant, with high probability. Then, $B^{\mathtt{trac}}(n) < B^{\mathtt{thr}}(n)$.*

Thus, Theorem 2.12 suggests a possible explanation for the tightness of the results in Section 2.3.3: In the study of quantified derandomization we constructed technical tools (i.e., pseudorandom restriction procedures, and extractors computable by weak circuits or formulas) that achieve almost the best possible parameters, yielding results that are essentially as tight as possible when using the black-box techniques above.

The key takeaway from Theorem 2.12, in my opinion, is that it points at a *concrete technical challenge* that we need to tackle if we want to solve CAPP using this approach (i.e., the approach of first reducing the error and then finding an input guaranteed to belong to a large "simplifying subset"). Specifically, to get a CAPP algorithm using this approach we need to construct *either* non-black-box restriction procedures, *or* non-black-box samplers. I elaborate on this and mention potential directions in Section 7.3.

[13] I focus on the complexity of Samp since we can replace the majority function MAJ by an approximate majority function, which has low computational complexity. Moreover, in the case of derandomization with one-sided error we can simply replace this function with a single $\vee$ gate.

[14] To be more precise, there are differences between the clean abstract formulations of the techniques (as presented above) and the actual techniques underlying the results in Section 2.3, but these differences are immaterial for the argument that follows in Theorem 2.12. See Section 7 for further explanations.

2.5 Arithmetic quantified derandomization

So far we discussed derandomization problems in which both the computed functions and the computational devices are Boolean. In this section we switch context to *arithmetic derandomization problems*, in which the functions and devices work over a predetermined field, and the complexity is measured by algebraic notions.

Specifically, a well-known arithmetic derandomization problem is that of constructing hitting-set generators (HSGs) for multivariate polynomials. Fixing a finite field $\mathbb{F}$, we consider a class $\mathcal{C}$ of polynomials $\mathbb{F}^n \to \mathbb{F}$. Our goal is to construct a generator $H\colon \{0,1\}^\ell \to \mathbb{F}^n$ such that for every polynomial $p \in \mathcal{C}$ there exists s for which $p(H(s)) \neq 0$; ideally, the seed length ℓ is small and H is efficiently computable. Two well-known lines of work focus on polynomials of low degree (see [NN93; LV98; Bog05; BV10; BHS08; Lov09; Vio09b; CTS13]) and on polynomials with a bounded number of monomials (see [LVW93; KS01; Lu12; ST18; ST19]).

2.5.1 The quantified problem, and a digest of what we know

A natural quantified version of the problem above is that of constructing hitting-set generators for polynomials that vanish extremely rarely. Specifically, for a given parameter $\epsilon > 0$, we want to construct a HSG for the class of polynomials $p\colon \mathbb{F}^n \to \mathbb{F}$ whose fraction of roots satisfies $\Pr_{x\in\mathbb{F}^n}[p(x) = 0] \leq \epsilon$. To parse the problem, recall that over large fields, any polynomial of degree $d \ll |\mathbb{F}|$ vanishes "rarely", i.e. on at most $\epsilon = d/|\mathbb{F}|$ fraction of its inputs. However, we will care about *extremely small values* of $\epsilon \ll d/|\mathbb{F}|$, and won't necessarily assume that the field is large (e.g., we also care about polynomials over $\mathbb{F}_2$), making the problem highly non-trivial.

Denote the field size by $q = |\mathbb{F}|$, the total degree of the polynomial by d, and the bound on the fraction of roots by $\epsilon = q^{-t}$ for an integer parameter $t \in \mathbb{N}$. The reason for the latter notation is that a random polynomial vanishes with probability q^{-1}, whereas a classical result of Warning [War35] asserts that any degree-d polynomial vanishing with probability less than q^{-d} has no roots at all. Thus, our interest is in $\epsilon = q^{-t}$ for values of $t = 1, ..., d$, where the value $t = 1$ will be "hard" (intuitively, since most polynomials vanish with probability about q^{-1}) and for any value $t > d$ the problem becomes trivial (as such polynomials have no roots).

Goldreich and Wigderson [GW14] proved the first result in this context, which was a HSG construction. A subsequent lower bound on the seed length of such HSGs was proved in [Tel19]. Both of these results were later subsumed by more general results proved by Doron, Ta-Shma, and the current author [DTST20], which I now describe. The main takeaway from the known results is the following:

> **Main takeaway:** There exist HSGs for degree-d polynomials that vanish with extremely small probability ϵ, with shorter seed than any HSG for general degree-d polynomials. However, there is currently a significant gap between the known lower- and upper-bounds on values of ϵ for which an improvement over HSGs for general degree-d polynomials is possible.

2.5.2 Constructions of HSGs

Over $\mathbb{F} = \mathbb{F}_2$, there exist HSGs for degree-d polynomials that vanish with extremely small probability ϵ whose seed length is smaller than the seed length of any possible HSG for general degree-d polynomials. Specifically, recall that there is a non-uniform HSG for general degree-d polynomials with seed length $\Theta(d \cdot \log(n/d))$, and that no HSG can have a shorter seed, regardless of its complexity. The following result asserts the existence of two HSGs for polynomials that vanish rarely – a non-uniform HSG and a polynomial-time-computable HSG – both of which have seed length $o(d \cdot \log(n/d))$ when the fraction of roots is at most $\epsilon \approx 2^{-d+o(d)}$:

Theorem 2.13 (HSGs for $\mathbb{F}_2$-polynomials that vanish rarely; see [DTST20])**.** *Let $n \in \mathbb{N}$ be sufficiently large and let $d > t + 4$ be integers. Then, there exist HSGs for degree-d polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish with probability at most $\epsilon = 2^{-t}$ of the two following types:*

1. *The HSG is* **non-uniform** *and has seed length is $O\left(d-t\right) \cdot \log\left(\frac{n}{d-t}\right)$.*
2. *The HSG is* **polynomial-time computable** *and has seed length $\tilde{O}(2^{d-t}) \cdot \log\left(\frac{n}{d-t}\right)$.*

The non-uniform HSG in Theorem 2.13 has seed length significantly smaller than $d \cdot \log(n/d)$ when $\epsilon \leq 2^{-d+d^{\Omega(1)}}$ (i.e., for $t = d - d^{\Omega(1)}$); and the polynomial-time computable HSG achieves such short seed when $\epsilon \leq 2^{-d+(1-\Omega(1))\cdot\log(d)}$ (i.e., for $t = d - (1-\Omega(1)) \cdot \log(d)$). Thus, intuitively, both of these results hold only for extremely small values of $\epsilon \approx 2^{-d+o(d)}$. Moreover, both results hold only over $\mathbb{F}_2$, and we don't currently have analogous constructions over larger fields.

2.5.3 Impossibility results

Complementing the constructions above, for fields of *any size* $q \in \{2, ..., \text{poly}(n)\}$, we have a lower bound on the seed length of any HSG for polynomials that have at most $\epsilon = q^{-t}$ roots, regardless of its complexity. Loosely speaking, while the (non-uniform) HSG in Theorem 2.13 has seed length $(d-t) \cdot \log(n)$, the lower bound below rules out a much shorter seed, of length $o(d/t) \cdot \log(n)$:

Theorem 2.14 (a lower bound on HSGs for polynomials that vanish rarely; see [DTST20])**.** *Let $n, d, t \in \mathbb{N}$ such that $d \leq n^{.49}$ and $t \leq \gamma \cdot d$, where $\gamma > 0$ is a universal constant, and let $\mathbb{F}$ be a field of size $2 \leq q \leq n^{100}$.* [15] *Then, any HSG for degree-d polynomials $\mathbb{F}_q^n \to \mathbb{F}_q$ requires a seed of length $\Omega((d/t) \cdot \log(n))$.*

Intuitively, the meaning of Theorem 2.14 is that the seed length of HSGs for polynomials that vanish rarely may only be smaller than that of HSGs for general polynomials for values of $\epsilon \leq q^{-d^{\Omega(1)}}$. [16] Indeed, there is a significant gap between the latter value and the smaller values of ϵ at which we already *know* from Theorem 2.13 that savings can occur, which are of the form $\epsilon \leq q^{-d+d^{\Omega(1)}}$, where $q = 2$.

[15] The constant power 100 may be replaced with any constant power; see Theorem 8.6.

[16] Needless to say, this lower bound does not rule out the possibility that it might nevertheless be *easier to construct efficient HSGs* in the case of polynomials that vanish rarely.

2.5.4 Two comments

I find it remarkable that we don't even know what is the optimal seed length of a *non-uniform* HSG for polynomials that vanish rarely. (For general polynomials, the standard answer of $\Theta(d \cdot \log(n/d))$ is easy to prove; see e.g., [DTST20, Preliminaries].) The upper bound in Theorem 2.13 is obtained by a standard probabilistic argument, but it relies on a non-trivial tight bound on the number of such polynomials proved by Kaufman, Lovett, and Porat [KLP12]. The lower bound in Theorem 2.14 is proved by a non-standard argument that relies on complicated technical tools (e.g., the disperser construction of Shaltiel and Umans [SU05]).

The second comment is that the question of the optimal seed length of HSGs for polynomials that vanish rarely turns out to be closely related to a *clean algebraic problem*. Specifically, define the degree-d closure of a set $S \subseteq \mathbb{F}^n$ to be the variety induced by the set of degree-d polynomials that vanish on S. [17] It is natural to ask whether or not there exist small sets with a large degree-d closure, and it turns out that such sets are essentially the hitting-sets for polynomials that vanish rarely (i.e., the two notions are closely related). See Theorem 8.8 for an explanation and precise details.

2.6 The connection to pseudoentropy

Lastly, it turns out that there is a very close relationship between quantified derandomization and pseudoentropy. In high-level, as first noted by Doron *et al.* [DMO+20] (relying on a technical result of Barak, Shaltiel and Wigderson [BSW03]), *PRGs for quantified derandomization are essentially equivalent to pseudoentropy generators*.

Recall that, informally, a distribution $\mathbf{w}$ over $\{0,1\}^n$ is pseudoentropic if it looks to every efficient distinguisher as though it has high entropy; this is a relaxation of pseudorandomness, since the latter is the special case with full entropy n (i.e., the uniform distribution). We are interested in a very weak notion of pseudoentropy, called metric pseudoentropy: A distribution has metric ϵ-pseudoentropy k if every efficient distinguisher C views $\mathbf{w}$ as ϵ-close to a distribution $\mathbf{h}_C$ with min-entropy at least k, where $\mathbf{h}_C$ may depend on the distinguisher C (see Definition 9.1).

The following result asserts that a distribution has metric pseudoentropy k if and only if the distribution fools distinguishers with $\Theta(2^k)$ exceptional inputs. In particular, this means that an efficient algorithm is a metric pseudoentropy generator (PEG) if and only if the algorithm solves QD_B, where the pseudoentropy is $k \approx \log(B(n))$.

Theorem 2.15 (metric pseudoentropy vs quantified derandomization; see Theorem 9.2)**.** *A distribution* $\mathbf{w}$ *over* $\{0,1\}^n$ *has metric* ϵ*-pseudoentropy* k *for size-*S *circuits* if and only if *it is* $\Theta(\epsilon)$*-pseudorandom for size-*S *circuits with at most* $B(n) = \Theta(2^k)$ *exceptional inputs.*

I suspect that implications of the connection in Theorem 2.15 can be explored further than they have been so far. One interesting implication, which was shown by Chen and the current author [CT21b], is the following.

[17] That is, the degree-d closure is the set of all points $x \in \mathbb{F}^n$ that satisfy the following: For every degree-d polynomial p that vanishes on S it holds that $p(x) = 0$.

A well-known paradigm for constructing PRGs for general derandomization, which dates back to [HIL+99] and has been used in [BSW03; STV01; FSU+13; DMO+20], is to construct a PRG that composes a PEG with a seeded extractor. Specifically, denoting the PEG by G_0, the PRG is $G(s_0, s_1) = \mathsf{Ext}(G_0(s_0), s_1)$. Implementing this idea has been challenging, since the original proofs that G is a PRG require the PEG to satisfy strong notions of pseudoentropy (i.e., stronger than metric pseudoentropy).

Doron *et al.* [DMO+20] proved that it suffices for G_0 to be a metric PEG, but for an inherently stronger class of distinguishers (see [DMO+20, Theorem 1.8]). Relying on Theorem 2.15 we can prove something stronger: It suffices for G_0 to be a metric PEG for the distinguisher class that we want to fool, up to a polynomial size overhead.

Theorem 2.16 ("extract from a pseudoentropic string" as a special case of "error-reduction and quantified derandomization"; see Theorem 9.3)**.** *Let* $\mathsf{Ext}\colon \{0,1\}^{\bar{n}} \times \{0,1\}^{O(\log(\bar{n}))} \to \{0,1\}^n$ *be an extractor with error* 10^{-3} *for min-entropy* $k \leq n - 10$ *that is computable in time* $\mathrm{poly}(n)$*, and let* $G_0\colon \{0,1\}^{\ell} \to \{0,1\}^{\bar{n}}$ *be a metric* 10^{-3}*-pseudoentropy generator with pseudoentropy* $k + 10$ *for circuits of size* n^c *(where* $c > 1$ *is a sufficiently large constant). Then,* $G(s_0, s_1) = \mathsf{Ext}(G_0(s_0), s_1)$ *is a* .01*-PRG for linear-sized circuits.*

The reader may wonder where lies the connection of Theorem 2.16 to quantified derandomization. The connection is in the *proof*: Instead of thinking of G as "extracting from a pseudoentropic string" (as in previous proofs, which were technically involved), we think of G as reducing the general derandomization problem to quantified derandomization and then solving the latter. Specifically, we think of Ext as performing error-reduction in a non-standard way, and of G_0 as solving the resulting quantified derandomization problem (relying on Theorem 2.15). This alternative perspective yields a short, elementary, and general proof (see Theorem 9.3).

3 Preliminaries

The machine model throughout this survey will be the multitape Turing machine. We say that $T\colon \mathbb{N} \to \mathbb{N}$ is a time function if T is time-computable and satisfies $T(n) \geq n$; whenever referring to time bounds we'll always implicitly assume that the bound is a time function. We denote random variables and distributions by boldface, and denote the uniform distribution over $\{0,1\}^n$ by $\mathbf{u}_n$.

3.1 Complete problems

Recall the following standard definition of CAPP, which uses two parameters and also allows capturing derandomization of algorithms with one-sided error:

Definition 3.1 (CAPP)**.** *The* Circuit Acceptance Probability Problem with parameters (α, β) *(*$\mathsf{CAPP}_{\alpha,\beta}$*, in short) is the following promise problem* (Y, N)*:*

1. *The set of* Y *instances consists of circuits* C *such that* $\Pr_x[C(x) = 1] \geq \alpha$*.*

2. *The set of* N *instances consists of circuits* C *such that* $\Pr_x[C(x) = 1] \leq \beta$

A standard result is that CAPP is complete for $prBPP$ under deterministic polynomial-time reductions (see, e.g., [Vad12, Corollary 2.31] or [Gol08, Exercise 6.14]). The following generalization, which was mentioned in Section 2.3, implies that for any function B, the problem QD_B defined in Definition 2.5 is complete for $\mathcal{BPTIME}_B$:

Theorem 3.2 (QD_B is complete for $\mathcal{BPTIME}_B$)**.** *For two time functions* $T, R\colon \mathbb{N} \to \mathbb{N}$*, let* $\Pi = (\mathsf{Y}, \mathsf{N})$ *be a promise problem that can be solved by a probabilistic algorithm that runs in time* T*, uses* R *random coins, and errs on any input given at most* $B(R)$ *random strings. Then,* Π *on inputs of length* n *reduces in deterministic time* $\tilde{O}(T(n))$ *to* QD_B *for circuits on* $R(n)$ *input bits of size* $\tilde{O}(T(n))$*.*

Proof. Fix a probabilistic multitape machine M that solves Π on n-bit inputs in time $T = T(n)$, with $R = R(n)$ random coins, and with at most $B(R)$ exceptional inputs. Given $x \in \{0,1\}^n$, we construct a circuit $C_x\colon \{0,1\}^R \to \{0,1\}$ that computes the decision of M at the fixed input x as a function of the random coins. This R-bit circuit is of size $\tilde{O}(T)$, and either accepts all but at most $B = B(R)$ of its input strings (in case $x \in \mathsf{Y}$) or rejects all but at most B of its input strings (in case $x \in \mathsf{N}$). ■

Recall that $prBPTIME_B$ was defined in Definition 1.1 under the simplifying assumption that the number of random coins R equals the running time T. Thus, as a corollary of Theorem 3.2, if QD_B for n-bit circuits of size $\tilde{O}(n)$ can be solved in deterministic time $\bar{T}$, then $prBPTIME_B[T] \subseteq prDTIME[\bar{T}(\tilde{O}(T(n)))]$.

3.2 PRGs for quantified derandomization

Many of the known algorithms for QD_B solve the problem in a black-box fashion, using PRGs. To properly define PRGs for QD_B we define a class of B-biased distinguishers, which are n-bit circuits that have at most $B(n)$ exceptional inputs. Then, PRGs that solve QD_B are simply PRGs that fool the class of B-biased distinguishers.

Definition 3.3 (B-biased circuits)**.** *For a function* $B\colon \mathbb{N} \to \mathbb{N}$*, we say that a Boolean circuit* C *with* n *input gates is* B-biased *if there exists* $\sigma \in \{0,1\}$ *such that* C *outputs* σ *given all but at most* $B(n)$ *input strings. Unless explicitly stated otherwise, we assume that* B*-biased distinguishers are circuits of linear size.*

Definition 3.4 (PRG)**.** *We say that a distribution* $\mathbf{w}$ *over* $\{0,1\}^n$ *is* ϵ-pseudorandom *for a class* $\mathcal{C}_n$ *of functions* $\{0,1\}^n \to \{0,1\}$ *if for every* $C \in \mathcal{C}_n$ *it holds that* $|\Pr[C(\mathbf{u}_n) = 1] - \Pr[C(\mathbf{w}) = 1]| \leq \epsilon$*. We say that an algorithm* G *is a* pseudorandom generator (PRG) with seed length $\ell(n)$ for a class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *if, when* G *takes input* 1^n *and a random seed of length* $\ell(n)$*, its output distribution is* ϵ*-pseudorandom for* $\mathcal{C}_n$*.*

Note that for any $B(n) = o(2^n)$, an ϵ-PRG G for a class of B-biased distinguishers satisfies the following: For every B-biased distinguisher C, if the acceptance probability of C is high then $\Pr_s[C(G(s)) = 1] \geq 1 - \epsilon - o(1)$, and if the acceptance probability of C is low then $\Pr_s[C(G(s)) = 1] \leq \epsilon - o(1)$.

3.3 Extractors and samplers

Recall that standard definition of seeded randomness extractors, which we'll simply refer to as extractors in this survey.

Definition 3.5 (min-entropy)**.** *The* min-entropy *of a distribution* $\mathbf{x}$*, denoted* $H_\infty(\mathbf{x})$*, is the largest integer* k *such that for every outcome* x *it holds that* $\Pr[\mathbf{x} = x] \leq 2^{-k}$*.*

Definition 3.6 (extractors)**.** *A function* $\mathsf{Ext}\colon \{0,1\}^{\bar{n}} \times \{0,1\}^{\ell} \to \{0,1\}^{n}$ *is an* extractor for min-entropy k with error $\epsilon > 0$ *(or a* (k,ϵ)*-extractor, in short) if for every distribution* $\mathbf{x}$ *over* $\{0,1\}^{\bar{n}}$ *with min-entropy* $H_\infty(\mathbf{x}) \geq k$*, the statistical distance between* $\mathsf{Ext}(\mathbf{x}, \mathbf{u}_\ell)$ *and* $\mathbf{u}_n$ *is at most* ϵ*.*

It is well-known that extractors are essentially equivalent to averaging samplers, which we'll refer to as samplers. Let us recall the definition of the latter and state this equivalence. The definition presented next is slightly non-standard, since instead of bounding the error probability of the sampler we will bound the *number* of random strings that are bad for the sampler.

Definition 3.7 (averaging samplers)**.** *A function* $\mathsf{Samp}\colon \{0,1\}^{\bar{n}} \to (\{0,1\}^{n})^{2^{\ell}}$ *is an averaging sampler with* B *bad inputs and error* $\epsilon > 0$ *(or a* (B,ϵ)*-sampler, in short) if for every* $T \subseteq \{0,1\}^{n}$*, for all but at most* B *strings* $z \in \{0,1\}^{\bar{n}}$ *it holds that* $\Pr_{s\in[2^{\ell}]}[\mathsf{Samp}(z)_s \in T] \in |T|/2^{n} \pm \epsilon$*.*

Theorem 3.8 (extractors are equivalent to samplers; see, e.g., [Vad12, Corollary 6.24])**.** *Consider two functions* $\mathsf{Ext}\colon \{0,1\}^{\bar{n}} \times \{0,1\}^{\ell} \to \{0,1\}^{n}$ *and* $\mathsf{Samp}\colon \{0,1\}^{\bar{n}} \to (\{0,1\}^{n})^{2^{\ell}}$ *such that for every* $z \in \{0,1\}^{\bar{n}}$ *and* $s \in \{0,1\}^{\ell}$ *it holds that* $\mathsf{Ext}(z,s) = \mathsf{Samp}(z)_s$*. Then,*

1. *If* Ext *is a* (k,ϵ)*-extractor then* Samp *is a* $(2^{k},\epsilon)$*-sampler.*
2. *If* Samp *is an* $(2^{k},\epsilon)$*-sampler then* Ext *is a* $(k + \log(1/\epsilon), 2\epsilon)$*-extractor.*

Given Theorem 3.8, the standard technique for error-reduction of probabilistic algorithm relies on extractors. Specifically, for a probabilistic algorithm A taking input x and randomness r that errs with probability at most $1/3$, and a $(k,.01)$-extractor Ext, we define $A'(x,r') = \mathsf{Maj}\{A(x,\mathsf{Ext}(r',s))\}_s$. Relying on Item (1) of Theorem 3.8, for any fixed input x we have that $A'(x,r')$ outputs the correct decision for all but at most 2^{k} exceptional random choices of r'.

Example 3.9. Using one of the state-of-the-art constructions of a polynomial-time computable extractor (e.g., from [LRV+03; GUV09; DKS+13]), for any constant $\epsilon > 0$ we can reduce the number of exceptional random strings to $B(T) = 2^{T^{\epsilon}}$ with only a polynomial overhead in the time complexity and in the number of random coins. More generally, for any constant $\epsilon > 0$ and any "nice" function B, we can convert a probabilistic algorithm with m random coins that errs with probability $1/3$ into an algorithm

that solves the same problem with $n = B^{-1}(2^{m/(1-\epsilon)})$ random coins and at most $B(n)$ exceptional random strings, incurring a multiplicative time overhead of $\mathrm{poly}(n)$.[18]

However, when considering classes of weak algorithms (e.g., uniform constant-depth circuits), in many cases these classes *provably* cannot compute extractors with parameters as above (see, e.g., Section 2.4, or [Vio05; GVW15]). Thus, for such classes this method can only yield relatively large target values of B.

4 Non-uniform derandomization

As in all derandomization problems, the non-uniform (i.e., non-explicit) results are a good starting point, since they suggest what we might ideally hope to achieve in a uniform algorithmic way.

Recall that Adleman's [Adl78] theorem asserts that $\mathcal{BPP} \subset \mathcal{P}/\mathrm{poly}$. In fact, using his argument we can get the finer bound $pr\mathcal{BPTIME}[T] \subset pr\mathcal{SIZE}[n \cdot \tilde{O}(T)]$, for every time function $T(n)$.[19] It is a great (and quite recent) open question whether or not the multiplicative overhead of n is actually necessary for general worst-case derandomization. Under a reasonable hardness assumption the overhead is indeed necessary, but this assumption is not very well understood at this point. (The assumption is the non-uniform variant of #NSETH; see [CT21b] for details.)

Nevertheless, for quantified derandomization we can *unconditionally do better*. As stated in Theorem 2.1, even for a relatively large number of exceptional inputs $B(T) = 2^{(1-\Omega(1))\cdot T}$ we can avoid the multiplicative overhead of n, and non-uniformly derandomize $pr\mathcal{BPTIME}_B$ with almost no overhead at all. In fact, as stated in the following result, we can do so in a black-box manner, by constructing a (non-uniform) PRG for B-biased circuits with a very short seed:

Theorem 4.1 (non-uniform PRGs for quantified derandomization)**.** *Let $\epsilon > 0$ be an arbitrarily small constant, let $B(n) = 2^{(1-\epsilon)\cdot n}$, and let $s = s(n)$. Then, there is a non-uniform .01-PRG with seed length $\log(s(n)/n) + \mathrm{loglog}(s(n)) + O(1)$ (and size exponential in its seed length) for the class of n-bit circuits of size $s(n)$ that are B-biased.*

Proof. Choose at random a set $W \subseteq \{0,1\}^n$ of size $|W| = \Theta(s'/n)$, where $s' = O(s \cdot \log(s))$. For every n-bit B-biased circuit C of size s, the probability that C evaluates to

[18] Other tradeoffs can also be obtained. For example, we can mitigate the blow-up in the number n of variables at the cost of obtaining larger circuits, using the lossless extractors with super-logarithmic seed of [GUV09, Theorem 5.14]. Moreover, when reducing the error of algorithms that have one-sided error, we can mitigate the blow-up in n while still paying only a multiplicative time overhead of $\mathrm{poly}(n)$, using the lossless disperser with logarithmic seed [TSUZ07, Theorem 1.4].

[19] To do so, we reduce the error of a given probabilistic algorithm below 2^{-n} using $O(n)$ repetitions, and then hard-wire a fixed choice of an $O(T \cdot n)$-bit random string that leads the algorithm to a correct decision on all 2^n inputs. The polylogarithmic overhead on T arises from technical issues of translations between uniform and non-uniform models (i.e., converting a Turing machine to a circuit).

its minority output on at least $.01 \cdot |W|$ of the strings in W is at most

$$\sum_{i=.01\cdot|W|}^{|W|} \binom{|W|}{i} \cdot (B/2^n)^i \cdot (1 - B/2^n)^{|W|-i} < |W| \cdot 2^{|W|} \cdot 2^{-\Omega_\epsilon(|W|\cdot n)} < 2^{-s'(n)} ,$$

where the last inequality relies on our choice of $|W|$. By a union-bound over $2^{s'(n)}$ circuits of size s, there exists a choice of W such that every B-biased size-s circuit evaluates to its majority output on at least $.99 \cdot |W|$ of the strings in W. ■

Thus, Theorems 2.1 and 4.1 suggest that even for a relatively large number of exceptional inputs $B(T) \approx 2^{.99 \cdot T}$, we can hope to get black-box quantified derandomization with almost no time overhead. Interestingly, the only setting in which quantified derandomization with such a large number of exceptional inputs is known is that of logspace algorithms, which is described in Appendix D.

5 Hardness vs quantified randomness

As explained in Section 2.2, assuming sufficiently strong lower bounds for non-uniform circuits we can make the derandomization results in Section 4 explicit. The following two results refer to algorithms that err on at most $B(n) = 2^{n^{1-\epsilon}}$ strings, for a very small $\epsilon > 0$, and show that under reasonable hardness hypotheses such algorithms can be derandomized with overhead that is just slightly larger than the one in Theorem 2.1.

The required hardness hypotheses in the results are that there exists a function in $\mathcal{E}$ that is hard for $\mathcal{SVN}$ circuits (the acronym stands for "Single-Value Non-deterministic circuits"), which are the non-uniform analogue of $\mathcal{NP} \cap co\mathcal{NP}$ defined as follows:

Definition 5.1 ($\mathcal{SVN}$ circuits)**.** *We say that a promise problem $\Pi = (\mathsf{Y}, \mathsf{N}) \subseteq \{0,1\}^* \times \{0,1\}^*$ is* solved by $\mathcal{SVN}$ circuits of size $s = s(n)$ *if there exists a circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that each C_n has n input gates, at most $s(n)$* non-deterministic input gates*, and at most $s(n)$ internal gates, and the following holds:*

1. *For any $x \in \mathsf{Y}$ there exists an assignment $w \in \{0,1\}^{s(|x|)}$ to the non-deterministic input gates such that $C_{|x|}(x, w) = 1$; and for all assignments $w' \in \{0,1\}^{s(|x|)}$ to the non-deterministic input gates it holds that $C_{|x|}(x, w') \in \{1, \perp\}$.*

2. *For any $x \in \mathsf{N}$ there exists an assignment $w \in \{0,1\}^{s(|x|)}$ to the non-deterministic input gates such that $C_{|x|}(x, w) = 0$; and for all assignments $w' \in \{0,1\}^{s(|x|)}$ to the non-deterministic input gates it holds that $C_{|x|}(x, w') \in \{0, \perp\}$.*

5.1 Lower bounds for $\mathcal{SVN}$ circuits imply very fast quantified derandomization

The first result, which was proved by Doron *et al.* [DMO+20], asserts that if there is a function in $\mathcal{DTIME}[2^{(1+\epsilon)\cdot n}]$ that is hard for $\mathcal{SVN}$ circuits of size $2^{(1-\epsilon)\cdot n}$, then we

can derandomize T-time algorithms that err on at most $2^{T^{1-O(\epsilon)}}$ random strings in time $T^{2+\epsilon}$. Moreover, they show that for any input length n, one can compute in advance, in time $T^{2+\epsilon}(n)$, a single pseudorandom set that later on allows to derandomize each particular algorithms with at most $2^{T^{1-O(\epsilon)}(n)}$ exceptional random strings on n-bit inputs in time $T^{1+\epsilon}(n)$; that is, the derandomization time $T^{2+\epsilon}$ can be partitioned to a preprocessing time of $T^{2+\epsilon}$, which is paid only once per input length for all algorithms, and then a derandomization time of $T^{1+\epsilon}$ for any particular algorithm.

Theorem 5.2 (hardness to quantified randomness with a near-quadratic overhead; see [DMO+20])**.** *For $\epsilon > 0$, assume that there exists $L \in \mathcal{DTIME}[2^{(1+\epsilon)\cdot n}]$ that cannot be computed by $\mathcal{SVN}$ circuits of size $2^{(1-\epsilon)\cdot n}$ for all sufficiently large $n \in \mathbb{N}$. Then, there exists a .01-PRG with seed length $O(\epsilon) \cdot \log(n)$ for B-biased circuits, where $B(n) = 2^{n^{1-O(\epsilon)}}$, such that the entire output-set of the PRG can be printed in time $n^{2+O(\epsilon)}$.*

The O-notation in Theorem 5.2 hides universal constants, and in their result the concluded PRG actually has a smaller error, of magnitude $n^{-\Omega(1)}$. (Reducing the error from $1/8$ to $n^{-\Omega(1)}$ does not improve the resulting algorithm for QD_B.)

Next, the following result from Chen and the current author [CT21b] shows a faster derandomization under a stronger hypothesis. Namely, if the entire truth-table of the hard function can be computed "in a batch" in time $2^{(1+\epsilon)\cdot n}$ (rather than just assuming that each entry is computable in such time as in Theorem 5.2), then the derandomization time is only $T^{1+\epsilon}$ rather than $T^{2+\epsilon}$. (The following result statement also parameterizes $B(n)$ and the hardness assumption separately.)

Theorem 5.3 (hardness to quantified randomness with a near-linear overhead; see [CT21b])**.** *For $\epsilon, \delta > 0$, assume that there exists $L \in \mathcal{DTIME}[2^n]$ such that for every $n \in \mathbb{N}$ the truth-table of L can be printed in time $2^{(1+\epsilon)\cdot n}$, but L cannot be computed by $\mathcal{SVN}$ circuits of size $2^{(1-\delta/4)\cdot n}$. Then, there exists a .01-PRG with seed length $\delta \cdot \log(n)$ for B-biased circuits, where $B(n) = 2^{n^{1-\delta}}$, such that the entire output-set of the PRG can be printed in time $n^{(1+\delta)\cdot(1+\epsilon)}$.*

Similarly to Theorem 5.2, in Theorem 5.3 too the O-notation hides a universal constant and the error in the original result is actually $n^{-\Omega(1)}$.

5.2 The common proof idea

The proofs of Theorems 5.2 and 5.3 can be seen as efficient implementations of a proof strategy that was introduced by Sipser [Sip86], and that predates the well-known reconstruction argument of Nisan [Nis91] and Nisan and Wigderson [NW94].

In high-level, the derandomization algorithm computes the truth-table f of the hard function; encodes this truth-table by an appropriate error-correcting code, to obtain a codeword $\bar{f}$; and outputs disjoint consecutive substrings of $\bar{f}$ as the pseudorandom set of strings. Specifically, to derandomize a probabilistic algorithm on an n-bit input, we compute the truth-table of the hard function on all inputs of length $\ell \approx (1+\epsilon) \cdot \log(n)$, so that the truth-table is of length $|f| \approx n^{1+\epsilon}$, and encode it to $\bar{f}$ of length $\approx n^{1+O(\epsilon)}$; this yields a pseudorandom set of $n^{O(\epsilon)}$ strings of length n.

To prove that this set is indeed pseudorandom, we assume that there exists a B-biased distinguisher D and construct a corresponding $\mathcal{SVN}$ circuit C of size $|f|^{1-\Omega(1)}$ that computes the hard function, which contradicts the hardness of f. To see how this is done, recall that the pseudorandom set is just consecutive substrings of $\bar{f}$. The key observation is that since the set $S \subseteq \{0,1\}^n$ of exceptional inputs of D is relatively small, i.e. of size only $B(n) \approx 2^{n^{1-\delta}}$, and a .01-fraction of the substrings of $\bar{f}$ lie in S, there exist short descriptions for a .01 of the substrings in $\bar{f}$. Specifically, we can describe each substring by its index in S, which yields a description of length $(.01 \cdot n^{O(\epsilon)}) \cdot \log(B(n)) = |f|^{1-\Omega(1)}$.

The observation above only yields an information-theoretic description of .01 of the entires of $\bar{f}$, but with some work we can also turn it into a relatively efficient procedure; that is, we can construct an $\mathcal{SVN}$ circuit of size $|f|^{1-\Omega(1)}$ computing a function that agrees with the function whose truth-table is $\bar{f}$ on .01 of the inputs. Given such a circuit, a local list-decoding procedure for the code yields an $\mathcal{SVN}$ circuit of size $|f|^{1-\Omega(1)}$ that computes the hard function f on all inputs. See [CT21b] for the full argument, which can be used to prove both results above.

5.3 Non-trivial PRGs for biased distinguishers imply circuit lower bounds

For general derandomization, the classical hardness vs randomness framework asserts that PRGs are essentially equivalent to circuit lower bounds for $\mathcal{E}$ (see, e.g., [Nis91; NW94; IW99; SU05; Uma03] and others).[20] Indeed, both Theorems 5.2 and 5.3 assert the existence of a PRG with seed length $\approx \epsilon \cdot \log(n)$ for biased distinguishers. We do not know if such a PRG is equivalent to lower bounds for $\mathcal{SVN}$ circuits as in the hypotheses of these results, but we do know that any non-trivial PRG for biased distinguishers necessitates lower bounds for *standard* circuits in $\mathcal{E}$. In fact, this relies on essentially the same argument as in the standard proof that PRGs for general derandomization require circuit lower bounds in $\mathcal{E}$.

To formally state this result, we consider a class $\mathcal{C}$ of Boolean circuits, and we denote by $\mathcal{C}_n$ all the $\mathcal{C}$-circuits with n input gates. (Note that we use the strict syntactic notion of the number of input gates, and in particular we do not allow circuits with less than n input gates to compute functions over n bits.) For a function $B(n)$, we denote by $\mathcal{C}_n^{\leq B}$ the subclass of $\mathcal{C}_n$ consisting of all circuits in $\mathcal{C}_n$ that reject at most $B(n)$ of their inputs, and we denote $\mathcal{C}^{\leq B} = \cup_{n\in\mathbb{N}} \mathcal{C}_n^{\leq B}$. Then, we have that:

Theorem 5.4 (black-box quantified derandomization implies circuit lower bounds)**.** *Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a circuit class. For $\ell(n) < n$ and $B(n) \geq 2^{\ell(n)}$, assume that there exists a $2^{O(\ell(n))}$-time computable hitting-set generator with seed length ℓ for $\mathcal{C}_n^{\leq B}$. Then, there exists $L \in \mathcal{DTIME}[2^{O(\ell(n))}]$ that cannot be decided by $\mathcal{C}_n$.*

Proof. On n-bit inputs, the hard problem L_n is the indicator function of $\{0,1\}^n \setminus$

[20] For a very recent hardness vs randomness result that avoids PRGs and circuit lower bounds, and instead relates non-black-box algorithms for general derandomization to a new type of uniform lower bounds (i.e., lower bounds for uniform machines on almost all inputs), see [CT21a].

$\left\{H(s) : s \in \{0,1\}^{\ell(n)}\right\}$, where H is the hitting-set generator; that is, $x \in L$ iff there does not exist $s \in \{0,1\}^{\ell(n)}$ such that $x = H(s)$. Note that any $C \in \mathcal{C}_n$ that decides L_n rejects at most $2^{\ell(n)} \leq B(n)$ inputs, and thus $C \in \mathcal{C}_n^{\leq B}$. However, this means that H is a HSG for C, and thus $C(H(s)) = 1$ for some $s \in \{0,1\}^{\ell(n)}$, a contradiction. ■

Remark 5.5. One important subtlety in Theorem 5.4 makes the result weaker than the analogous result for general derandomization. In Theorem 5.4, a shorter seed yields a hard problem in a smaller complexity class (since the hard problem is in $\mathcal{DTIME}[2^{\ell(n)}]$). When starting from a PRG for general derandomization (i.e., for distinguishers that aren't necessarily biased) and using an argument analogous to the proof of Theorem 5.4, we can also use a shorter seed in order to get a lower bound for *larger* circuits. Unfortunately, when starting from a PRG for quantified derandomization (i.e., for biased distinguishers) we do *not* know how to trade off a shorter seed for a lower bound for larger circuit.[21]

6 Quantified derandomization of specific circuit classes

In this section I elaborate on Section 2.3. First, Section 6.1 includes proof of Theorem 2.6, which asserts that any better-than-brute-force algorithm for quantified derandomization of general circuits implies breakthrough circuit lower bounds. Then, Sections 6.2, 6.3 and 6.4 elaborate on the results regarding $\mathcal{AC}^0$ circuits, $\mathcal{TC}^0$ circuits, and probabilistic formulas, respectively.

6.1 General Boolean circuits

We are interested in Boolean circuits of fan-in two over the De Morgan basis with no structural restrictions (e.g., no restrictions on the depth or on the fan-out of internal gates). The algorithmic method for circuit lower bounds, which was pioneered by Ryan Williams and extensively developed in the last decade (see, e.g., [Wil13; Wil11; SW13; BSV14; MW18; CW19; Che19; CR20; CLW20], following early results such as [IKW02; KI04]), asserts that a better-than-brute-force algorithm for CAPP implies superpolynomial lower bounds. Specifically, as shown in [Wil13], if for all $k \in \mathbb{N}$ there exists

[21] To see why, recall that the standard approach to obtain lower bounds for larger circuits is to define a hard problem over $2\ell(n)$ input bits rather than over n input bits, and argue that this problem is hard for circuits of size n that is large as a function of the input length (i.e., the hard problem requires deciding whether the input is a prefix of $H(s)$ for some $s \in \{0,1\}^{\ell}$). To do so we consider distinguishers with n input gates that are sensitive only to the first $2\ell(n)$ input gates, identify this class with circuits over $2\ell(n)$ input gates of size n, and argue that neither of the two classes can avoid the output-set of the HSG. The gap between the two classes can be bridged using padding: If there is a distinguisher with $2\ell(n)$ input gates, we can add dummy input gates and obtain a distinguisher with n input gates. However, for quantified derandomization this gap seems meaningful, and we cannot bridge it using the padding argument (since adding dummy input gates significantly increases the number of exceptional inputs).

an algorithm for CAPP of n-bit circuits of size n^k that runs in time $2^n/n^{\omega(1)}$, then $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$, which would be a major breakthrough in complexity theory.[22]

As mentioned in Section 1.2, CAPP efficiently reduces to QD_B, and therefore algorithms for QD_B yield corresponding algorithms for CAPP. Combined with the algorithmic method above, it follows that sufficiently fast algorithms for QD_B imply circuit lower bounds for $\mathcal{NEXP}$. Actually, as shown in the following result, for *any value of* $B = B(n)$, *essentially any improvement over the brute-force algorithm* for QD_B implies that $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$. As pointed out by Ryan Williams, this generalizes the result in [Wil13], which is the special case with $B(n) = 2^n/3$.

Theorem 6.1 (beating the brute-force quantified derandomization implies circuit lower bounds)**.** *Suppose that for some $B(n) < 2^n$ and all $k \in \mathbb{N}$ there exists a non-deterministic machine M that gets as input an n-bit circuit C of size n^k, runs in time $B(n) \cdot (\log(B(n)))^{-\omega(1)}$, accepts if C accepts all its inputs, and rejects if C rejects all but at most $B(n)$ of its inputs. Then $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$.*

An even better result than Theorem 6.1 would deduce the conclusion from a running time of $\tilde{O}(n^k) \cdot B \cdot (\log B)^{-\omega(1)}$ rather than $B \cdot (\log B)^{-\omega(1)}$ (since the naive brute-force algorithm runs in time $\tilde{O}(n^k) \cdot B$). For $B(n) \geq 2^{n^{\Omega(1)}}$ the difference between the two is immaterial (because in this case $\log(B(n))^{-\omega(1)} = n^{-\omega(1)}$), whereas for $B(n) = 2^{n^{o(1)}}$ the proof below shows that the conclusion follows even if the algorithm only handles circuits of a fixed polynomial size n^k (where $k \in \mathbb{N}$ is a universal constant), and so the gap between the two formulations is a fixed universal polynomial.

Proof of Theorem 6.1. We will rely on the result of Williams [Wil13], which asserts that if for all $k_0 \in \mathbb{N}$ there exists a non-deterministic machine solving $\mathsf{CAPP}_{1,\frac{1}{2}}$ for m-bit circuits of size m^{k_0} in time $2^m/m^{\omega(1)}$ then $\mathcal{NEXP} \not\subset \mathcal{P}/\text{poly}$.

We are given a circuit $C_0 \colon \{0,1\}^m \to \{0,1\}$ of size m^{k_0} that either accepts all its inputs, or rejects all but at most $2^m/2$ of its inputs. We will use the disperser $\mathsf{Disp} \colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ from [TSUZ07, Theorem 1.4] for error-reduction, instantiated with input length n such that $m = \log(B(n))$ (i.e., $n = B^{-1}(2^m)$), error $\epsilon = .01$, min-entropy $k = \log(B(n))$, and seed length $O(\log(n))$. Then, the circuit $C \colon \{0,1\}^n \to \{0,1\}$ defined by $C(z) = \bigwedge_{s \in \{0,1\}^\ell} C_0(\mathsf{Disp}(z,s))$ satisfies the following:

1. The circuit size is $2^\ell \cdot T_{\mathsf{Disp}}(n) \cdot m^{k_0} \leq n^c \cdot m^{k_0}$, where T_{Disp} is the polynomial time complexity of Disp and $c \in \mathbb{N}$ is a universal constant.

2. If C_0 accepts all its inputs then C accepts all of its inputs, and if C_0 rejects all but at most $2^m/2$ of its inputs then C rejects all but at most $B(n)$ of its inputs.

Using the hypothesized non-deterministic machine for QD_B we can distinguish between the two latter cases in time $B(n) \cdot (\log(B(n)))^{-\omega(1)} = 2^m/m^{\omega(1)}$. ∎

[22] This result also scales down to weaker circuit classes, but for now we are concerned only with general circuits. Also, this result still holds if the algorithm is non-deterministic and only solves $\mathsf{CAPP}_{1,\frac{1}{2}}$.

Recent results regarding the algorithmic method, in particular that of Murray and Williams [MW18], allow to trade off a stronger hypothesis – namely, a mildly faster algorithm for CAPP – for a stronger concluded circuit lower bound. This tradeoff extends to the setting of algorithms for QD_B. The following result asserts that in the setting of a polynomial number $B(n) = \mathrm{poly}(n)$ of exceptional inputs, an algorithm for QD_B running in time $B^{1-\epsilon}$ (rather than $B \cdot (\log B)^{-\omega(1)}$ as in Theorem 6.1) yields stronger circuit lower bounds.

Theorem 6.2 (beating the brute-force quantified derandomization implies circuit lower bounds)**.** *There exists a universal constant $c \in \mathbb{N}$ such that the following holds. Suppose that for some $B(n) = \mathrm{poly}(n)$ there exists $\epsilon > 0$ and a non-deterministic machine M that gets as input an n-bit circuit C of size n^c, runs in time $B(n)^{1-\epsilon}$, accepts if C accepts all its inputs, and rejects if C rejects all but at most $B(n)$ of its inputs. Then, for all $k \in \mathbb{N}$ it holds that $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$.*

Proof. The proof is similar to the proof of Theorem 6.1, except that we use the result of Murray and Williams [MW18] instead of that of [Wil13]: They proved that if for some $\delta \in (0,1)$ there exists a non-deterministic machine solving $\mathsf{CAPP}_{1,\frac{1}{2}}$ for m-bit circuits of size $2^{\delta \cdot m}$ in time $2^{(1-\delta)\cdot m}$, then for all $k \in \mathbb{N}$ it holds that $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$.

Let $B(n) = n^a$ and let $\delta = \delta(\epsilon, a)$ be sufficiently small. We are given a circuit $C_0 \colon \{0,1\}^m \to \{0,1\}$ of size $2^{\delta \cdot m}$, and we reduce its error using the disperser of [TSUZ07] with the same parameters as in the proof of Theorem 6.1. The resulting circuit C is of size $2^\ell \cdot n^{k_1} \cdot 2^{\delta \cdot m}$, which is bounded by n^c for a universal $c > k_1$ (since $m = \log(B(n)) = a \cdot \log(n)$ and $\delta = \delta(\epsilon, a) > 0$ is sufficiently small). The hypothesized algorithm for QD_B of C runs in time $B(n)^{1-\epsilon} = 2^{(1-\epsilon)\cdot m} < 2^{(1-\delta)\cdot m}$, where the inequality relies on $\delta > 0$ being sufficiently small. ■

In contrast to the analogous results for CAPP, we do not know how to smoothly scale Theorems 6.1 and 6.2 to weaker circuit classes (for comparison see, e.g., [SW13; BSV14; CW19]). The reason is that the first step of reducing CAPP to QD_B does not work as well in weak circuit classes, as such classes are (provably) incapable of computing optimal extractors and dispersers (see, e.g., the results in Section 7, or [Vio05; GVW15]). Nevertheless, in high-level, the results for "threshold values" of weak circuits (as described in Section 2.3) still rely on the same proof approach as in Theorems 6.1 and 6.2, while replacing the error-reduction step with a weaker error-reduction step, which yields weaker results than in the case of general circuits.

6.2 Constant-depth circuits

In this section we consider $\mathcal{AC}^0$ circuits, which are circuit families of constant depth and unbounded fan-in over the De Morgan basis. The best known lower bound for $\mathcal{AC}^0$ by Håstad [Hås87; Hås14] asserts that parity requires depth-d circuits of size $2^{\Omega(n^{1/(d-1)})}$. This lower bound remained unimproved for 35 years, and we know that even relatively mild improvements of the exponent in this lower bound (e.g., showing

a lower bound of $2^{n^{3.01/(d-1)}}$) would be a major breakthrough in complexity theory (see, e.g., [Tel20] for a discussion of the latter statement).

The best currently known PRG for $\mathcal{AC}^0$ has seed length $\tilde{O}(\log(s)^d \cdot \log(n))$ for constant error, where s is the circuit size; this PRG was first constructed by Trevisan and Xue [TX13], and its analysis was improved by Tal [Tal17b] and again, very recently, by Kelley [Kel21] (who obtained the seed length stated above). An improvement in the exponent d of the $\log(s)$ term (e.g., to $\log(s)^{0.99 \cdot d}$) would finally yield lower bounds for $\mathcal{AC}^0$ circuits that are larger than in Håstad's long-standing lower bound (see [TX13, "Barriers to Further Progress"] for an explanation).

6.2.1 Tractable values

In Goldreich and Wigderson's [GW14] original paper, they proved that there exists a polynomial-time computable .01-PRG with seed length $O(\log(n))$ for B-biased depth-d circuits, where $B(n) = 2^{n^\epsilon}$ and $\epsilon > 0$ can be arbitrarily small.

Theorem 6.3 (quantified derandomization of $\mathcal{AC}^0$ with subexponential B; see [GW14, Theorem 1.3 in the Full Version])**.** *For any $d \in \mathbb{N}$ and $k \in \mathbb{N}$ and $\epsilon > 0$, there exists a polynomial-time computable .01-PRG with seed length $O(\log(n))$ for B-biased depth-d circuits of size n^k, where $B(n) = 2^{n^\epsilon}$.* [23]

Their result was later extended in [Tel19] to support a larger number $B(n)$ of exceptional inputs, at the expense of a longer seed. The extreme "high-end" point of this tradeoff supports $B(n) = 2^{n/\mathrm{polylog}(n)}$ with seed length $\tilde{O}(\log^3(n))$. The main advantage in the latter result is that *the seed length does not depend on the depth d*; as mentioned above, a PRG with such seed length for $\mathcal{AC}^0$ circuits that are not necessarily biased would be a major breakthrough. (In fact, as we'll see below, even a PRG with such seed length for B-biased circuits where B is larger would yield a breakthrough.)

Theorem 6.4 (quantified derandomization of $\mathcal{AC}^0$; see [Tel19, Theorem 5.16])**.** *For any $d \in \mathbb{N}$ and $k \in \mathbb{N}$ and $t(n) \leq 2k \cdot \log(n)$, there exists a polynomial-time computable .01-PRG with seed length $\tilde{O}(t(n)^2 \cdot \log(n))$ for B-biased depth-d circuits of size n^k, where $B(n) = 2^{n^{1-1/\Omega(t)}/t^{d-2}}$.*

Corollary 6.5 (quantified derandomization of $\mathcal{AC}^0$, a special case)**.** *For any two constants $d \geq 2$ and $k \in \mathbb{N}$ there exists a .01-PRG with seed length $\tilde{O}(\log^3(n))$ for the class of B-biased distinguishers of depth d and size n^k, where $B(n) = 2^{\Omega(n/\log^{d-2}(n))}$.*

Theorem 6.4 is not explicitly stated in [Tel19, Theorem 5.16], which only states a construction of a HSG with an identical seed length. However, the construction in [Tel19] essentially already gives a PRG, and I now explain the few needed changes.

[23] The original result states the existence of a $(1/3)$-PRG, but the stronger statement follows immediately from their proof (see [GW14, Theorem 3.2 in the Full Version]).

Proof sketch for Theorem 6.4. The generator in [Tel19] works in two steps. In the first step the generator outputs a restriction ρ such that for any circuit C over n input bits of depth d and size n^k, with probability $1 - 2^{-10}$ the restriction keeps $\log(B(n)) + 2$ variables alive and $C\restriction_\rho$ can be $(1/2)$-approximated by a depth-2 formula C' of size $\mathrm{poly}(n)$. In the second step the generator uses a PRG for depth-2 circuits to approximate the acceptance probability of C'.

The goal in the original proof was to find one satisfying assignment for C that accepts almost all of its inputs, and therefore the original proof was satisfied with the statements that C' is $(1/2)$-close to $C\restriction_\rho$ and that $|\rho^{-1}(\star)| \geq \log(B(n)) + 2$. (The original proof also argued that C' is lower-sandwiching, but we will not need this claim.) For our purposes we need the following two stronger statements:

1. Replace the approximation factor of $1/2$ by an approximation factor of 2^{-10}.
2. Replace the number of living variables $\log(B(n)) + 2$ by $\log(B(n)) + 10$ (such that the size of the corresponding subcube is at least $2^{10} \cdot B(n)$).

The first statement above follows by modifying the constant 2 in the denominator of the parameter β in the original proof to the constant 2^{10}; this change doesn't affect the proof in any meaningful way (to see that the statement above follows, note that this modifies the claimed approximation factor in Claim 5.18 from $1/2d$ to $2^{-10} \cdot d$). The second statement follows immediately from the original proof.[24]

Given these two statements, denoting $\epsilon = 2^{-10}$, with probability at least $1 - \epsilon$ we have that $C\restriction_\rho$ has acceptance probability either at most ϵ or at least $1 - \epsilon$, and therefore the acceptance probability of C' is either at most 2ϵ or at least $1 - 2\epsilon$. Continuing just like in the original proof, we use an ϵ-PRG for C', which is a (4ϵ)-PRG for C (where the constant 4 accounts for the error ϵ in choosing a restriction and for the error of 3ϵ in the approximation of the acceptance probability of C by the PRG for C'). ■

6.2.2 Threshold values

The threshold result for $\mathcal{AC}^0$ circuits refers to $B(n)$ that is only slightly larger than $B(n) = 2^{n/\log^{d-2}(n)}$ in Corollary 6.5. Specifically, it refers to $B(n) = 2^{n/\log^{d-d_0}}$ for $d_0 > 11$ (instead of $d_0 = 2$). I'll first state the general form of this result, which reduces CAPP for depth-d_0 circuits to QD_B for depth-d circuits (where $d > d_0$), and then state two interesting special cases. The result is from [Tel19], with the main technical tool constructed by Cheng and Li [CL16].

Theorem 6.6 (threshold result for quantified derandomization of $\mathcal{AC}^0$; see [Tel19, Theorem 1.1], following [CL16])**.** *For any $d_0 \geq 2$ and $d \geq d_0 + 11$ it holds that* CAPP *for depth-d_0 circuits of linear size on n_0 input bits reduces, in deterministic polynomial time, to* QD_B *for depth-d circuits on $n = O(n_0^{3600})$ input bits of size $n^{O_d(1)}$, where $B(n) = 2^{n/\log^{d-d_0-11}(n)}$.*

[24] To see this, note that we did not specify the constant hidden inside the Ω-notation in the expression $B(n) = 2^{n^{1-1/\Omega(t)}/t^{d-2}}$. Thus, the difference between $\log(B(n)) + 2$ and $\log(B(n)) + 10$ (or $c \cdot \log(B(n))$ for any universal constant $c > 1$) is immaterial.

Theorem 6.6 is slightly stronger than the result stated in [Tel19], both in terms of parameters and since it reduces CAPP to QD_B (rather than reducing general derandomization with *one-sided error* to quantified derandomization). Let me thus sketch the proof, which is still very similar to the original one.

Proof sketch for Theorem 6.6. Given a linear-sized circuit $C\colon \{0,1\}^{n_0} \to \{0,1\}$ of depth d_0 as input for CAPP, we print a circuit $C'\colon \{0,1\}^n \to \{0,1\}$ as input for QD_B, where $C'(z) = \mathsf{ApxMaj}\{C(\mathsf{Ext}(z,s))\}_s$ and ApxMaj is an approximate majority function.[25] The main thing to verify is the complexity of C, given appropriate constructions for Ext and for ApxMaj.

The extractor used originally in [Tel19] was from [CL16, Theorem 1.3], and in this proof we use another extractor construction from the same work [CL16, Theorem 4.11], which has a smaller depth overhead. Specifically, for $a = d - d_0 - 10 \geq 1$, the latter extractor has depth $a+7$, size $n^{O_a(1)}$, input length $n = O(n_0^{3600})$, seed length $O_a(\log(n))$, and supports min-entropy $\Theta(n/\log^a(n))$. Also, we use Viola's [Vio09a] uniform construction of an ApxMaj function computable by depth-3 circuits of size $n^{k'}$, for some universal constant $k' \in \mathbb{N}$. Thus, for some $k = k(a) \leq k(d)$, the final depth of C' is $a + 7 + d_0 + 3 = d$, and its size is $n^k + n^k \cdot |C| + (n^k)^{k'} = n^{O_d(1)}$. ■

In the following two corollaries of Theorem 6.6 we assume that there is an algorithm for quantified derandomization of $\mathcal{AC}^0$ whose running time is identical to that of the algorithm in Theorem 6.5, but that can handle larger values of $B(n)$. The first corollary refers to a $B(n)$ for which such an algorithm would yield a better CAPP algorithm for $\mathcal{AC}^0$ (i.e., than the best currently known CAPP algorithm); the second corollary refers to a (slightly larger) $B(n)$ for which such an algorithm would yield new *circuit lower bounds* for $\mathcal{AC}^0$, for circuits that are larger than in [Hås87].

Corollary 6.7 (threshold result for quantified derandomization of $\mathcal{AC}^0$; a special case)**.** *Assume that for some $d \geq 14$ and a sufficiently large $k = k_d \in \mathbb{N}$ it holds that QD_B for depth-d circuits of size n^k can be solved in time $2^{\tilde{O}(\log^3(n))}$, where $B(n) = 2^{n/\log^{d-14}(n)}$. Then,* CAPP *for depth-3 circuits of linear size can be solved in time $2^{\tilde{O}(\log^3(n))}$.*

Corollary 6.8 (threshold result for quantified derandomization of $\mathcal{AC}^0$; a special case)**.** *Assume that for some $d \geq 18$ and a sufficiently large $k = k_d \in \mathbb{N}$ it holds that QD_B for depth-d circuits of size n^k can be solved in time $2^{\tilde{O}(\log^3(n))}$, where $B(n) = 2^{n/\log^{d-18}(n)}$. Then, there is a problem in $\mathcal{E}^{\mathcal{NP}}$ that cannot be decided by $\mathcal{AC}^0$ circuits of depth 5 and size $2^{\Omega(n^{2/7})}$.*

Proof. For $d_0 = 7$, let C be a depth-d_0 circuit over n_0 input bits and of size $n_1 = 2^{n_0^{2/d_0}}$. We pad its number of inputs to n_1, to obtain a linear-sized depth-d_0 circuit C'. Then we apply the reduction in Theorem 6.6 to obtain a depth-d circuit C'' on $n = O(n_1^{3600})$

[25] That is, it can be any function that outputs 1 when the relative Hamming weight of its input is at least 2/3 and outputs s0 when the relative Hamming weight of its input is at most 1/3.

input bits and of size n^{k_d} with $B(n) = 2^{n/\log^{d-d_0-11}(n)}$ exceptional inputs, where k_d is a constant that depends on d. By our assumption, we can solve QD_B on C'' in time

$$2^{\tilde{O}(\log^3(n))} = 2^{\tilde{O}(n_0^{6/d_0})} = 2^{o(n_0)} ,$$

and hence we can solve CAPP for C in time $2^{o(n_0)}$. Using the results of Ben-Sasson and Viola [BSV14, Theorem 1.4] following [Wil14], it follows that there is a problem in $\mathcal{E}^{\mathcal{NP}}$ that does not have $\mathcal{AC}^0$ circuits on n_0 input bits of depth $d_0 - 2 = 5$ and size $2^{n_0^{2/d_0}}/n_0^{O(1)} > 2^{c_0 \cdot n_0^{2/d_0}} = 2^{c_0 \cdot n_0^{2/7}}$, for some constant $c_0 < 1$. ∎

Remark 6.9. I suspect that it might be possible to improve the parameters in Corollary 6.8 using techniques similar to the ones introduced by Chen and Williams [CW19]. In particular, the current proof seems wasteful, in the sense that the overhead between the class of circuits to which we apply a CAPP algorithm and the class of circuits for which we deduce lower bounds (i.e., two layers of depth) seems too large.[26]

6.3 Constant-depth circuits with threshold gates

In this section we consider LTF circuits, which are circuit families of constant depth and unbounded fan-in gates that can compute any linear threshold function (i.e., any function of the form $\Phi(x_1, ..., x_n) = \mathrm{sgn}(\sum_{i\in[n]} w_i \cdot x_i - \theta)$, where $w_1, ..., w_n, \theta \in \mathbb{R}$). This class contains $\mathcal{TC}^0$, which is the special case in which all gates compute the majority function.[27] We measure the size of the circuit as its number of *wires*.

The best known lower bound for $\mathcal{TC}^0$ by Impagliazzo, Paturi, and Saks [IPS97] asserts that parity requires depth-d LTF circuits with $n^{1+2.42^{-d}}$ wires. This lower bound was extended several years ago to an average-case lower bound for LTF circuits of depth d with $n^{1+c^{-d}}$ wires, for some constant $c \gg 2.42$ (see [CSS16]); and very recently a PRG with seed length $n^{1-\Omega(1)}$ was constructed for LTF circuits of depth d with $n^{1+(c')^{-d}}$ wires, for some $c' > c$ (see [HHT+21]).

Proving lower bounds that hold for larger circuits (or improving the PRG to work for larger circuits), even just for circuits with $n^{c_0^{-d}}$ wires for some small $c_0 > 1$, is one of the most prominent current frontiers in circuit complexity (see, e.g., [Aar16; CT19]).

6.3.1 Tractable values

The following algorithm for quantified derandomization of LTF circuits works for circuits with $n^{1+c^{-d}}$ wires and $B(n) = 2^{n^{1-(c')^{-d}}}$ exceptional inputs, for some constants

[26] The current proof uses the PCP of [BSV14], which adds two layers and a multiplicative polynomial size overhead, but does not exploit the fact that one of these two layers has gates of fan-in three. Moreover, the techniques from [CW19] add only one layer of constant fan-in.

[27] In some texts $\mathcal{TC}^0$ is actually defined using LTF gates rather than majority gates. The two definitions are equivalent up to an overhead of a single layer and polynomially many gates (see [GHR92; GK98]), but since we will care about precise size bounds I will distinguish between the two cases.

$c, c' > 2$. Indeed, up to the particular constant c this is essentially the same subclass of LTF circuits for which we already have a PRG [HHT+21] (i.e., for which we know of a non-trivial *general* derandomization). However, the known PRG has seed length $n^{1-\Omega(1)}$ and therefore enumerating over its seeds yields general derandomization with running time $2^{n^{1-\Omega(1)}}$; in contrast, the quantified derandomization algorithm below runs in time that is almost polynomial (i.e., in time $n^{O(\log\log^2(n))}$.

Theorem 6.10 (quantified derandomization of LTF circuits; see [Tel18, Theorem 5.1])**.** *Let $d \geq 1$, let $\epsilon > 0$, and let $\delta = d \cdot 30^{d-1} \cdot \epsilon$. Then, there exists a deterministic algorithm that solves QD_B for depth-d LTF circuits with $n^{1+\epsilon}$ wires in time $n^{O(\log\log^2(n))}$, where $B(n) = \frac{1}{10} \cdot 2^{n^{1-\delta}}$.*

The following special case of Theorem 6.10 will be convenient for comparison with the threshold values below.

Corollary 6.11 (quantified derandomization of LTF circuits, a special case)**.** *For any $d \geq 1$ there exists a deterministic algorithm that solves QD_B for depth-d LTF circuits with $n^{1+60^{-d}}$ wires in time $n^{O(\log\log^2(n))}$, where $B(n) = 2^{n^{1-1.61^{-d}}}$.*

Proof. We use Theorem 6.10 with $\epsilon = 60^{-d}$, and rely on the fact that $\delta = (d/30) \cdot 2^{-(d-1)} < 1.61^{-d}$ for all d, which implies that $B(n) = \frac{1}{10} \cdot 2^{n^{1-\delta}} > 2^{n^{1-1.61^{-d}}}$. ■

In addition, a quantified derandomization result for a stronger class of threshold circuits was proved by Kabanets and Lu [KL18] . Specifically, they considered degree-Δ PTF circuits, in which each gate can compute a function of the form $\Phi(x_1, ..., x_n) = \mathrm{sgn}(p(x_1, ..., x_n))$, where p is a real polynomial of degree at most Δ. (Note that LTF circuits are the special case where $\Delta = 1$.) They showed a quantified derandomization algorithm that runs in time $\exp(\log(n)^{O(\Delta)^2})$ and can handle $B(n) = 2^{n^{1-7/\sqrt{c}}}$ exceptional inputs, where c is the constant in the size bound $n^{1+c^{-d}}$.

Theorem 6.12 (quantified derandomization of PTF circuits; see [KL18, Theorem 1.3])**.** *For any three constants $c \geq 122$ and $\Delta, d \in \mathbb{N}$ there exists a deterministic algorithm that solves QD_B for depth-d PTF circuits of degree Δ with $n^{1+c^{-d}}$ wires, whose running time is $2^{\log(n)^{O(\Delta^2)}}$ and where $B(n) = 2^{n^{1-7/\sqrt{c}}}$.*

Interestingly, the algorithms in Theorems 6.10 and 6.12 are not PRGs for biased distinguishers (in contrast to the quantified derandomization algorithms for $\mathcal{AC}^0$), but rather *non-black-box* algorithms that get as input a description of the circuit, rely on the information in that description, and decide whether the circuit accepts almost all of its inputs or rejects almost all of its inputs. See futher discussion of this fact in Section 7.

6.3.2 Threshold values

The first threshold result for $\mathcal{TC}^0$ circuits in [Tel18] was later on superseded by the result below, which was proved by Chen and the current author [CT19]. The result

below refers to $B(n)$ of the same form as in Theorem 6.10, but to circuits that are *slightly larger*. Specifically, the algorithm in Theorem 6.10 can handle circuits with $n^{1+c^{-d}}$ wires, for $c > 30$, whereas the result below refers to circuits with $n^{1+c^{-d}}$ wires for c that is smaller than the golden ratio. Moreover, the threshold result below holds also for $\mathcal{TC}^0$ circuits (i.e., circuits with MAJ gates) rather than only for LTF circuits.

Theorem 6.13 (threshold result for quantified derandomization of $\mathcal{TC}^0$; see [CT19, Theorem 44])**.** *For any $d_0, k \in \mathbb{N}$ and $d \geq d_0 + 7$ and $c < \frac{1+\sqrt{5}}{2}$ it holds that* CAPP *for depth-d_0* LTF *circuits on n_0 input bits with n_0^k wires reduces, in deterministic polynomial time, to QD_B for depth-d* LTF *circuits on $n = n_0^{O_{k,d}(1)}$ input bits with $n^{1+c^{-d}}$ wires, where $B(n) = 2^{n^{c^{-d}}}$.*

For convenience, we state this result with the particular value $c = 1.61 < \frac{1+\sqrt{5}}{2}$, and also state the circuit lower bound implications of a potential algorithm for quantified derandomization of $\mathcal{TC}^0$.

Corollary 6.14 (threshold result for quantified derandomization of $\mathcal{TC}^0$; a special case)**.** *Assume that for every $d \geq 9$ and every $\epsilon > 0$ there exists an algorithm for QD_B of depth-d* LTF *circuits with $n^{1+1.61^{-d}}$ wires that runs in time 2^{n^ϵ}, where $B(n) = 2^{n^{1.61^{-d}}}$. Then, there exists a problem in $\mathcal{NEXP}$ that cannot be decided by polynomial-sized $\mathcal{TC}^0$ circuits.*

We stress that the allowed running time for the algorithm in Corollary 6.14 is considerably larger than the running time of the known algorithm from Theorem 6.10 (i.e., the allowed runtime is 2^{n^ϵ} whereas the known algorithm runs in time $n^{\mathrm{polyloglog}(n)}$).

6.4 De Morgan formulas

We now consider formulas of fan-in two over the De Morgan basis; allowing these formulas to have arbitrary polynomial size yields the complexity class $\mathcal{NC}^1$. The best known lower bound for formulas, which was proved by Håstad's [Hås98] (following [Sub61; Khr71; And87; IN93; PZ93]) with subsequent log-factors improvements by Tal [Tal14; Tal17a], asserts that Andreev's function cannot be computed by formulas of size $n^3/\mathrm{polylog}(n) = n^{3-o(1)}$ (see [DM18; GTN19] for related formula lower bounds).[28] Moreover, average-case lower bounds for formulas of size $n^{3-o(1)}$ have been proved in a sequence of recent works (see [San10; KR13; IK17; Tal17a; KRT17; Bog18]).[29] Indeed, proving hardness of explicit functions for De Morgan formulas of larger size (say, arbitrary polynomial size) is a long-standing challenge, since the 1960's.

[28] The best lower bound of [Tal17a] actually holds for a variation of Andreev's function introduced in [KR13], which is called the generalized Andreev function.

[29] To be more accurate, the best average-case lower bounds from [KRT17; Bog18] assert that for any parameter $r \leq n$, formulas of size $n^{3-o(1)}/r^2$ cannot compute a corresponding function in $\mathcal{P}$ with success probability more than $1/2 + 2^{-r}$.

In terms of CAPP algorithms, a polynomial-time computable $\frac{1}{\text{poly}(n)}$-PRG with seed length $s^{1/3} \cdot 2^{O(\log^{2/3}(s))} = s^{1/3+o(1)}$ was constructed by Impagliazzo, Meka, and Zuckerman [IMZ12] (note that this PRG can fool formulas of size $n^{3-o(1)}$ with non-trivial seed length, and hence yields lower bounds for formulas of size $n^{3-o(1)}$). An ϵ-PRG that can support smaller error $\epsilon > 0$ was very recently constructed by Hatami *et al.* [HHT+21]; the seed length of the latter PRG is $s^{3-o(1)} \cdot \text{polylog}(n/\epsilon)$.

Interestingly, both the tractable values and the threshold values that we state below refer to formulas of size that is *smaller than the formula size in the known lower bounds*: Specifically, they refer to formulas of sub-quadratic size, rather than to formulas of sub-cubic size. Nevertheless, there is still similarity to the case of $\mathcal{TC}^0$ from Section 6.3, in the sense that the quantified derandomization results refer to circuits/formulas of approximately the size needed to compute the *parity function*.

Probabilistic formulas. The results stated below were proved by Chen, Jin, and Williams [CJW20]. To make the tractable values and the threshold values as close as possible, they considered the stronger computational model of probabilistic formulas, which are arbitrary distributions $\mathbf{F}$ over formulas.[30]

For any fixed input $x \in \{0,1\}^n$, if there exists $\sigma \in \{0,1\}$ such that $\Pr[\mathbf{F}(x) = \sigma] \geq 2/3$ then the value of $\mathbf{F}$ at x is σ, and we say that $\mathbf{F}$ accepts x or rejects x, accordingly; otherwise, if no such σ exists (e.g., $\Pr[\mathbf{F}(x) = 1] = 1/2$), we say that $\mathbf{F}$ is undecided at x. A B-biased probabilistic formula $\mathbf{F}$ on n input bits either accepts all but $B(n)$ of its inputs or rejects all but $B(n)$ of its inputs (in both cases, the probabilistic formula may be undecided on the $B(n)$ minority inputs). For $B(n) = o(2^n)$, an ϵ-PRG for B-biased probabilistic formulas is an algorithm G such that

$$\left| \Pr[\mathbf{F}(G(1^n, \mathbf{s})) = 1] - \Pr[\mathbf{F}(\mathbf{u}_n) = 1] \right| \leq \epsilon \,,$$

where the probabilities in both expressions above are taken both over $\mathbf{F}$ and over a choice of input for $\mathbf{F}$ (i.e., over $G(1^n, \mathbf{s})$ or $\mathbf{u}_n$). [31]

6.4.1 Tractable values

The following quantified derandomization algorithm from [CJW20] runs in polynomial time and works for formulas of near-quadratic size; more accurately, it allows to trade off the formula size for the number of exceptional inputs.

[30] They referred to this model as *generalized* probabilistic formulas, where the generalization lies in the fact that the probabilistic formula is allowed to be undecided on some inputs (i.e., it decides a promise problem rather than a language).

[31] Indeed, this definition allows $\mathbf{F}$ to be undecided on some strings in the output-set of the PRG, as long as $\mathbf{F}(G(\mathbf{s}))$ behaves similarly to $\mathbf{F}(\mathbf{u}_n)$. Nevertheless, in cases where the probabilistic formula satisfies the "$\mathcal{BPP}$ promise" (i.e., either accepts or rejects any input), the known PRG construction satisfies a stronger property. See Remark 6.17 for further details.

Theorem 6.15 (quantified derandomization of De Morgan formulas; see [CJW20, Theorem 1.9])**.** *There exists a universal constant $c > 1$ such that the following holds. For any $c \leq s(n) \leq n/20$ there is a $(4/10)$-PRG for B-biased generalized probabilistic formulas of size $n^{2-c/\log\log(n)}/s(n)^2$, with seed length $O(\log(n))$ and polynomial running time, where $B(n) = 2^{s(n)}$.*

Corollary 6.16 (quantified derandomization of De Morgan formulas; a special case)**.** *For any $\epsilon > 0$ there is a $(4/10)$-PRG for B-biased generalized probabilistic formulas of size $n^{2-2\epsilon-o(1)}$ with seed length $O(\log(n))$ and polynomial running time, where $B(n) = 2^{n^\epsilon}$.*

The statement in [CJW20] only asserts the existence of a HSG (for B-biased probabilistic formulas that accept almost all of their inputs), but their proof can be adapted to yield a PRG for B-biased formulas; I explain how below. The obtained PRG has a relatively large error of $4/10$, but since this error is bounded away from $1/2$, when $B(n) = o(2^n)$ we can still use this PRG to distinguish between B-biased formulas that accept almost all of their inputs and ones that reject almost all of their inputs.

Proof of Theorem 6.15. Now, let $\mathbf{F}$ be a n-bit probabilistic formula of size at most $n^{2-c/2\log\log(n)}/s(n)^2$, and assume that accepts all but $B(n)$ of its inputs. (The proof for the case that $\mathbf{F}$ rejects all but $B(n)$ of its inputs is symmetric.) For a sufficiently large constant $k > 1$, consider the n-bit probabilistic formula $\mathbf{F}'$ such that

$$\mathbf{F}'(x) = \mathsf{THR}_{.65}\left(\mathbf{F}^{(1)}(x), \mathbf{F}^{(2)}(x), ..., \mathbf{F}^{(k)}(x)\right) ,$$

where the $\mathbf{F}^{(i)}$'s are independent RVs each choosing $F \sim \mathbf{F}$, and $\mathsf{THR}_{.65}$ is the function that outputs 1 if and only if at least 0.65 of its inputs are 1.

The probabilistic formula $\mathbf{F}'$ exists only in our analysis, and the PRG will not need to actually construct it. We state a few properties of $\mathbf{F}'$:

1. For any x that $\mathbf{F}$ accepts, $\Pr[\mathbf{F}'(x) = 1] \geq 1 - .001$.
2. For any x that $\mathbf{F}'$ accepts, $\Pr[\mathbf{F}(x) = 1] \geq 0.64$.
3. The size of $\mathbf{F}'$ is at most $n^{2-c/\log\log(n)}/s(n)^2$. (This is since the size of $\mathbf{F}'$ is larger than the size of $\mathbf{F}$ only by a constant multiplicative factor, and we bounded the size of $\mathbf{F}$ by $n^{2-c/2\log\log(n)}/s(n)^2$.)

Now, the pseudorandom restriction procedure from [CJW20, Theorem 3.2] samples a restriction that satisfies the following two properties:[32]

1. With probability at least 0.999 over $\rho \sim \boldsymbol{\rho}$ it holds that $|\rho^{-1}(\star)| \geq 10s(n)$.

[32]The statement in [CJW20] only guarantees that the probability of $|\rho^{-1}(\star)| \geq pn/2$, where $p = 20 \cdot (s(n)/n)$, is at least $2/3$. However, in their proof of Theorem 3.2, the probability that this event does not happen is bounded by $\frac{(1-p+o(1))\cdot pn + o(1)\cdot(pn)^2}{(pn/2-o(1))^2} = \frac{(1-p+o(1))\cdot s+o(s^2)}{(s/2-o(1))^2} < .001$, where we relied on the fact that $s = s(n)$ is sufficiently large. Similarly, the proof in [CJW20] bounds $\mathbb{E}[\mathcal{L}(\mathbf{F})]$ by $o(1)$ and later on only relies on the looser bound of $1/18$.

2. For every formula F' of size $n^{2-c/\log\log(n)}/s(n)^2$, with probability at least $1-o(1)$ over $\rho \sim \boldsymbol{\rho}$ it holds that $\mathbb{E}[\mathcal{L}(F'|_\rho)] = o(1)$, where $\mathcal{L}(F')$ is the number of leaves of a formula F'.

Relying on the two properties of $\boldsymbol{\rho}$, with probability at least $0.999 - o(1)$ over $\rho \sim \boldsymbol{\rho}$ and $F' \sim \mathbf{F}'$ it holds that $F'|_\rho$ is a constant function, and this constant equals the most common value of F' (since the size of the subcube that ρ keeps alive is $2^{10s(n)} > B(n)$). In this case, if we complete ρ to an n-bit string x by (say) padding with zeroes, we have that $F'(x)$ is the most common value of F'. Finally, since $\mathbb{E}_{F' \sim \mathbf{F}'}[\Pr_{x\in\{0,1\}^n}[F'(x) = 1]] = \Pr_{x\in\{0,1\}^n, F\sim\mathbf{F}}[F(x) = 1] \geq (1-o(1)) \cdot 0.999$, the probability over $F' \sim \mathbf{F}'$ that the most common value of F' is 0 is at most 0.002.

So far we described a procedure that uses $O(\log(n)) = O(\log(n))$ random coins and outputs a string $\mathbf{w}$ (i.e., $\mathbf{w}$ is the completion of $\boldsymbol{\rho}$ with zeroes) such that

$$\mathbb{E}_{w\sim\mathbf{w}}\left[\Pr_{F'\sim\mathbf{F}'}[F'(w) = 1]\right] = \Pr_{F'\sim\mathbf{F}', w\sim\mathbf{w}}\left[F'(w) = 1\right] \geq 1 - 0.001 - o(1) - 0.002 > 0.99\,,$$

which implies that $\mathbf{F}'$ accepts $\mathbf{w}$ with probability at least $1 - 1/30$. Thus, relying on the properties of $\mathbf{F}'$, we have that $\Pr_{w\sim\mathbf{w}, F\sim\mathbf{F}}[F(w) = 1] \geq (1-1/30) \cdot 0.64 \geq 0.618$. ■

Remark 6.17. The proof above actually shows that with probability at least $29/30$ over a choice of seed s for the PRG G it holds that $\Pr[\mathbf{F}(G(s)) = 1] \geq 0.64$. Thus, if the probabilistic formula $\mathbf{F}$ satisfies the "$\mathcal{BPP}$ promise" (i.e., for each input, either accepts with probability at least $2/3$ or rejects with probability at least $2/3$), then the value of $\mathbf{F}$ on a random output string of the PRG is $(1/30)$-close to the value of $\mathbf{F}$ on a uniformly random string.

6.4.2 Threshold values

In [CJW20] they complemented Theorem 6.15 by showing that $\mathsf{CAPP}_{\frac{1}{2},0}$ for polynomial-sized formulas (i.e., general derandomization for formulas with one-sided error) reduces to QD_B for formulas of near-quadratic size (indeed, it even reduces to quantified derandomization for formulas with one-sided error). To see how tight this result is, recall that the algorithm in Corollary 6.16 works for formulas of size $n^{2-2\epsilon-o(1)}$, and note that the reduction below yields formulas of size $n^{2-\epsilon+\delta}$, where $\delta > 0$ is an arbitrarily small constant, and in both results ϵ is the constant such that $B(n) = 2^{n^\epsilon}$.

Theorem 6.18 (threshold result for quantified derandomization of De Morgan formulas; see [CJW20, Theorem 1.10])**.** *For any $\epsilon \in (0,1)$ and $\delta > 1$ and $k \in \mathbb{N}$ it holds that $\mathsf{CAPP}_{\frac{1}{2},0}$ for formulas on n_0 input bits of size n_0^k reduces, in deterministic polynomial time, to QD_B for probabilistic formulas of size $n^{2-\epsilon+\delta}$ on $n = (n_0)^{O_{\epsilon,\delta}(1)}$ input bits, where $B(n) = 2^{n^\epsilon}$.*

I should clarify what "reduces to QD_B for probabilistic formulas" means here, since probabilistic formulas are arbitrary distributions (over formulas) and might not necessarily have a concise description. One interpretation, which the proof in [CJW20]

already explicitly mentions, is that a PRG for the probabilistic formulas (which does not need to get any description of the probabilistic formula as input) yields an algorithm for CAPP. Another interpretation, which is implicit in [CJW20] and made explicit in the proof sketch below, is that the reduction produces an efficient algorithm that samples the target probabilistic formula; thus, this is an explicit reduction, and it suffices to solve QD_B for probabilistic formulas by a "non-black-box" algorithm.

Proof sketch for Theorem 6.18. Given a description of a formula F_0 on n_0 input bits of size n_0^k and $\epsilon, \delta > 0$, we use the algorithm in [CJW20, Lemma 4.2] to sample in polynomial time a probabilistic formula $\mathbf{F}_1$ on $n_1 = n_0^{O_\beta}(1)$ input bits of size $n_1^{2+\beta}$, where $\beta = \epsilon\delta/4$, such that:

1. If F_0 accepts at least $1/2$ of its inputs, then $\mathbf{F}_1$ accepts all but at most $2^{n_1^\beta}$ of its inputs (on the remaining inputs $\mathbf{F}_1$ may either reject or be undefined).
2. If F_0 rejects all of its inputs, then $\mathbf{F}_1$ rejects all of its inputs.

Then, the algorithm in the proof of [CJW20, Theorem 1.10] gets a description of the machine that samples $\mathbf{F}_1$, and samples in polynomial time a probabilistic formula $\mathbf{F}$ on $n < n_1^2$ input bits of size $n^{2+\delta-\epsilon}$ such that:

1. If $\mathbf{F}_1$ accepts all but $2^{n_1^\beta}$ of its inputs, then $\mathbf{F}$ accepts all but at most 2^{n^ϵ} of its inputs (on the remaining inputs $\mathbf{F}$ may either reject or be undefined).
2. If $\mathbf{F}_1$ rejects all of its inputs, then $\mathbf{F}$ rejects all of its inputs.

Thus, the combination of the two algorithms above is an efficient procedure that samples $\mathbf{F}$, given a description of F_0. ■

As mentioned above, the target of the reduction in the proof of Theorem 6.18 is a probabilistic formula that either accepts almost all of its inputs, or rejects *all* of its inputs. Thus, Theorem 6.18 actually reduces $\mathsf{CAPP}_{\frac{1}{2},0}$ to the "one-sided error" version of QD_B, which is more relaxed and is thus potentially easier to solve.

Using the results of Murray and Williams [MW18], a polynomial-time algorithm for QD_B of formulas of near-quadratic size, as in Theorem 6.18, would yield breakthrough circuit lower bounds. That is:

Corollary 6.19 (threshold result for quantified derandomization of De Morgan formulas)**.** *Assume that for some $\epsilon \in (0,1)$ and $\delta > 1$ there exists a polynomial-time algorithm that solves QD_B for probabilistic formulas of size $n^{2-\epsilon+\delta}$, where $B(n) = 2^{n^\epsilon}$. Then, for every $k \in \mathbb{N}$ there exists a problem in $\mathcal{NP}$ that cannot be decided by formulas of size n^k.*

7 Extractors, restriction procedures, and their limitations

As explained in Section 2.4, all the results in Section 6 are proved using similar high-level techniques: Pseudorandom restriction procedures (used for algorithms demonstrating tractable values) and extractors computable by low-depth circuits (used for reductions of CAPP to QD_B demonstrating threshold values).

In this section I'll define abstract black-box versions of these techniques and discuss their limitations, as well as concrete technical challenges that these limitations point at. The specific constructions of pseudorandom restrictions and of extractors that underlie the results in Section 6 are described in Appendices B and C.

7.1 Defining the black-box techniques

The first notion will be of a black-box restriction procedure, which does not refer to the given circuit but is rather a distribution over restrictions that simplifies *every* circuit in the class, with high probability.

Definition 7.1 (distribution of simplifier sets)**.** *Let $\mathcal{C}_n$ be a class of circuits with n input gates. We say that a distribution $\mathbf{X}_n$ over subsets of $\{0,1\}^n$ is a* distribution of simplifier sets of size more than B for $\mathcal{C}_n$ *if the following holds:*

1. *Every subset in the support of $\mathbf{X}_n$ is of size more than B.*
2. *For every $C \in \mathcal{C}_n$ it holds that $\Pr_{X \sim \mathbf{X}_n}[C\restriction_X \text{ is constant}] > 1/2$.*

Let me stress that the subsets in Definition 7.1 are *not* necessarily subcubes, but may be arbitrary subsets of $\{0,1\}^n$. This is why I refer to them as "simplifier sets" rather than as "restrictions". (As a demonstration of the value in this generalization, let me note that the algorithm for QD_B in Theorem 2.11 finds a large *affine subspace* on which the circuit simplifies; see [Tel19, Section 6] for details.)

The second notion refers to samplers (equivalently, to extractors) computable in restricted circuit classes. The reason that this notion is black-box is that we require the sampler to sample *every possible subset* approximately correctly. This is an "overkill", since in our proof we will use the sampler to reduce the error of a single circuit C of bounded size whose description is explicitly given to us, and thus we only need the sampler to sample the single subset $C^{-1}(1)$ approximately correctly.

Definition 7.2 ($\mathcal{C}$-computable sampler)**.** *Let* $\mathsf{Samp}\colon \{0,1\}^n \to (\{0,1\}^{n_0})^{2^\ell}$ *be a (B,ϵ)-sampler. For a class $\mathcal{C}_n$ of circuits with n input gates, we say that* Samp is computable in $\mathcal{C}_n$ *if for every fixed $s \in \{0,1\}^\ell$, each output bit of the function $\mathsf{Samp}^{(s)}(z) = \mathsf{Samp}(z)_s$ is computable by a circuit in $\mathcal{C}_n$.*

The notion of "computable in $\mathcal{C}$" in Definition 7.2 is relaxed: We only require that for each *fixed output index s* of the sampler, each output bit of $\mathsf{Samp}(\cdot)_s$ will be computable by a $\mathcal{C}$-circuit (i.e., we can think of the circuit as having s hard-wired).

7.2 The application to derandomization and its limitations

Let me repeat the high-level ideas for using the notions in Definitions 7.1 and 7.2 towards derandomization (elaborating on the descriptions in Section 2.4), and then state the limitation of these notions for this application.

Suppose that we are given a circuit $C \in \mathcal{C}$ over n_0 input bits as input for CAPP, and we want to first reduce the problem to QD_B and then solve QD_B. In the first step, we want a reduction to QD_B of an n-bit circuit $C' \in \hat{\mathcal{C}}$ such that B is small, the class $\hat{\mathcal{C}}$ is not much stronger than $\mathcal{C}$, and n is not much larger than n_0. To do so we use a sampler $\mathsf{Samp}\colon \{0,1\}^n \to (\{0,1\}^{n_0})^{2^\ell}$ with $B^{\mathtt{thr}}$ bad inputs, and construct the circuit $C'(z) = \mathsf{MAJ}\{C(\mathsf{Samp}(z)_s)\}_s$. Indeed, C' has at most $B^{\mathtt{thr}}$ exceptional inputs, since for all but $B^{\mathtt{thr}}$ of the inputs z to C' the sampler Samp will sample the event $C^{-1}(1)$ correctly, up to a small error of $1/10$. Also, if Samp is computable by relatively simple circuits, then the complexity of C' is not too large compared to C. (To further reduce the complexity overhead we can replace the majority function by an approximate majority function, or even by an $\vee$ gate if we are only interested in solving derandomization with one-sided error, i.e. $\mathsf{CAPP}_{\frac{1}{2},0}$.)

Now we have a circuit $C' \in \hat{\mathcal{C}}$ and we want to solve the quantified derandomization problem for C'. Following an idea of Goldreich and Wigderson [GW14], if we can efficiently sample simplifier sets of size more than $B^{\mathtt{trac}}$ for C', then we can efficiently solve $\mathsf{QD}_{B^{\mathtt{trac}}}$ for C'. Specifically, if we can sample $X \sim \mathbf{X}_n$ in time T using seed length s, then we can solve $\mathsf{QD}_{B^{\mathtt{trac}}}$ for C' in time $T \cdot 2^s \cdot \tilde{O}(|C'|)$. [33] This is since for most choices of $X \sim \mathbf{X}_n$ it holds that $C'|_X$ is constant, and this constant is the majority output of C' (because $|X|$ is larger than the number $B^{\mathtt{trac}}$ of exceptional inputs).

At this point enters the limitation of the combination of these two black-box techniques: If a circuit class $\hat{\mathcal{C}}$ can compute a sampler with $B^{\mathtt{thr}}$ bad inputs, then distributions of simplifier sets for $\hat{\mathcal{C}}$ are necessarily supported by sets of size $B^{\mathtt{trac}}$ that is smaller than $B^{\mathtt{thr}}$. In other words, *samplers cannot reduce* CAPP *to a quantified derandomization problem that can be solved using only simplifier sets*. The following result, which formalizes the foregoing statement, is the formal version of Theorem 2.12.

Theorem 7.3 (a limitation of two black-box techniques in quantified derandomization; see [Tel17])**.** *Let* $C\colon \{0,1\}^{n_0} \to \{0,1\}$*, and let* $\mathsf{Samp}\colon \{0,1\}^n \to (\{0,1\}^{n_0})^{2^\ell}$ *be a sampler with* $B^{\mathtt{thr}}$ *bad inputs such that* $2^\ell \leq \frac{1}{5} \cdot 2^{n_0/4}$*. Assume that* Samp *is computable in a circuit class* $\hat{\mathcal{C}}_n$*. Then, for any distribution over simplifier sets of size more than* $B^{\mathtt{trac}}$ *for* $\hat{\mathcal{C}}_n$ *it holds that* $B^{\mathtt{trac}} < B^{\mathtt{thr}}$*.*

The meaning of Theorem 7.3 is that for *any* class $\mathcal{C}$ for which we want to solve CAPP (i.e., for any class from which the initial circuit C comes), if we reduce CAPP to $\mathsf{QD}_{B^{\mathtt{thr}}}$ of $\hat{\mathcal{C}}$-circuits using black-box samplers, then simplifier sets for $\hat{\mathcal{C}}$ of size more than $B^{\mathtt{trac}}$ satisfy $B^{\mathtt{trac}} < B^{\mathtt{thr}}$. Also note that Theorem 7.3 holds regardless

[33] The precise requirement from the algorithm that "samples $X \sim \mathbf{X}_n$" is that it can efficiently find an arbitrary input $x \in X$, where X is sampled from $\mathbf{X}_n$.

of the seed length or computational complexity of sampling from the distribution of simplifier sets (i.e., it suffices to assume that such a distribution exists).

Remark 7.4. In Theorem 7.3 the gap between the parameter $B^{\mathtt{thr}}$ to which we can reduce CAPP and the parameter $B^{\mathtt{trac}}$ that can be handled with simplifier sets is only guaranteed to be a single bit. However, in most applications the gap seems likely to be noticeably larger: This is since the circuit class to which we reduce CAPP is one that is not only capable of computing Samp, but also capable of computing the more complicated function $C'(z) = \mathsf{MAJ}\{C(\mathsf{Samp}(z)_s)\}_s$. Intuitively, we expect that simplifier sets for this more complicated class will be of even smaller size, and thus we expect the gap between $B^{\mathtt{trac}}$ and $B^{\mathtt{thr}}$ to be typically larger.

Remark 7.5. Theorem 7.3 also holds if we replace the two notions of $\mathcal{C}$-computable samplers and distributions over simplifier sets with stricter notions, as follows. In Definition 7.2, instead of only requiring each output bit of the function $\mathsf{Samp}^{(s)}(z) = \mathsf{Samp}(z,s)$ to be computable in $\mathcal{C}_n$, we require that the mapping of z to *all output strings* $\{\mathsf{Samp}(z,s)\}_{s\in\{0,1\}^\ell}$ to be computable by a multi-output circuit in $\mathcal{C}_n$; and in Definition 7.1, we require that for every $C \in \mathcal{C}_n$ with multiple output bits, $\mathbf{X}_n$ simplifies each output bit of C with (marginal) probability more than $1/2$. The technical constructions underlying the results in Section 6 in fact satisfy these two stricter notions (see Appendices B and C for details), and I presented Theorem 7.3 using the original relaxed notions merely for simplicity.

Caveats: The fine print. The limitation in Theorem 7.3 applies to the techniques underlying all the results in Section 6. To be specific, while the foregoing techniques do not *strictly* adhere to the two clean notions defined in Definitions 7.1 and 7.2, they are nevertheless "close enough" to the clean notions so that the limitation still applies to them. Let me now explain why this is the case.

The first gap between the clean notions and the actual techniques is that in the cases of $\mathcal{AC}^0$ and of LTF circuits, the restriction procedures do not simplify the given circuit to a constant, but only simplify it so that it is *close* to a constant; that is, the restricted circuit is constant on $1-\delta$ of the inputs in X, for a very small $\delta > 0$. This is not a significant issue, since in the results above $\delta > 0$ is sufficiently small to allow essentially the same proof as that of Theorem 7.3 to follow through.[34]

The second gap is that for LTF circuits, the quantified derandomization algorithm is actually *not* a black-box algorithm: Given a circuit C, it finds a large subcube $X_C \subseteq \{0,1\}^n$ such that $C{\restriction}_{X_C}$ simplifies. This initially appears to be precisely a non-black-box technique that we are looking for, but unfortunately the underlying technical result

[34]Specifically, the proof relies on the fact that there exists a subset X such that for at least $1/3$ of the seeds, at least $1/4$ of the sampler's output bits given this seed are constant in X. If we are only guaranteed that these output bits are $(1-\delta)$-close to a constant, but with a small enough δ, by a union-bound there is a subset $X' \subseteq X$ of size $|X'| = (1-o(1))\cdot|X|$ such that these bits are constant on X'. In this case the proof follows through with a minor increase in the size of the allowed set X, which does not seem meaningful given that Theorem 7.3 is probably not fully tight (as mentioned in Remark 7.4).

is "black-box enough". In more detail, Theorem 7.3 holds even if the distribution of simplifier sets depends on the given circuit, as long as it satisfies the following: When the circuit has multiple output bits, the distribution simplifies each of the output bits of the circuit with marginal probability more than $1/2$. This is precisely what happens in the restriction procedure for LTF circuits (see [Tel18] for details).

The third gap is that the reduction in [CJW20] of CAPP to QD_B for formulas, they use dispersers instead of extractors, and apply an additional trick of reducing CAPP to QD_B of probabilistic formulas. Nevertheless, the known distributions of simplifier sets for formulas have sufficiently strong properties (namely, they assert "shrinkage with high probability") so that the limitation still holds for the type of construction as in [CJW20]; see Appendix C.3 for a detailed explanation.

The last and more meaningful gap between the clean notions and the actual techniques is that for $\mathcal{AC}^0$ and LTF circuits, the restriction procedures assert that the circuit is approximated not by a constant, but by a "very simple" circuit (either a depth-two formula or a single LTF).[35] Such an approach is a-priori a promising one to bypass the limitation in Theorem 7.3. However, for the specific algorithms underlying the results above, simplifying to a "very simple" circuit rather than to a constant is an optimization that improves relatively minor terms,[36] which seem likely to be smaller than the slackness that exists in Theorem 7.3 (and was mentioned in Remark 7.4).

7.3 Relaxations that do (and do not) suffice to bypass the limitation

Theorem 7.3 means that if we want to solve CAPP by first reducing the error and then finding a large subset on which the circuit simplifies, then *at least one* of the two steps must be executed in a non-black-box fashion. Let me spell out two natural approaches to do so, one for each of the steps.

A non-black-box restriction procedure. Given a circuit C' for QD_B (we think of C' as obtained by reducing the error of a circuit C using samplers), one natural way to bypass the limitation in Theorem 7.3 is to try and sample from a large subset $X_{C'} \subseteq \{0,1\}^n$ that *depends on* C' such that $C'|_{X_{C'}}$ is constant. Recall that there is *always* a *huge* subset, of size $(1-o(1)) \cdot 2^n$, on which C' is constant (i.e., the preimage of the majority output of C'), and thus the question is purely algorithmic.

The main caveat to look out for is the fact (mentioned above) that the limitation in Theorem 7.3 continues to hold even if $X_{C'}$ depends on C', in case the algorithm operates in a specific way that is "somewhat black-box". In particular, the limitation holds when the algorithm uses a distribution $\mathbf{X}_{C'}$ that depends on C', and simplifies each of

[35] The algorithm for QD_B then does not just evaluate the original circuit on an arbitrary input from X, but estimates the acceptance probability of the "very simple" circuit on X using an efficient PRG.

[36] For $\mathcal{AC}^0$ this saves a single unit in the constant polylogarithmic power in the expression $B(n) = 2^{n/\mathrm{polylog}(n)}$ (see [Tel19]), whereas for LTF circuit this optimizes the constant c in the size bound $n^{1+c^{-d}}$, yet still leaves it far above the required $c = 1.61$ in the threshold result (see [Tel18; CT19]).

the gates in any layer of C' with marginal probability more than $1/2$. (See [Tel17] for further details.)

A non-black-box sampler. Another approach is to use error-reduction that is based on non-black-box samplers. Specifically, recall that a sampler as in Definition 7.2 samples any subset $T \subseteq \{0,1\}^{n_0}$ approximately correctly, regardless of the computational complexity of deciding T. In our case we will use the sampler to sample just one subset, namely $C^{-1}(1) \subseteq \{0,1\}^{n_0}$, which is not only decidable by a circuit of bounded size, but such that we are explicitly given a description of this circuit.

This seems to be a promising approach to bypass the limitation in Theorem 7.3. That is, in this approach our goal is to construct a "sampler" that, compared to a general-purpose sampler as in Definition 7.2, has a smaller number of bad inputs, but correctly samples less subsets (i.e., it only samples subsets decidable by circuits from the relevant class, or only samples the specific subset C^{-1}).[37]

8 Polynomials that vanish extremely rarely

We are interested in multivariate polynomials $p\colon \mathbb{F}^n \to \mathbb{F}$, where $\mathbb{F}$ is a finite field and n is sufficiently large. I'll denote the field size by $q = |\mathbb{F}|$ and the total degree of a polynomial by $d = \deg(p)$, and our goal is to construct hitting-set generators for polynomials that vanish extremely rarely, where "rarely" here refers to the *probability* that the polynomial vanishes $\Pr_{x\in\mathbb{F}^n}[p(x) = 0]$ (i.e., to the fraction of roots).

Definition 8.1 (polynomials that vanish rarely)**.** *For $n, d, q, t \in \mathbb{N}$, let $\mathcal{P}_{n,d,q,t}$ be the set of polynomials $p\colon \mathbb{F}_q^n \to \mathbb{F}_q$ of degree at most d that vanish on at most an $\epsilon = q^{-t}$ fraction of their inputs; that is, $\Pr_{x\in\mathbb{F}_q^n \to \mathbb{F}_q}[p(x) = 0] \leq \epsilon$.*

Definition 8.2 (HSGs for polynomials that vanish rarely)**.** *We say that $H\colon \{0,1\}^\ell \to \mathbb{F}_q^n$ is a* HSG for $\mathcal{P}_{n,d,q,t}$ *if for every $p \in \mathcal{P}_{n,d,q,t}$ there exists $s \in \{0,1\}^\ell$ such that $p(H(s)) \neq 0$. We stress that the* seed length ℓ *of H is measured in bits.*

8.1 Upper bounds over $\mathbb{F}_2$

Kaufman, Lovett, and Porat [KLP12] proved an upper bound on the number of degree-d polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish on at most an $\epsilon = 2^{-t}$ fraction of their roots; specifically, they showed that the number of such polynomials is at most $2^{O(K)}$ where $K = \frac{d^2 \cdot t}{d-t+1} \cdot n^{d-t+1}$. This bound is almost tight, and the possible minor tightening is immaterial for our application (see [ASW15] and [DTST20, After the proof of Theorem 22] for explanations). By a standard probabilistic argument first noted by Doron, Ta-Shma and the current author [DTST20], it follows that:

[37] When trying to bypass the limitation in Theorem 7.3 with a "sampler" that correctly samples all subsets decidable by circuits of size S, the seed length ℓ needs to be larger than $\log(S)$. This is since the proof of Theorem 7.3 shows that, assuming a distribution over simplifier sets exists, there is a subset that is sampled *incorrectly* and that is decidable by a DNF of size 2^ℓ (see [Tel17, Section 4.2]).

Theorem 8.3 (non-uniform HSG for polynomials that vanish rarely; see [DTST20, Theorem 22])**.** *Let $n, d, t \in \mathbb{N}$ where $t < d \leq n$. Then, there exists a non-uniform HSG for $\mathcal{P}_{n,d,2,t}$ with seed length $O\left((d-t) \cdot \log(n/(d-t))\right)$.*

Goldreich and Wigderson [GW14] constructed a *polynomial-time computable* HSG with seed length $O(\log(n))$ for polynomials that vanish with probability at most $\epsilon = O(2^{-d}) = 2^{-(d-O(1))}$, and a different analysis of their construction was subsequently given in [Tel19]. The latter analysis was then extended in [DTST20] to show the following more general construction:

Theorem 8.4 (efficient HSG for polynomials that vanish rarely; see [DTST20, Theorem 3])**.** *Let $n \in \mathbb{N}$ be sufficiently large, and let $d > t + 4$ be integers that may depend on n. Then, there exists a polynomial-time computable HSG for $\mathcal{P}_{n,2,d,t}$ with seed length $O\left((d-t) \cdot (2^{d-t} + \log(n/(d-t)))\right)$.*

Note that the result of [GW14] is the special case of Theorem 8.4 with $t = d - O(1)$. The HSG in Theorem 8.4 is essentially Viola's [Vio09b] PRG for low-degree polynomials, instantiated for a suitable degree that depends on $d - t$. (Viola's PRG, in turn, computes the element-wise sum of independent copies of a PRG for linear polynomials as in [NN93].) The analysis boils down to approximating any polynomial that vanishes rarely by a probabilistic low-degree polynomial, and relying on the fact that a PRG for low-degree polynomials also fools any function that is approximated by probabilistic low-degree polynomials (see [DTST20, Theorem 24] for details).

8.2 Lower bound over general finite fields

The following lower bounds by Doron, Ta-Shma and the current author [DTST20] hold for *any* HSG for polynomials that vanish rarely, regardless of its computational complexity, and over fields of size $2 \leq q \leq \text{poly}(n)$. I'll first state a nice special case, which holds either for degree up to $n^{.49}$, or over fields of size $q \geq n$ and for degree up to $n^{.99}$; for simplicity, in this special case we assume $q \leq n^{100}$. Then I'll state the more general result, which holds also for a broader range of parameters but is technically cumbersome to state.

Theorem 8.5 (lower bound on HSGs for polynomials that vanish rarely; a nice special case)**.** *Let $n \in \mathbb{N}$ be sufficiently large, let $d \leq n^{.99}$, let $q \leq n^{100}$ be a prime power, and let $t \leq \delta \cdot d$, where $\delta > 0$ is a sufficiently small universal constant. Further assume that either $q \geq n$, or $d \leq n^{.49}$. Then, the seed length of any HSG for $\mathcal{P}_{n,q,d,t}$ is at least $\Omega\left(\frac{d}{t} \cdot \log\left(\frac{n}{d/t}\right)\right)$.*

Theorem 8.6 (lower bound on HSGs for polynomials that vanish rarely; see [DTST20, Theorem 28])**.** *For any two constants $\gamma > 0$ and $c > 1$ there exists a constant $\delta > 0$ such that the following holds. Let $n \in \mathbb{N}$ be sufficiently large, let $d \leq n/4$, let $q \leq n^c$ be a prime power, and let $t \leq \delta \cdot d$. Assume that:*

1. (main condition:) $d/t \leq \delta \cdot \min\left\{\frac{q-1}{\log(q)} \cdot n^{\gamma}, n^{1-(\gamma+1/c)}\right\}$.

2. (auxiliary condition that holds for typical settings:) $\frac{q-1}{\log(q)} \cdot \log(nt/d) \geq 1/\delta$.

Then, the seed length of any HSG for degree-d polynomials $\mathbb{F}_q^n \to \mathbb{F}_q$ that vanish with probability at most $\epsilon = \sqrt{2} \cdot q^{-t}$ is at least $\Omega\left(\frac{d}{t} \cdot \log\left(\frac{n^{1-(\gamma+1/c)}}{d/t}\right)\right)$.

The proof of Theorem 8.6 applies the idea of disperser-based error-reduction, which is natural for Boolean circuits, in the arithmetic setting of polynomials. Specifically, recall that the seed length of any HSG for *general* n_0-variate degree-d_0 polynomials is at least $\Omega(d_0 \cdot \log(n_0/d_0))$ (see, e.g., [DTST20, Preliminaries]). The proof idea is to transform any degree-d_0 polynomial $p_0 \colon \mathbb{F}^{n_0} \to \mathbb{F}$ into a polynomial $p \colon \mathbb{F}^n \to \mathbb{F}$ defined by

$$p(z) = \bigvee_{i \in \{0,1\}^\ell} p_0(\mathsf{Disp}(z,i)) \,,$$

where Disp is a sufficiently good disperser such that p vanishes extremely rarely and has relatively low degree $d > d_0$.[38] If such a transformation exists, then any HSG for degree-d polynomials that vanish with probability at most $\epsilon = \Pr_z[p(z) = 0]$ whose seed length is $K_{n,d,\epsilon}$ yields a HSG for *general* polynomials of degree d_0 with seed length $K_{n,d,\epsilon} + \ell$ (where ℓ is the seed length of the disperser above). By the known lower bound on the latter, it follows that $K_{n,d,\epsilon} \geq \Omega(d_0 \cdot \log(n_0/d_0)) - \ell$.

Unfortunately, this idea does not work as-is, since the blow-up both in the degree (i.e., $d_0 \mapsto d$) and in the number of variables (i.e., $n_0 \mapsto n$) is too large to yield the sought lower bound. To implement the idea with smaller blow-ups and prove the better lower bound, in [DTST20] they use an additional trick of approximating the p by a specific probabilistic polynomial of lower degree, and adapt the extractor construction of Shaltiel and Umans [SU05] to yield a good disperser Disp that is computable by a linear function over $\mathbb{F}$. Further details can be found in [DTST20, Section 2.1].

8.3 The connection to small sets with large degree-d closures

Nie and Wang [NW15] introduced the natural notion of a degree-d closure of a set $S \subseteq \mathbb{F}^n$, which is the variety induced by (i.e., set of common roots of) the set of polynomials of degree at most d that vanish on S. Readers who are familiar with algebraic geometry may recognize this as a natural bounded-degree analogue of the *Zariski closure* of S.

Definition 8.7 (degree-d closure)**.** *Let $\mathbb{F}$ be a finite field, let $n, d \in \mathbb{N}$, and let $S \subseteq \mathbb{F}^n$. The* degree-d closure of S *is*

$$\mathtt{Cl}^{(d)}(S) = \left\{ x \in \mathbb{F}^n : \forall p \in \mathcal{I}_S^{(d)}, p(x) = 0 \right\} \,,$$

[38] The property of Disp that we need for p to vanish extremely rarely is that for almost all z's there exists i such that $\mathsf{Disp}(z,i)$ "hits" the set $S = \{x : p_0(x) \neq 0\}$. This property is natural in the case that p_0 is of low degree compared to the field size (i.e., $d_0 \ll q$, which implies that S is large), and is less straightforward in other settings (i.e., when S might be small); see [DTST20] for details.

where $\mathcal{I}_S^{(d)}$ is the set of polynomials $p\colon \mathbb{F}^n \to \mathbb{F}$ of degree at most d that vanish on S (i.e., for all $s \in S$ it holds that $p(s) = 0$).

As a nice example, observe that the degree-d closure of any $d+1$ points on a fixed line in $\mathbb{F}^n$ contains the entire line. A natural question is whether there exists a *very small* set with a *very large* degree-d closure. This is where the connection to derandomization enters: Observe that $\mathtt{Cl}^{(d)}(S) = \mathbb{F}^n$ (i.e., the closure of S is the entire space) if and only if S is a hitting-set for degree-d polynomials. (Since in both cases, the only degree-d polynomial that vanishes on S is the zero polynomial.)

A more robust connection between degree-d closures and hitting-sets for polynomials, which refers to a closure that is large but isn't necessarily the entire space $\mathbb{F}^n$, comes from considering polynomials that *vanish rarely*. Loosely speaking, the following result from [DTST20] asserts that sets with a *large* degree-d closure are hitting-sets for degree-d polynomials that *vanish rarely*, and vice versa; that is:

Theorem 8.8 (small sets with large closures vs hitting-sets for polynomials that vanish rarely)**.** *Let $\mathbb{F}$ be a field of size $q = |\mathbb{F}|$, let $t < d < n$ be integers, and let $S \subseteq \mathbb{F}^n$. Then,*

$$\left|\mathtt{Cl}^{(d)}(S)\right| > q^{n-t} \implies S \text{ is a hitting-set for } \mathcal{P}_{n,q,d,t} \implies \left|\mathtt{Cl}^{(d/2(t+1))}(S)\right| > \frac{1}{2}\cdot q^{n-t}\,.$$

Indeed, Theorem 8.8 does not show a complete equivalence between the two notions, because the RHS refers to degree $\approx d/2t$ rather than d. Thus, intuitively, the result means that constructing a small set with a large degree-d closure is at least as hard as constructing a hitting-set for polynomials that vanish rarely; and while the result also shows a converse implication, it is nevertheless possible that constructing a hitting-set for polynomials that vanish rarely is an easier problem.

9 Quantified derandomization and pseudoentropy

As explained in Section 2.6, there is a close connection between quantified derandomization and metric pseudoentropy, as defined by Barak, Shaltiel and Wigderson [BSW03]. Let us recall the definition and then prove the equivalence between metric pseudoentropy and pseudorandomness for biased distinguishers.

Definition 9.1 (metric pseudoentropy, as defined in [BSW03])**.** *We say that a distribution $\mathbf{w}$ over $\{0,1\}^n$ has* metric ϵ-pseudoentropy k for a class $\mathcal{C} \subseteq \{\{0,1\}^n \to \{0,1\}\}$ *if for every $C \in \mathcal{C}$ there exists a distribution $\mathbf{h}_C$ over $\{0,1\}^n$ with min-entropy[39] at least k such that $\Pr[C(\mathbf{w}) = 1] \in \Pr[C(\mathbf{h}_C) = 1] \pm \epsilon$.*

The following is the formal version of Theorem 2.15. It was proved by Doron *et al.* [DMO+20, Section 5], and appears in the form below in [CT21b, Proposition 3.11]. For completeness, let me include the short proof.

[39] Recall that the min-entropy of a distribution is the largest integer k such that every outcome has probability 2^{-k} or less.

Theorem 9.2 (metric pseudoentropy vs pseudorandomness for biased distinguishers)**.** *For any distribution* $\mathbf{w}$ *over* $\{0,1\}^n$ *and every* $k < n$ *and* $\epsilon, \delta > 0$ *the following holds:*

1. *If* $\mathbf{w}$ *has metric* ϵ*-pseudoentropy at least* k *for circuits of size* S*, then* $\mathbf{w}$ *is* $(\epsilon + \delta)$*-pseudorandom for size-*S *circuits with at most* $B(n) = \delta \cdot 2^k$ *exceptional inputs.*

2. *If* $\mathbf{w}$ *is* ϵ*-pseudorandom for size-*S *circuits with at most* $B(n) = 2^k$ *exceptional inputs, then* $\mathbf{w}$ *has metric* ϵ*-pseudoentropy at least* k *for size-*S *circuits.*

Proof. The core of the proof is the following lemma proved by Barak, Shaltiel and Wigderson [BSW03, Lemma 3.3]:

Lemma 9.2.1. *A distribution* $\mathbf{w}$ *has metric* ϵ*-pseudoentropy at least* k *for size-*S *circuits if and only if for every size-*S *circuit* D *over* n *bits and every* $\sigma \in \{0,1\}$ *it holds that* $\Pr[D(\mathbf{w}) = \sigma] \leq \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \epsilon$.

Now, assume that $\mathbf{w}$ has metric ϵ-pseudoentropy at least k for size-S circuits, and let D be an n-bit size-S circuit with at most $\delta \cdot 2^k$ exceptional inputs. Then, for the majority output σ of D the bound $\Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \epsilon$ is trivial (recall that $k \leq n-1$); and for the minority output σ' of D it holds that

$$\Pr[D(\mathbf{w}) = \sigma'] \leq \Pr[D(\mathbf{u}_n) = \sigma'] \cdot 2^{n-k} + \epsilon \leq \delta + \epsilon \,,$$

where the second inequality is since $\Pr[D(\mathbf{u}_n) = \sigma'] \leq \delta \cdot 2^{k-n}$.

For the other direction, assume that $\mathbf{w}$ is ϵ-pseudorandom for size-S circuits with at most 2^k exceptional inputs, and let us prove that for every size-S circuit $D\colon \{0,1\}^n \to \{0,1\}$ and $\sigma \in \{0,1\}$ it holds that $\Pr[D(\mathbf{w}) = \sigma] \leq \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \epsilon$. To see this, note that if $\Pr[D(\mathbf{u}_n) = \sigma] > 2^{k-n}$ then the bound is trivial; and otherwise, D has at most 2^k exceptional inputs, in which case

$$\Pr[D(\mathbf{w}) = \sigma] \leq \Pr[D(\mathbf{u}_n) = \sigma] + \epsilon \leq \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \epsilon \,. \quad \blacksquare$$

One implication of the equivalence in Theorem 9.2, which is essentially the content of Theorem 2.16, is that PRG constructions that "extract randomness from a pseudoentropic string" can be analyzed in an easier and more general way by thinking of the construction as "error-reduction and then quantified derandomization". To see this, let me restate Theorem 2.16 with more general parameters and include a full proof, which fleshes out an idea explained in [CT21b, Section 2.2].

Theorem 9.3 ("extract from a pseudoentropic string" as a special case of "error-reduction and quantified derandomization")**.** *Let* $\mathsf{Ext}\colon \{0,1\}^{\bar{n}} \times \{0,1\}^{O(\log(\bar{n}))} \to \{0,1\}^n$ *be a* (k,ϵ)*-extractor, where* $k \leq \bar{n} - \log(1/\epsilon)$*, that is computable in time* $\mathrm{poly}(n)$*, and let* $G_0\colon \{0,1\}^\ell \to \{0,1\}^{\bar{n}}$ *be a metric* ϵ*-pseudoentropy generator with pseudoentropy* $k + \log(1/\epsilon)$ *for circuits of size* n^c *for a sufficiently large constant* $c > 1$ *that depends on* Ext*. Then,* $G(s_0, s_1) = \mathsf{Ext}(G_0(s_0), s_1)$ *is a* 4ϵ*-PRG for linear-sized circuits.*

Proof. Fix a linear-sized circuit C over n bits, and denote the acceptance probability of C by $\mu = \Pr_{x\in\{0,1\}^n}[C(x) = 1]$. For $\bar{\ell} = O(\log(\bar{n}))$, we call a string $z \in \{0,1\}^{\bar{n}}$ good if $\Pr[\mathsf{Ext}(z, \mathbf{u}_{\bar{\ell}}) \in C^{-1}(1)] \in \mu \pm \epsilon$.

We define a function $\bar{C}$ over $\bar{n}$ input bits that accepts its input $z \in \{0,1\}^{\bar{n}}$ if and only if z is good. Note that $\bar{C}$ can be computed by a circuit of size n^c, where $c \in \mathbb{N}$ is a sufficiently large constant that depends on the precise $\mathrm{poly}(n)$ time complexity of Ext and on its seed length. (Indeed, the circuit for $\bar{C}$ has the value μ hard-wired, whereas our algorithm G does not "know" μ; but at this point we are only defining $\bar{C}$ as a thought experiment in the analysis.)

By the properties of Ext it holds that $\bar{C}$ accepts all but 2^k inputs $z \in \{0,1\}^{\bar{n}}$. We also claim that any distribution $\mathbf{w}$ that is 2ϵ-pseudorandom for $\bar{C}$ yields a distribution $\mathsf{Ext}(\mathbf{w}, \mathbf{u}_{\bar{\ell}})$ that is 4ϵ-pseudorandom for C. To see this, note that

$$\begin{aligned}
\Pr[C(\mathsf{Ext}(\mathbf{w}, \mathbf{u}_{\bar{\ell}})) = 1] &\leq \Pr[\mathbf{w} \text{ is not good}] + (\mu + \epsilon) \\
&= \Pr[\bar{C}(\mathbf{w}) = 0] + \mu + \epsilon \\
&\leq 2^{k-\bar{n}} + \mu + 3\epsilon && (|C^{-1}(0)| \leq 2^k|) \\
&\leq \mu + 4\epsilon\,, && (k \leq \bar{n} - \log(1/\epsilon))
\end{aligned}$$

and by a similar calculation $\Pr[C(\mathsf{Ext}(\mathbf{w}, \mathbf{u}_{\bar{\ell}})) = 0] \leq 1 - \mu + 4\epsilon$.

Finally, by Theorem 9.2 (instantiated with $\delta = \epsilon$ and with pseudoentropy $k + \log(1/\epsilon)$), we have that G_0 is 2ϵ-pseudorandom for $\bar{C}$. Thus, the distribution $\mathsf{Ext}(G_0(\mathbf{u}_\ell), \mathbf{u}_{\bar{\ell}})$ is 4ϵ-pseudorandom for C, as we wanted. ■

10 A host of concrete challenges

The area of quantified derandomization is rich with unsolved open problems. Let me name a few prominent ones, while suggesting that interested readers also refer to specific papers such as [GW14; Tel19; CT19; CJW20; CT21b].

The first problem calls for improving the connection between quantified derandomization and *circuit lower bounds*. Theorems 2.2 and 2.3 deduce very fast quantified derandomization, but only under hardness assumptions that refer to $\mathcal{SVN}$ circuits, which are stronger assumptions than lower bounds for standard circuits. Can we deduce very fast quantified derandomizatoin from lower bounds for standard circuits?

Open Problem 1: Deduce very fast quantified derandomization from lower bounds for standard circuits. *Show that $pr\mathcal{BPTIME}_B[n] \subseteq pr\mathcal{DTIME}[n^{2.01}]$, for some $B(n) \gg n$, under the assumption that $\mathcal{DTIME}[2^n]$ is hard for circuits of size $2^{(1-o(1))\cdot n}$.*

A second main direction in which we hope to strengthen the conditional quantified derandomization in Theorems 2.2 and 2.3 is to allow for more exceptional random strings. Recall that the non-uniform quantified derandomization in Theorem 2.1 holds for $B(T) = 2^{(1-\Omega(1))\cdot T}$, whereas the conditional quantified derandomization in Theorems 2.2 and 2.3 holds only for the smaller value $B(T) = 2^{T^{1-\Omega(1)}}$. Can we strengthen the latter results to hold for larger B's?

Open Problem 2: Hardness-to-quantified-randomness with relatively many exceptional inputs. *For some $B(T) = 2^{T^{1-o(1)}}$, show that $prBPTIME_B[n] \subseteq prDTIME[n^{2.01}]$ under appealing hardness hypotheses (e.g., similar to the ones in Theorems 2.2 and 2.3).*

Can we construct a new algorithm for general derandomization (i.e., for CAPP) by first applying error-reduction and then solving the corresponding quantified derandomization problem? This question is particularly appealing for classes of low-depth circuits such as $\mathcal{AC}^0$, $\mathcal{TC}^0$, or FORMULAS, for which the known reductions of CAPP to QD_B yield parameters that are remarkably close to those that known algorithms for QD_B can handle. For concreteness, let me suggest this problem for the class $\mathcal{TC}^0$:

Open Problem 3: An algorithm for CAPP via quantified derandomization. *Improve either the algorithm in Corollary 6.11 or the reduction in Theorem 6.13 such that the tractable values for $\mathcal{TC}^0$ surpass the threshold values for $\mathcal{TC}^0$. Deduce that there exists a non-trivial CAPP algorithm for $\mathcal{TC}^0$, and consequently that $\mathcal{NEXP} \not\subset \mathcal{TC}^0$.*

The following open problem calls for materializing an approach outlined in Section 7.3. As explained there, a natural way to bypass the limitation in Theorem 2.12 is to construct a non-black-box sampler: Such an algorithm gets as input an n-bit circuit C and a long random string $z \in \{0,1\}^{\bar{n}}$ (for some $\bar{n} > n$), and outputs a small sample $S \subseteq \{0,1\}^n$ such that for all but a small number of strings z it holds that $C^{-1}(1)$ is sampled approximately correctly in S. The question presented next asks whether we can construct such samplers with parameters that are better than the ones possible for general-purpose samplers. For concreteness, the question focuses on the case of $\mathcal{AC}^0$, but it is nevertheless interesting for any natural circuit class.

Open Problem 4: Construct a non-black-box sampler that outperforms general-purpose samplers. *For example, construct a polynomial-sized $\mathcal{AC}^0$ circuit that gets as input a description of an n-bit $\mathcal{AC}^0$ circuit C and a random string $z \in \{0,1\}^{\bar{n}}$, and for all but at most $2^{\bar{n}^{1-\Omega(1)}}$ strings z it outputs a sample $S \subseteq \{0,1\}^n$ such that $\Pr_{s\in S}[C(s) = 1] \in \Pr_{x\in\{0,1\}^n}[C(x) = 1] \pm .01$.*

Turning to polynomials that vanish extremely rarely, as mentioned in Section 2.5, I find it remarkable that we do not even know the optimal size of a *non-uniform* hitting-set for such polynomials. Is the optimal seed length closer to $(d-t)\cdot\log(n/(d-t))$ as in the upper bound in Theorem 2.13, or closer to $(d/t)\cdot\log(n/(d/t))$ as in the lower bound in Theorem 2.14? Since the upper bound is obtained by simply choosing strings at random (and relying on a bound on the number of polynomials), this question boils down to asking whether or not a random set of strings is an optimal HSG.

Open Problem 5: What is the optimal size of HSGs for polynomials that vanish extremely rarely? *What is the minimal seed length of a hitting-set (not necessarily an efficiently computable one) for degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$ that vanish with probability at most ϵ, as a function of $n, d, |\mathbb{F}|$, and ϵ? What is the minimal seed length when $|\mathbb{F}| = 2$?*

As a special case of Open Problem 5, recall that for fields of size larger than two, even non-tight upper bounds such as the ones in Theorem 2.13 are currently unknown. The following question calls for showing *some* upper bounds for fields of size $|\mathbb{F}| > 2$, even in the "slightly non-trivial" case where the fraction of roots is $\epsilon = O(q^{-d})$. (Recall that a polynomial with less than q^{-d} roots has no roots at all [War35], and therefore the problem is non-trivial only for $\epsilon \geq q^{-d}$.)

Open Problem 6: Show results similar to Theorem 2.13 over fields other than $\mathbb{F}_2$. *For a field $\mathbb{F}$ of size $q > 2$ and every $c > 1$, construct an HSG with seed length $o(d \cdot \log(n))$ for degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$ that vanish with probability at most $c \cdot q^{-d}$. (The HSG may even be non-uniform.)*

The next open problem refers to quantified derandomization of $\mathcal{AC}^0[\oplus]$ circuits. Recall that at the moment we do not know of a quantified derandomization algorithm even for depth-3 circuits with $B(n) = 2^{n^{\Omega(1)}}$ exceptional inputs, and the culprit is the last remaining open case of $\oplus \circ \mathsf{AND} \circ \oplus$ circuits. Can we handle this case?

Open Problem 7: Solve the remaining open case of depth-3 $\mathcal{AC}^0[\oplus]$ circuits. *For every $k \in \mathbb{N}$ and an arbitrarily small $\epsilon = \epsilon(k) > 0$, construct an algorithm that solves QD_B for $\oplus \circ \mathsf{AND} \circ \oplus$ circuits of size n^k, where $B(n) = 2^{n^\epsilon}$.*

The last open problem is stated in a broad and somewhat vague way, but I believe that there is value in reminding the reader of it. This problem refers to further exploring the close connection between quantified derandomization (a relatively new notion) and pseudoentropy (a classic notion), in the hope of finding further implications of this connection (other than Theorem 2.16).

Open Problem 8: Explore the connection between quantified derandomization and pseudoentropy. *As I mentioned in Section 2.6, I believe that more interesting implications of this connection exist and have yet to be discovered.*

Additional open problems, which focus on quantified derandomization of logspace machines and of interactive proofs, are suggested in Appendix D.

Acknowledgements

I'm grateful to Oded Goldreich for many valuable comments on a draft of the survey, for elaborate discussions about the conceptual perspective, and for pointing out the elementary proof of Theorem 2.1 (replacing an original complicated proof). I thank Ryan Williams for encouraging me to write the survey, for pointing out that Theorem 2.6 is a strict generalization of his result [Wil13], and for providing good writing advice. I'm grateful to Avi Wigderson for several useful comments and additions, and in particular for pointing out a flaw that existed in the definitions in an early draft. I thank Lijie Chen for very useful comments, and for suggesting the same elementary proof of Theorem 2.1 suggested by Oded. And I'm grateful to William Hoza for sharing his proof of Theorem D.1 and for his permission to include it in the survey.

References

[Aar16] Scott Aaronson. "$P \stackrel{?}{=} NP$". In: *Open Problems in Mathematics*. Ed. by John Forbes Nash Jr. and Michael Th. Rassias. Springer International Publishing, 2016, pp. 1–122.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[Adl78] Leonard Adleman. "Two theorems on random polynomial time". In: *Proc. 19th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1978, pp. 75–83.

[And87] A. E. Andreev. "On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes". In: *Vestnik Moskovskogo Universiteta. Seriya I. Matematika, Mekhanika* 1 (1987), pp. 70–73, 103.

[ASW15] Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. "Reed-Muller codes for random erasures and errors". In: *IEEE Transactions on Information Theory* 61.10 (2015), pp. 5229–5252.

[AW85] Miklos Ajtai and Avi Wigderson. "Deterministic simulation of probabilistic constant depth circuits". In: *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1985.

[BBL92] Paul Beame, Erik Brisson, and Richard Ladner. "The complexity of computing symmetric functions using threshold circuits". In: *Theoretical Computer Science* 100.1 (1992), pp. 253–265.

[BHS08] Markus Bläser, Moritz Hardt, and David Steurer. "Asymptotically Optimal Hitting Sets Against Polynomials". In: *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part I*. Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP). 2008, pp. 345–356.

[Bog05] Andrej Bogdanov. "Pseudorandom generators for low degree polynomials". In: *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*. 2005, pp. 21–30.

[Bog18] Andrej Bogdanov. "Small Bias Requires Large Formulas". In: *Proc. 45th International Colloquium on Automata, Languages and Programming (ICALP)*. 2018, 22:1–22:12.

[BSV14] Eli Ben-Sasson and Emanuele Viola. "Short PCPs with projection queries". In: *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*. 2014, pp. 163–173.

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. "Computational analogues of entropy". In: *Proc. 7th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2003, pp. 200–215.

[BV10] Andrej Bogdanov and Emanuele Viola. "Pseudorandom bits for polynomials". In: *SIAM Journal of Computing* 39.6 (2010), pp. 2464–2486.

[Che19] Lijie Chen. "Non-deterministic Quasi-Polynomial Time is Average-case Hard for ACC Circuits". In: *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2019.

[CJW20] Lijie Chen, Ce Jin, and Richard Ryan Williams. "Sharp threshold results for computational complexity". In: *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 2020, 1335–1348.

[CL16] Kuan Cheng and Xin Li. "Randomness Extraction in AC0 and with Small Locality". In: *Electronic Colloquium on Computational Complexity: ECCC* 23 (2016), p. 18.

[CLW20] Lijie Chen, Xin Lyu, and Richard Ryan Williams. "Almost-Everywhere Circuit Lower Bounds from Non-Trivial Derandomization". In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020.

[CR20] Lijie Chen and Hanlin Ren. "Strong average-case lower bounds from non-trivial derandomization". In: *Proc. 52th Annual ACM Symposium on Theory of Computing (STOC)*. 2020, pp. 1327–1334.

[CSS16] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. "Average-case lower bounds and satisfiability algorithms for small threshold circuits". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. 2016, 1:1–1:35.

[CT19] Lijie Chen and Roei Tell. "Bootstrapping results for threshold circuits "just beyond" known lower bounds". In: *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 34–41.

[CT21a] Lijie Chen and Roei Tell. "Hardness vs Randomness, Revised: Uniform, Non-Black-Box, and Instance-Wise". In: *Electronic Colloquium on Computational Complexity: ECCC* 28 (2021), p. 080.

[CT21b] Lijie Chen and Roei Tell. "Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost". In: *Proc. 53st Annual ACM Symposium on Theory of Computing (STOC)*. 2021.

[CTS13] Gil Cohen and Amnon Ta-Shma. "Pseudorandom Generators for Low Degree Polynomials from Algebraic Geometry Codes". In: *Electronic Colloquium on Computational Complexity: ECCC* 20 (2013), p. 155.

[CW19] Lijie Chen and R. Ryan Williams. "Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity". In: *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 2019, 19:1–19:43.

[CW89] Aviad Cohen and Avi Wigderson. "Dispersers, deterministic amplification, and weak random sources". In: *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 14–19.

[DGJ+10] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. "Bounded independence fools halfspaces". In: *SIAM Journal of Computing* 39.8 (2010), pp. 3441–3462.

[DKS+13] Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. "Extensions to the method of multiplicities, with applications to Kakeya sets and mergers". In: *SIAM Journal of Computing* 42.6 (2013), pp. 2305–2328.

[DM18] Irit Dinur and Or Meir. "Toward the KRW composition conjecture: cubic formula lower bounds via communication complexity". In: *Computational Complexity* 27.3 (2018), pp. 375–462.

[DMO+20] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. "Nearly Optimal Pseudorandomness From Hardness". In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020.

[DTST20] Dean Doron, Amnon Ta-Shma, and Roei Tell. "On hitting-set generators for polynomials that vanish rarely". In: *Proc. 24th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2020, Art. 7–22.

[FSU+13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. "On beating the hybrid argument". In: *Theory of Computing* 9 (2013), pp. 809–843.

[GHR92] Mikael Goldmann, Johan Håstad, and Alexander Razborov. "Majority gates vs. general weighted threshold gates". In: *Proc. 7th Annual Structure in Complexity Theory Conference*. 1992, pp. 2–13.

[Gil74] John T. Gill III. "Computational complexity of probabilistic Turing machines". In: *Proc. 6th Annual ACM Symposium on Theory of Computing (STOC)*. 1974, pp. 91–95.

[GK98] Mikael Goldmann and Marek Karpinski. "Simulating Threshold Circuits by Majority Circuits". In: *SIAM Journal of Computing* 27.1 (1998), pp. 230–246.

[GMR13] Parikshit Gopalan, Raghu Meka, and Omer Reingold. "DNF sparsification and a faster deterministic counting algorithm". In: *Computational Complexity* 22.2 (2013), pp. 275–310.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. New York, NY, USA: Cambridge University Press, 2008.

[Gol11] Oded Goldreich. "In a World of P=BPP". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*. 2011, pp. 191–232.

[GTN19] Anna Gál, Avishay Tal, and Adrian Trejo Nuñez. "Cubic formula size lower bounds based on compositions with majority". In: *Proc. 10th Conference on Innovations in Theoretical Computer Science (ITCS)*. 2019, Art. No. 35, 13.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. "Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes". In: *Journal of the ACM* 56.4 (2009), Art. 20, 34.

[GVW15] Oded Goldreich, Emanuele Viola, and Avi Wigderson. "On Randomness Extraction in AC0". In: *Proc. 30th Annual IEEE Conference on Computational Complexity (CCC)*. 2015, pp. 601–668.

[GW14] Oded Goldreich and Avi Widgerson. "On derandomizing algorithms that err extremely rarely". In: *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*. Full version available online at *Electronic Colloquium on Computational Complexity: ECCC*, 20:152 (Rev. 2), 2013. 2014, pp. 109–118.

[Hås14] Johan Håstad. "On the correlation of parity and small-depth circuits". In: *SIAM Journal of Computing* 43.5 (2014), pp. 1699–1708.

[Hås87] Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, 1987.

[Hås98] Johan Håstad. "The shrinkage exponent of De Morgan formulas is 2". In: *SIAM Journal of Computing* 27.1 (1998), pp. 48–64.

[HHT+21] Pooya Hatami, William M. Hoza, Avishay Tal, and Roei Tell. "Fooling Constant-Depth Threshold Circuits". In: *Electronic Colloquium on Computational Complexity: ECCC* 28 (2021), p. 002.

[HIL+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal of Computing* 28.4 (1999), pp. 1364–1396.

[Hoz21] William M. Hoza. Private Communication. 2021.

[IK17] Russell Impagliazzo and Valentine Kabanets. "Fourier concentration from shrinkage". In: *Computational Complexity* 26.1 (2017), pp. 275–321.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "In search of an easy witness: exponential time vs. probabilistic polynomial time". In: *Journal of Computer and System Sciences* 65.4 (2002), pp. 672–694.

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. "Pseudorandomness from shrinkage". In: *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2012, pp. 111–119.

[IN93] Russell Impagliazzo and Noam Nisan. "The effect of random restrictions on formula size". In: *Random Structures & Algorithms* 4.2 (1993), pp. 121–133.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. "Size-depth tradeoffs for threshold circuits". In: *SIAM Journal of Computing* 26.3 (1997), pp. 693–707.

[IW99] Russell Impagliazzo and Avi Wigderson. "P = BPP if E requires exponential circuits: derandomizing the XOR lemma". In: *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*. 1999, pp. 220–229.

[Kel21] Zander Kelley. "An Improved Derandomization of the Switching Lemma". In: *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 2021.

[Khr71] V. M. Khrapčenko. "A certain method of obtaining estimates from below of the complexity of π-schemes". In: *Matematicheskie Zametki* 10 (1971), pp. 83–92.

[KI04] Valentine Kabanets and Russell Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds". In: *Computational Complexity* 13.1-2 (2004), pp. 1–46.

[KL18] Valentine Kabanets and Zhenjian Lu. "Satisfiability and derandomization for small polynomial threshold circuits". In: *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2018, Art. No. 46, 19.

[KLP12] Tali Kaufman, Shachar Lovett, and Ely Porat. "Weight distribution and list-decoding size of Reed-Muller codes". In: *IEEE Transactions on Information Theory* 58.5 (2012), pp. 2689–2696.

[KM02] Adam R. Klivans and Dieter van Melkebeek. "Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses". In: *SIAM Journal of Computing* 31.5 (2002), pp. 1501–1526.

[KR13] Ilan Komargodski and Ran Raz. "Average-case lower bounds for formula size". In: *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC)*. 2013, pp. 171–180.

[KRT17] Ilan Komargodski, Ran Raz, and Avishay Tal. "Improved average-case lower bounds for De Morgan formula size: matching worst-case lower bound". In: *SIAM Journal of Computing* 46.1 (2017), pp. 37–57.

[KS01] Adam R. Klivans and Daniel Spielman. "Randomness efficient identity testing of multivariate polynomials". In: *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*. 2001, pp. 216–223.

[Li16] Xin Li. "Improved two-source extractors, and affine extractors for polylogarithmic entropy". In: *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2016, pp. 168–177.

[Lov09] Shachar Lovett. "Unconditional pseudorandom generators for low-degree polynomials". In: *Theory of Computing* 5 (2009), pp. 69–82.

[LRV+03] Chi-Jen Lu, Omer Reingold, Salil Vadhan, and Avi Wigderson. "Extractors: Optimal up to Constant Factors". In: *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*. 2003, pp. 602–611.

[Lu12] Chi-Jen Lu. "Hitting set generators for sparse polynomials over any finite fields". In: *Proc. 27th Annual IEEE Conference on Computational Complexity (CCC)*. 2012, pp. 280–286.

[LV98] Daniel Lewin and Salil Vadhan. "Checking polynomial identities over any field: towards a derandomization?" In: *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*. 1998, pp. 438–447.

[LVW93] M. Luby, B. Velickovic, and A. Wigderson. "Deterministic approximate counting of depth-2 circuits". In: *Proc. 2nd Israel Symposium on Theory and Computing Systems*. 1993, pp. 18–24.

[MW18] Cody Murray and Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP". In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018.

[Nis91] Noam Nisan. "Pseudorandom bits for constant depth circuits". In: *Combinatorica* 11.1 (1991), pp. 63–70.

[NN93] Joseph Naor and Moni Naor. "Small-bias probability spaces: efficient constructions and applications". In: *SIAM Journal of Computing* 22.4 (1993), pp. 838–856.

[NW15] Zipei Nie and Anthony Y. Wang. "Hilbert functions and the finite degree Zariski closure in finite field combinatorial geometry". In: *Journal of Combinatorial Theory. Series A* 134 (2015), pp. 196–220.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[PS94] Ramamohan Paturi and Michael E. Saks. "Approximating threshold circuits by rational functions". In: *Information and Computation* 112.2 (1994), pp. 257–272.

[PZ93] Michael S. Paterson and Uri Zwick. "Shrinkage of De Morgan formulae under restriction". In: *Random Structures & Algorithms* 4.2 (1993), pp. 135–150.

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. "Extracting all the randomness and reducing the error in Trevisan's extractors". In: *Journal of Computer and System Sciences* 65.1 (2002), pp. 97–128.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. "Bounds for dispersers, extractors, and depth-two superconcentrators". In: *SIAM Journal of Computing* 13.1 (2000), pp. 2–24.

[San10] Rahul Santhanam. "Fighting perebor: new and improved algorithms for formula and QBF satisfiability". In: *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 183–192.

[Ser07] Rocco A. Servedio. "Every linear threshold function has a low-weight approximator". In: *Computational Complexity* 16.2 (2007), pp. 180–209.

[Sip86] Michael Sipser. "Expanders, randomness, or time versus space". In: *Proc. Conference on Structure in Complexity Theory*. 1986, pp. 325–329.

[ST17] Rocco Servedio and Li-Yang Tan. "Deterministic search for CNF satisfying assignments in almost polynomial time". In: *Proc. 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2017.

[ST18] Rocco A. Servedio and Li-Yang Tan. "Luby-Veličković-Wigderson revisited: improved correlation bounds and pseudorandom generators for depth-two circuits". In: *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. Vol. 116. 2018, Art. No. 56, 20.

[ST19] Rocco A. Servedio and Li-Yang Tan. "Improved pseudorandom generators from pseudorandom multi-switching lemmas". In: *Proc. 23rd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2019, Art. No. 45, 23.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. "Pseudorandom generators without the XOR lemma". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 236–266.

[SU05] Ronen Shaltiel and Christopher Umans. "Simple extractors for all min-entropies and a new pseudorandom generator". In: *Journal of the ACM* 52.2 (2005), pp. 172–216.

[Sub61] B. A. Subbotovskaja. "Realization of linear functions by formulas using $\vee$, &, $-$". In: *Soviet Mathematics. Doklady* 2 (1961), pp. 110–112.

[SW13] Rahul Santhanam and Ryan Williams. "On medium-uniformity and circuit lower bounds". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 2013, pp. 15–23.

[Tal14] Avishay Tal. "Shrinkage of De Morgan formulae by spectral techniques". In: *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2014, pp. 551–560.

[Tal17a] Avishay Tal. "Formula lower bounds via the quantum method". In: *Proc. 49th Annual ACM Symposium on Theory of Computing (STOC)*. 2017, pp. 1256–1268.

[Tal17b] Avishay Tal. "Tight Bounds on the Fourier Spectrum of AC0". In: *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. 2017, 15:1–15:31.

[Tel17] Roei Tell. "A Note on the Limitations of Two Black-Box Techniques in Quantified Derandomization". In: *Electronic Colloquium on Computational Complexity: ECCC* 24 (2017), p. 187.

[Tel18] Roei Tell. "Quantified Derandomization of Linear Threshold Circuits". In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 855–865.

[Tel19] Roei Tell. "Improved bounds for quantified derandomization of constant-depth circuits and polynomials". In: *Computational Complexity* 28.2 (2019), pp. 259–343.

[Tel20] Roei Tell. *On implications of better sub-exponential lower bounds for AC0*. Accessed at https://sites.google.com/site/roeitell/Expositions, June 22, 2021. 2020.

[Tod91] Seinosuke Toda. "PP is as hard as the polynomial-time hierarchy". In: *SIAM Journal of Computing* 20.5 (1991), pp. 865–877.

[Tre01] Luca Trevisan. "Extractors and Pseudorandom Generators". In: *Journal of the ACM* 48.4 (2001), pp. 860–879.

[TSUZ07] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. "Lossless condensers, unbalanced expanders, and extractors". In: *Combinatorica* 27.2 (2007), pp. 213–240.

[TX13] Luca Trevisan and TongKe Xue. "A derandomized switching lemma and an improved derandomization of AC0". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 2013, pp. 242–247.

[Uma03] Christopher Umans. "Pseudo-random generators for all hardnesses". In: *Journal of Computer and System Sciences* 67.2 (2003), pp. 419–440.

[Vad12] Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

[Vio05] Emanuele Viola. "The complexity of constructing pseudorandom generators from hard functions". In: *Computational Complexity* 13.3-4 (2005), pp. 147–188.

[Vio09a] Emanuele Viola. "On approximate majority and probabilistic time". In: *Computational Complexity* 18.3 (2009), pp. 337–375.

[Vio09b] Emanuele Viola. "The sum of d small-bias generators fools polynomials of degree d". In: *Computational Complexity* 18.2 (2009), pp. 209–217.

[War35] Ewald Warning. "Bemerkung zur vorstehenden Arbeit von Herrn Chevalley". In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 11 (1935), pp. 76–83.

[Wil11] Ryan Williams. "Non-uniform ACC circuit lower bounds". In: *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*. 2011, pp. 115–125.

[Wil13] Ryan Williams. "Improving Exhaustive Search Implies Superpolynomial Lower Bounds". In: *SIAM Journal of Computing* 42.3 (2013), pp. 1218–1244.

[Wil14] Ryan Williams. "Algorithms for circuits and circuits for algorithms: Connecting the tractable and intractable". In: *Proc. International Congress of Mathematicians (ICM)*. 2014, pp. 659–682.

Appendix A Error-reduction by itself is not enough

Can we construct a better-than-brute-force algorithm for CAPP via the naive approach of first reducing CAPP to QD_B using a standard sampler-based error-reduction, and then using a brute-force algorithm for QD_B (i.e., solving quantified derandomization by evaluating the given circuit over some fixed $O(B(n))$ inputs)?

The following result shows a negative answer to this question: Any such algorithm will be slower than the brute-force algorithm that simply evaluates the original circuit on all of its inputs. The meaning of this result is that when constructing CAPP algorithms that are based on an initial step of sampler-based error-reduction, a *non-trivial algorithm for quantified derandomization is necessary*. The statement below shows that this is the case even for derandomization with one-sided error (i.e., for $\mathsf{CAPP}_{\frac{1}{2},0}$) and even when using dispersers rather than samplers.

Definition A.1 (disperser)**.** *A function* $\mathsf{Disp}\colon \{0,1\}^{\bar{n}} \times \{0,1\}^{\ell} \to \{0,1\}^{n}$ *is a* (k,ϵ)-disperser *if for every* $T \subseteq \{0,1\}^{n}$ *of density* $|T|/2^{n} \geq \epsilon$*, for all but at most* 2^{k} *strings* $z \in \{0,1\}^{\bar{n}}$ *there exists* $s \in \{0,1\}^{\ell}$ *such that* $\mathsf{Disp}(z,s) \in T$*.*

Theorem A.2 (disperser-based error-reduction should be coupled with non-trivial algorithms for quantified derandomization)**.** *Consider the following algorithm for* $\mathsf{CAPP}_{\frac{1}{2},0}$*. Given an* n*-bit circuit* C*, let* $\mathsf{Disp}\colon \{0,1\}^{\bar{n}} \times \{0,1\}^{\ell} \to \{0,1\}^{n}$ *be an arbitrary* $(k,.01)$*-disperser for some value of* $k \leq n$*. The algorithm:*

1. *Constructs the* $\bar{n}$*-bit circuit* C' *such that* $C'(z) = \bigvee_{s\in\{0,1\}^{\ell}} C(\mathsf{Disp}(z,s))$*.*
2. *Evaluates* C' *over (arbitrary) fixed* $2^{k}+1$ *inputs.*
3. *Outputs "yes" if and only if* C' *accepted one of the inputs.*

Then, the running time of this algorithm is larger than $2^{n} \cdot \tilde{O}(|C|)$*.*

Proof. Radhakrishnan and Ta-Shma [RTS00] proved that for any $(k,.01)$-disperser $\mathsf{Disp}\colon \{0,1\}^{\bar{n}} \times \{0,1\}^{\ell} \to \{0,1\}^{n}$ it holds that $n \leq k + \ell - O(1)$ (i.e., an entropy loss is inherent). Also note that the size of C' is more than $2^{\ell} \cdot |C|$, even without taking into account the complexity of Disp. Thus, the running time of the algorithm is $(2^{k}+1) \cdot \tilde{O}(|C'|) > 2^{k+\ell} \cdot \tilde{O}(|C|) \geq 2^{n} \cdot \tilde{O}(|C|)$. ∎

Appendix B Pseudorandom restrictions for low-depth circuits and formulas

In this section I describe the technical results underlying the algorithms for quantified derandomization that were presented in Section 6. These technical results assert the existence of *efficient pseudorandom restriction procedures* that yield simplifier sets, in the sense of Definition 7.1.

B.1 Width-dependent derandomization of Håstad's switching lemma

Let me start with the class $\mathcal{AC}^0$. Using standard techniques following [Hås87], the problem of constructing a pseudorandom restriction procedure reduces to the problem of derandomizing Håstad's switching lemma [Hås87]; that is, to the problem of constructing a pseudorandom distribution of restrictions that simplifies every depth-2 formula into a decision tree of bounded depth, with high probability (see, e.g., [Tel19, Proof of Theorem 5.16] for an explanation).[40]

Note that for our application (i.e., to construct an algorithm for QD_B) we want to pseudorandomly choose *both the variables to fix and the values for fixed variables*. This should be distinguished from applications for which we only need to pseudorandomly choose which variables to fix, while leaving the choice of values to be completely uniform. (A very recent result of Kelley [Kel21] showed that the latter task can be solved in polynomial time with seed length $O(\log(n))$.)

To optimize the trade-off between $B(n)$ and the seed length, we will be interested in derandomization of the switching lemma for depth-two formulas of *bounded width* (see [Tel19] for an explanation of why this is the case). We denote the formula size by $m \geq n$ and its width by w, and for our application we can assume wlog that $w \leq O(\log(m))$ and we fix the error probability to be $1/\mathrm{poly}(m)$ for a sufficiently large polynomial.

For such parameters, Trevisan and Xue [TX13] constructed a pseudorandom restriction algorithm with seed length $\tilde{O}(w) \cdot \log^2(m)$, and Goldreich and Wigderson [GW14] constructed such an algorithm with seed length $\tilde{O}(2^w) \cdot \log(m)$. The following result from [Tel19] improved on both these results by constructing a pseudorandom restriction algorithm with seed length $\tilde{O}(w^2 \cdot \log(m))$: [41]

Proposition B.1 (width-dependent derandomization of Håstad's switching lemma; see [Tel19, Theorem 1.4])**.** *Let $m, n \in \mathbb{N}$, let $w \leq O(\log(m))$, and let $\delta = \delta(n) > 0$. Then, there exists an algorithm that gets as input a random seed of length $\tilde{O}(w^2 \cdot \log(mn/\delta))$, runs in time $\mathrm{poly}(n)$, and outputs a restriction $\rho \in \{0,1,\star\}^n$ such that for every n-bit depth-2 formula F of size m and width w, with probability $1 - O(\delta)$ the following holds:*

1. *The number of variables kept alive by ρ is $\Omega(n/w)$.*
2. *There exist "lower-sandwiching" and "upper-sandwiching" formulas $F^{\mathtt{low}}$ and $F^{\mathtt{high}}$ for F [42] such that both $F^{\mathtt{low}}{\restriction_\rho}$ and $F^{\mathtt{high}}{\restriction_\rho}$ can be computed by decision trees of depth $O(\log(1/\delta))$, and each of the two formulas agrees with $F{\restriction_\rho}$ on $1 - \delta$ of the inputs.*

Proposition B.1 is the main technical result underlying the algorithm for QD_B in Theorem 2.7. Observe that, crucially, both $F^{\mathtt{low}}$ and $F^{\mathtt{high}}$ agree with F on $1 - \delta$ of

[40] We will focus on pseudorandom distributions that achieve the same bound on the decision tree depth, and approximately the same error probability, as in Håstad's original result [Hås87]. Pseudorandom restriction procedures that achieve worse parameters but are more efficient are known (these date back to [AW85], with a recent construction presented in [GMR13]).

[41] Strictly speaking, the result of [GW14] is still better in the case of $w = O(1)$, since it yields seed length $O(\log(n))$ rather than $\tilde{O}(\log(n))$.

[42] That is, for every $x \in \{0,1\}^n$ it holds that $F^{\mathtt{low}}(x) \leq F(x) \leq F^{\mathtt{high}}(x)$.

the inputs *in the subcube that corresponds to* ρ; that is, they approximate F *after* the restriction. Also, we can take δ to be an arbitrarily large polynomial in m without noticeably affecting the seed length.

B.2 Pseudorandom restrictions for threshold circuits

For constant-depth linear threshold circuits (LTF circuits), even *random* restriction procedures (let alone pseudorandom procedures) are relatively new. Impagliazzo, Paturi, and Saks [IPS97] showed a random restriction procedure in which neither the fixed variables nor their values are chosen uniformly; this procedure sufficed to show worst-case lower bounds, but does not suffice for many applications, such as proving average-case lower bounds or constructing quantified derandomization algorithms.

Several years ago Chen, Santhanam, and Srinivasan [CSS16] (relying on results developed in [Ser07; DGJ+10] and other works) showed a random restriction procedure for LTF circuits in which the variables are chosen in an adaptive way that depends on the given circuit, but values for fixed variables are chosen uniformly; they used this procedure to deduce average-case lower bounds for LTF circuits. This restriction procedure was subsequently derandomized and refined in [Tel18], yielding the following result, which is the main technical result underlying Theorem 2.8:

Proposition B.2 (pseudorandom restrictions for LTF circuits; see [Tel18, Proposition 3.1])**.** *Let* $c, d \geq 1$*, let* $\epsilon > 0$ *be a sufficiently small constant, and let* $\delta = d \cdot 30^{d-1} \cdot \epsilon$*. Then, there exists a polynomial-time algorithm that for every* $n \in \mathbb{N}$*, when given as input an* LTF *circuit over* n *input bits of depth* d *with at most* $n^{1+\epsilon}$ *wires, and a random seed of length* $O(\log(n) \cdot \mathrm{loglog}(n))$*, with probability at least* $1 - n^{-\epsilon/2}$ *outputs the following:*

1. *A restriction* ρ *that keeps at least* $n^{1-\delta}$ *variables alive.*
2. *An* LTF *that is* $(1 - n^{-c})$*-close to* $C\restriction_\rho$*.*

Note that the original statement in [Tel18] only asserts that Φ is $(9/10)$-close to $C\restriction_\rho$, but the proof already shows that the closeness is $1 - n^{-c}$ for every desired constant $c \in \mathbb{N}$. (To see this, note that in Claim 5.11.1 of the full version, the bound on the closeness of each biased gate to the corresponding constant after all the restriction is stated to be $\delta_t = 1 - n^{-c}$ for an arbitrary constant $c \in \mathbb{N}$.)

Let me also note that another pseudorandom restriction procedure for LTF circuits was very recently shown by Hatami *et al.* [HHT+21]. In this procedure the failure probaiblity is $\exp(-n^{\Omega(1)})$ instead of $n^{-\Omega(1)}$, but only the variables to be fixed are chosen pseudorandomly, whereas values for fixed variables are chosen uniformly.

Kabanets and Lu [KL18] showed a result analogous to Proposition B.2 that holds for the stronger class of PTF circuits of low degree; this is the main technical result underlying the algorithm for QD_B of PTF circuits in Theorem 6.12. They also showed a similar result for PTF circuits in which each gate computes a sparse polynomial (i.e., a polynomial with n^Δ monomials for a small constant Δ).

Proposition B.3 (pseudorandom restrictions for low-degree PTF circuits; see [KL18, Proof of Theorem 4.4]). *Let $c, d \geq 1$, let $E \geq 11$, and let $\Delta\colon \mathbb{N} \to \mathbb{N}$ such that $\Delta \ll \sqrt{\epsilon_d \cdot \log(n)/\mathrm{loglog}(n)}$, where $\epsilon_d = E^{-2(d-1)}$. Let $\mathcal{C}_n$ be the class of* PTF *circuits over n input bits of depth d with $n^{1+\epsilon_d}$ wires in which each gate computes a* PTF *with degree at most $\Delta(n)$. Then, there exists an algorithm that gets as input $C \in \mathcal{C}_n$ and a random seed of length $\log(n)^{O(\Delta(n)^2)}$, and with probability at least $1 - n^{\Omega(1)}$ outputs the following:*

1. *A restriction ρ that keeps at least $n^{1-6/E}$ variables alive.*
2. *A* PTF *with at most $n^{\epsilon_d \cdot \Delta(n)}$ monomials that is $(1 - n^{-c})$-close to $C|_\rho$.*

Proposition B.3 is not explicitly stated in [KL18] (which is a conference version), but as explained there after the statement of Theorem 4.7, this result follows immediately by mimicking the proof of Theorem 4.4 (which is an analogous result for PTF circuits in which each gate computes a sparse polynomial). Also, similarly to Proposition B.2, in [KL18] the closeness parameter is taken to be $9/10$ rather than $1 - n^{-c}$, but the latter value is immediate from their proof. (To see this, in the proof of Theorem 4.4, instantiate Lemma 4.5 with an arbitrarily large constant $c \geq 1$ instead of with $c = 1$.)

The restriction procedures are non-black-box. The algorithms in Propositions B.2 and B.3 both work in a non-black-box fashion: They get as input a circuit C, and tailor a restriction that is specifically designed to simplify C. However, as mentioned in Section 7, a key component in these procedures is already "somewhat black-box" (i.e., going layer-by-layer, these restrictions are pseudorandom distributions that simplify each of the gates in the layer with high marginal probability). Moreover, both procedures can be made fully black-box at the expense of simplifying the circuit not to a single LTF or PTF, but rather to the more complicated model of a relatively shallow decision tree with LTFs or PTFs at its leaves; see [HHT+21] for an explanation.

B.3 Pseudorandom restrictions for formulas

Random restrictions for De Morgan formulas have been extensively studied since the 1960's, focusing on the well-known implication that a formula is expected to shrink (in size) under such restrictions (see, e.g., [Sub61; PZ93; IN93; Hås98; Tal14]). However, only in the last decade have *pseudorandom versions* been constructed.

Impagliazzo, Meka, and Zuckerman [IMZ12] constructed a pseudorandom restriction procedure that shrinks any formula of size s to be of size $p^2 \cdot s^{1+o(1)}$, with probability $1 - n^{-O(1)}$; this procedure has seed length $2^{O(\log^{2/3}(s))} = s^{o(1)}$. Hatami *et al.* [HHT+21] showed a pseudorandom retriction procedure that supports a much smaller failure probability $\epsilon \ll s^{-O(1)}$, but shrinks any formula to a decision tree of depth $s^{o(1)} \cdot \mathrm{polylog}(1/\epsilon)$ with formulas of size $p^{2-o(1)} \cdot s$ at its leaves; the seed length for this procedure is $s^{o(1)} \cdot \mathrm{polylog}(n/\epsilon)$.

For quantified derandomization we do not need the strong concentration bounds above on the shrinkage of the formula, and shrinkage in expectation suffices. For

this application, Chen, Jin, and Williams [CJW20] showed a procedure that uses seed length only $O(\log(n))$ and indeed obtains shrinkage in expectation:

Proposition B.4 (pseudorandom restrictions for formulas; see [CJW20])**.** *Let $p\colon \mathbb{N} \to \mathbb{N}$. Then, there exists an algorithm that gets as input a random seed of length $O(\log(n))$, runs in time $\mathrm{poly}(n)$, and outputs a restriction $\rho \in \{0,1,\star\}^n$ such that:*

1. *With probability at least $2/3$ it holds that ρ keeps at least $pn/2$ variables alive.*
2. *For every n-variable formula it holds that*

$$\mathbb{E}[L(F\restriction_\rho)] \leq \left(p^2 \cdot L(F) + p \cdot \sqrt{L(F)}\right) \cdot n^{c/\mathrm{loglog}(n)} ,$$

where $c > 1$ is a universal constant.

Proposition B.4 is the main technical result underlying the algorithm for QD_B of formulas in Theorem 2.10. In addition, the pseudorandom restriction in [CJW20] is even stronger, since it guarantees the existence of a circuit C of size $\mathrm{polylog}(n)$ that gets as input the random seed (of length $O(\log(n))$) and an index $i \in [n]$ of an output, an prints the i^{th} coordinate of the restriction ρ.

Appendix C Extractors computable by low-depth circuits and formulas

In this section I describe the technical results underlying the reductions of CAPP to QD_B that were presented in Section 6. These technical results are constructions of extractors that are computable in *weak circuit classes*. The precise notion of being computable in a weak circuit class will differ across the constructions presented below, but in general it will be at least as strict as the one in Definition 7.2 (and hence the limitation in Theorem 7.3 applies to the results that use these constructions).

In general, there are very efficient constructions of extractors with good parameters: For example, each output bit of Trevisan's [Tre01] extractor (and of its improvement in [RRV02]) is just a parity of the input. However, in the following results we will be interested in computing extractors by circuits or formulas *that are too weak to even compute the parity* of their input.

C.1 Extractors computable by $\mathcal{AC}^0$ circuits

Goldreich and Wigderson [GW14, Theorem 3.4 in the full version] constructed an $\mathcal{AC}^0$ circuit computing a function that can be thought of as a middle-point between a standard extractor (which outputs a distribution close to uniform) and a non-black-box extractor as referred to in Section 7.3 (which outputs a distribution that only looks uniform to a circuit whose description is given to the non-black-box extractor). Specifically, the output distribution of their function looks uniform to any $\mathcal{AC}^0$ observer; this

is equivalent to a sampler that only samples correctly subsets that are decidable by $\mathcal{AC}^0$ circuits. Their function was computable by $\mathcal{P}$-uniform $\mathcal{AC}^0$ circuits, had $n_0 = n^{\Omega(1)}$ output bits, and supported min-entropy $k = 2^{n/\mathrm{polylog}(n)}$.

Their construction was later superseded by a construction of standard extractors that are computable by $\mathcal{P}$-uniform $\mathcal{AC}^0$ circuits, which was shown by Cheng and Li [CL16]. (That is, the construction of [CL16] is of a standard extractor rather than of a non-black-box one, and also has better parameters than the one in [GW14].) In fact, there are various different such constructions in [CL16], supporting different trade-offs between the parameters; let me mention one such construction of theirs:

Proposition C.1 (extractors in uniform $\mathcal{AC}^0$; see [CL16, Theorem 4.11])**.** *For any $d \geq 7$ there exists an extractor family $\left\{\mathsf{Ext}_n \colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^{n_0}\right\}_{n\in\mathbb{N}}$ with seed length $\ell = O(\log(n))$, output length $n_0 = \left\lfloor n^{1/3600} \right\rfloor$, min-entropy $k = \Theta(n/\log^{d-7}(n))$, and error $n^{-1/600}$, such that the function mapping $(z,s) \in \{0,1\}^n \times \{0,1\}^\ell$ to $\mathsf{Ext}_n(z,s)$ is computable by $\mathcal{P}$-uniform $\mathcal{AC}^0$ circuits of depth d and size $\mathrm{poly}(n)$.*

The parameters of Proposition C.1 are close to the best possible (and various optimizations and tradeoffs appear in [CL16]). This follows from a lower bound of Viola [Vio05] (see also [GVW15]), which asserts that $\mathcal{AC}^0$ circuits of size $\mathrm{poly}(n)$ and depth d can compute extractors for min-entropy at most $k = n/\log^{d-1}(n)$, even if the seed is very long compared to the output length (i.e., even if the seed is of length n_0^{999}). A similar lower bound follows by combining Theorem 7.3 with Håstad's switching lemma [Hås87]. (In fact, Theorem 7.3 yields a more general approach for showing such lower bounds, since the simplifier set need not be a subcube and may even partially depend on the circuit that it simplifies (as explained in Section 7).)

C.2 Extractors computable by extremely sparse threshold circuits

Recall that the parity function can be computed by LTF circuits of depth d and size $n^{1+c_\oplus^{-d}}$, for some constant $c_\oplus \geq \frac{1+\sqrt{5}}{2}$ (see [BBL92; PS94]). Thus, if we instantiate Trevisan's [Tre01] extractor Ext with seed length close to $\log(n)$ and output length n^ϵ for a small constant $\epsilon > 0$, we can compute the mapping $z \mapsto \{\mathsf{Ext}(z,s)\}_s$ by a uniform $\mathcal{TC}^0$ circuit of super-quadratic size. (This is since this extractor only computes parities of the input, and since for these parameters the circuit that prints the outputs of the extractor on all seeds has $n^{1+O(\epsilon)}$ output bits.)

As far as I know, the first extractor that is computable by uniform $\mathcal{TC}^0$ circuits of *super-linear* size was constructed in [Tel18]; each output bit of this extractor is still a parity of the input, but these parities are computed "in a batch" rather than paying $n^{1+c_\oplus^{-d}}$ per each output bit. This construction was later improved by Chen and the current author [CT19], who showed a construction with seed length and output length as above that uses only $n^{1+c^{-d}}$ wires, for any $c < c_\oplus$; that is:

Proposition C.2 (extractors in uniform $\mathcal{TC}^0$ of super-linear size)**.** *For any $d \geq 7$ and $c < c_\oplus$ there exists an extractor family $\left\{\mathsf{Ext}_n \colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^{n_0}\right\}_{n\in\mathbb{N}}$ with seed length*

$\ell = (1+\exp(-d)) \cdot \log(n)$, *output length* $n_0 = n^{\exp(-d)}$, *min-entropy* $k = n^{1-\exp(-d)}$, *and error* $\epsilon > 0$, *such that the following holds: The function mapping* $z \in \{0,1\}^n$ *to the output-set* $(\mathsf{Ext}_n(z,s))_{s\in\{0,1\}^\ell}$ *is computable by* $\mathcal{P}$*-uniform* $\mathcal{TC}^0$ *circuits of depth* d *and size* $n^{1+c^{-d}}$.

Note that the circuits in Proposition C.2 are $\mathcal{TC}^0$ circuits rather than LTF circuits; that is, to compute the extractor we only use unweighted majority gates rather than (the stronger) linear threshold functions.

C.3 Dispersers computable by formulas of subquadratic size

Recall that the parity function can be computed by formulas of size $O(n^2)$. Thus, a naive implmentation of Trevisan's extractor with seed length close to $\log(n)$ and output length n^ϵ for a small constant $\epsilon > 0$ yields formulas of size $O(n^{3+O(\epsilon)})$.

The reduction of CAPP to QD_B by Chen, Jin, and Williams [CJW20] yields formulas of *sub-quadratic* size, using two ideas. The first idea is to combine a standard linear extractor with naive error reduction; the addition of naive error reduction yields slightly poorer extraction properties, but also reduces the computational complexity (intuiviely, since naive error reduction has very low complexity but poor extractor properties). In particular, the combination yields the following construction:

Proposition C.3 (dispersers computable by uniform sub-quadratic formulas)**.** *For any* $\epsilon \in (0,1)$ *and* $\delta > 0$ *there exists a family of functions* $\widehat{\mathsf{Disp}}_n \colon \{0,1\}^n \times \{0,1\}^{O(\log(n))} \to \{0,1\}^{n_0}$, *where* $n_0 = n^{\Omega_{\delta,\epsilon}(1)}$, *that satisfies the following:*

1. **Seeds are pairs.** *The seed of* $\widehat{\mathsf{Disp}}$ *is a pair* $(s,i) \in \{0,1\}^{O(\log(n))} \times \{0,1\}^{\epsilon\cdot\log(n)}$.
2. **Computable by formulas of sub-quadratic size:** *For each fixed* $s \in \{0,1\}^{O(\log(n))}$, *the mapping of* $x \in \{0,1\}^n$ *to the tuple* $(\widehat{\mathsf{Disp}}_n(x,(s,i)))_{i\in\{0,1\}^{\epsilon\cdot\log(n)}}$ *is computable by* $\mathcal{P}$*-uniform formulas of size* $n^{2-\epsilon+\delta}$.
3. **Disperser with density** $\Omega(n^{-\epsilon})$**:** *For every* $T \subseteq \{0,1\}^{n_0}$ *such that* $|T|/2^{n_0} \geq 9/10$, *for all but at most* 2^{n^ϵ} *inputs* $x \in \{0,1\}^n$ *there exists* $i \in \{0,1\}^{\epsilon\cdot\log(n)}$ *such that* $\Pr_s\left[\widehat{\mathsf{Disp}}(x,(s,i)) \in T\right] \geq 2/3$.

Proof. For two constants $\alpha > 0$ and $\beta < 1$ that will be defined below, and for $n_1 = n^\beta$, let $\mathsf{Ext} \colon \{0,1\}^{n_1} \times \{0,1\}^{O(\log(n_1))} \to \{0,1\}^{n_0}$ be the extractor that is implicit in the work of Li [Li16, Theorem 3.14] and was explicitly stated in [CJW20, Theorem 4.1], where $n_0 = n_1^{\alpha/2}$; the min-entropy of Ext is n_1^α, its error is $n_1^{-\alpha}$, and it can be computed by $\mathcal{P}$-uniform formulas of size $n_1^{2+\alpha}$. We think of any n-bit string x as a sequence of $r = n/n_1$ disjoint substrings $x_1,\ldots,x_r$ of length n_1, and define $\widehat{\mathsf{Disp}}(x,(s,i)) = \mathsf{Ext}(x_i,s)$; that is, the random seed of $\widehat{\mathsf{Disp}}$ consists of an index $i \in [r]$ and of a seed s for Ext, and $\widehat{\mathsf{Disp}}$ applies Ext with seed s to the i^{th} substring of n_1 bits in its input x.

The seed length of $\widehat{\mathsf{Disp}}$ is $(1-\beta)\cdot\log(n) + O(\log(n))$, and its output length is $n_0 = n^{\beta\cdot\alpha/2}$. Also, for each fixed s, the mapping $x \mapsto (\widehat{\mathsf{Disp}}_n(x,(s,i)))_{i\in[r]}$ is computable by $\mathcal{P}$-uniform formulas of size $r \cdot n_1^{2+\alpha}$. Now, let $T \subseteq \{0,1\}^{n_0}$ be of density

at least $9/10$. For every fixed $i \in [r]$ there exist at most $2^{n_1^\alpha}$ strings $x_i \in \{0,1\}^{n_1}$ such that $\Pr[\mathsf{Ext}(x_i, \mathbf{s}) \in T] < 9/10 - n^{-\alpha}$. Thus, the number of strings $x = (x_1, ..., x_r)$ such that for all $i \in [r]$ it holds that $\Pr[\mathsf{Ext}(x_i, \mathbf{s}) \in T] < 9/10 - n^{-\alpha}$ is at most $2^{n_1^\alpha \cdot r}$. Hence, for all but at most $2^{n_1^\alpha \cdot r}$ of the strings $x \in \{0,1\}^n$ there exists $i \in [r]$ such that $\Pr[\widehat{\mathsf{Disp}}(x, (\mathbf{s}, i)) \in T] = \Pr[\mathsf{Ext}(x_i, \mathbf{s}) \in T] \geq 9/10 - o(1) > 2/3$.

To conclude we need to choose $\alpha > 0$ and $\beta < 1$ such that $n_1^\alpha \cdot r \leq n^\epsilon$ (for the number of exceptional inputs) and $r \cdot n_1^{2+\alpha} \leq n^{2-\epsilon+\delta}$ (for the size bound) and $(1-\beta) \cdot \log(n) < \epsilon \cdot \log(n)$ (for the seed length). Choosing $\beta = \frac{1-\epsilon}{1-\alpha}$ and a sufficiently small $\alpha = \alpha_{\epsilon,\delta} > 0$ suffices. ■

The second idea of [CJW20] is that in their reduction, instead of the standard approach of reducing CAPP of a formula F to QD_B for $F'(x) = \bigvee_{s,i} F(\widehat{\mathsf{Disp}}(x, (s, i)))$, they reduce CAPP of F to QD_B for a *probabilistic formula*, defined as follows:

$$\mathbf{F}(x) = \bigvee_{i \in [r]} F(\widehat{\mathsf{Disp}}(x, (\mathbf{s}, i))) \,,$$

where $\mathbf{s}$ (i.e., the first part of the seed) is the only random choice made by the probabilistic formula $\mathbf{F}$. By Proposition C.3, each formula in the support of $\mathbf{F}$ is of size $n^{2-\epsilon+\delta}$, and if F accepts at least $9/10$ of its inputs, then for all but 2^{n^ϵ} of the inputs x for $\mathbf{F}$ it holds that $\Pr[\mathbf{F}(x) = 1] \geq 2/3$.

The limitation in Theorem 7.3 still applies to this construction. The limitation in Theorem 7.3 is proved under the hypothesis that the distribution of simplifier sets simplifies every circuit in the class (in the current setting this will refer to every formula of bounded size) with probability at least $1/2$. This hypothesis suffices to deduce a limitation on extractor-based construction. In the setting of formulas the known distribution of simplifier sets has a considerably higher success probability (i.e., $1 - n^{-O(1)}$ instead of $1/2$), and thus its existence suffices to deduce a limitation also on disperser-based constructions as in Proposition C.3.

In particular, the following claim asserts that a disperser construction as in Proposition C.3 cannot be computed by formulas of size $n^{2-2\epsilon+o(1)}$ (as in Corollary 6.16) instead of $n^{2-\epsilon+\delta}$. The claim even rules out a weaker disperser construction, in which we do not have a density guarantee (as in Item (3)) and in which only require the disperser to be computable by formulas of the corresponding size on each fixed seed (rather than requiring a batch-computation property as in Item (2)).

Claim C.4. *For any $\epsilon > 0$, there does not exist an $(n^\epsilon, .01)$-disperser $\mathsf{Disp}\colon \{0,1\}^n \times \{0,1\}^{O(\log(n))} \to \{0,1\}^{n_0}$, where $n_0 = n^{\Omega(1)}$, such that for every fixed $s \in \{0,1\}^{O(\log(n))}$ it holds that $\mathsf{Disp}^{(s)}(x) = \mathsf{Disp}(x, s)$ is computable by a formula of size $n^{2-2\epsilon+o(1)}$.*

Proof. Assume towards a contradiction that such construction exists, and let $\varphi = \varphi(\epsilon) > 0$ be a sufficiently small constant. For $p = n^{-1+\epsilon+\varphi}$, let $\mathbf{X}$ be a distribution over subcubes $X \subset \{0,1\}^n$ of size at least $2^{p \cdot n/2} = 2^{n^{\epsilon+\varphi}/2}$ that shrinks every formula of

size S to be of size $p^2 \cdot S^{1+o(1)}$, with probability at least $1 - S^{-c}$ for an arbitrarily large constant $c > 1$ (see [IMZ12, Lemma 4.8]).[43]

Let $\mathcal{F} = \left\{\mathsf{Disp}^{(s)}\right\}_{s\in\{0,1\}^{O(\log(n))}}$. Note that there are $\mathrm{poly}(n)$ functions in $\mathcal{F}$, and each function has $n_0 = n^{\Omega(1)}$ output bits. Taking the constant $c > 1$ in the error bound above to be sufficiently large, there exists $X \sim \mathbf{X}$ such that the formula size of every function $\mathsf{Disp}^{(s)} \in \mathcal{F}$ decreases by a factor of $p^2 \cdot n^{o(1)}$; in particular, each $\mathsf{Disp}^{(s)}$ is computable by a formula of size $p^2 \cdot n^{2-2\epsilon+o(1)} = n^{2\varphi+o(1)}$.[44]

It follows that on the subset X, each function $\mathsf{Disp}^{(s)} \in \mathcal{F}$ is sensitive to less than $n^{2\varphi+o(1)}$ input bits. Hence, the support size of Disp when given inputs from X satisfies

$$\left|\bigcup_{x\in X, s\in\{0,1\}^{O(\log(n))}} \mathsf{Disp}(x,s)\right| \leq \mathrm{poly}(n) \cdot 2^{n^{2\varphi+o(1)}} \leq 2^{n^{2\varphi+o(1)}}.$$

Taking φ to be sufficiently small such that $n^{2\varphi} < \sqrt{n_0}$, there exists a set $T \subseteq \{0,1\}^{n_0}$ of size more than $2^{n_0} - 2^{\sqrt{n_0}} = (1 - o(1)) \cdot 2^{n_0}$ that avoids Disp on a set $X \subseteq \{0,1\}^n$ of size $2^{n^{\epsilon+\Omega(1)}}$, a contradiction to the hypothesized properties of Disp. ∎

Appendix D Quantified derandomization of logspace and of proof systems

In this appendix I mention two interesting directions that were raised in the original work of Goldreich and Wigderson [GW14] but have not been explored further so far.

D.1 Quantified derandomization of logspace

Can we simulate probabilisitic logspace machine in deterministic logspace if the number of exceptional random strings is extremely small? As reported in [GW14], Mike Saks showed in the 1990s that this is indeed possible, even when the number of exceptional random strings is relatively not that small:

Theorem D.1 (quantified derandomization of logspace; attributed to Saks [GW14, Appendix A of the Full Version])**.** *Let $L \subseteq \{0,1\}^*$ be decidable by a probabilistic logspace machine M such that for some constant $\epsilon > 0$, on n-bit inputs M uses $T = T(n)$ bits of randomness and errs on at most $B(T) = 2^{(1-\epsilon)\cdot T}$ random choices. Then, $L \in \mathcal{L}$.*

[43] The subsets in the support of the distribution from [IMZ12] are of size $p \cdot n/2$ only with very high probability (rather than always). I ignore this issue for simplicity, as we can always modify the distribution such that it is supported only on subsets of sufficiently large size $p \cdot n/2$, while preserving the property that each size-S formula is simplified with probability at least $1 - S^{-c}$.

[44] To elaborate, each $\mathsf{Disp}^{(s)}$ is a multi-output function computable by a collection of n_0 formulas. Let $\mathcal{S}$ be the sub-collection of formulas of size less than n^{φ}/n_0, and let $\mathcal{L}$ be the sub-collection of formulas of size at least n^{φ}/n_0. For each $F \in \mathcal{L}$, with probability $1 - 1/\mathrm{poly}(n)$ its size decreased by a multplicative factor of $p^2 \cdot n^{o(1)}$; and the total contribution to size of the formulas in $\mathcal{S}$ is at most n^{φ}. Thus, with probability $1 - 1/\mathrm{poly}(n)$ the size of $\mathsf{Disp}^{(s)}$ after the restriction is at most $p^2 \cdot S^{1+o(1)} + n^{\varphi} \leq n^{2\varphi+o(1)}$.

The number $B(T) = 2^{(1-\Omega(1))\cdot T}$ of exceptional random strings in Theorem D.1 matches the non-uniform derandomization in Theorem 2.1, and is indeed significantly larger than in all other settings in this survey (i.e., in all other settings the number of exceptional random strings was $B(T) = 2^{o(T)}$).

Saks' original quantified derandomization algorithm was non-black-box: Given as input a description of a polynomial-sized read-once branching program (ROBP), the algorithm relies on the description to find its most likely output. (Recall that the ROBP represents the computation of a probabilistic logspace machine on a fixed input as a function of the random coins.) William Hoza [Hoz21] strengthened this result by constructing a *black-box* algorithm (i.e., a PRG for biased ROBPs) that yields the same parameters; the proof below presents Hoza's construction.

Proof of Theorem D.1 by William Hoza. For any $\epsilon = \epsilon(n) > 0$ and any $B(n) \leq \epsilon \cdot 2^n$, we construct an ϵ-PRG for of B-biased ROBPs over n input bits of w, whose seed length is $\ell = \ell(n) = \frac{n}{n-\log(B)} \cdot \log(2nw/\epsilon)$. Given seed $s \in \{0,1\}^\ell$, the PRG simply outputs the n-bit string $(s, s, s, ..., s) \in (\{0,1\}^\ell)^{n/\ell}$ (for simplicity we assume that n/ℓ is an integer). Note that this PRG is indeed computable in logspace, and that for $B(n) = 2^{(1-\Omega(1))\cdot n}$ its seed length satisfies $\ell(n) = O(\log(nw/\epsilon))$.

To see that this construction works, fix an ROBP as above, and let $\sigma \in \{0,1\}$ be its less likely output. Index the layers of the ROBP by $0, ..., n$ where 0 is the layer of the starting vertex and n is the last layer, and consider the vertices at layers indexed $0, \ell, ..., i \cdot \ell, ..., n$. For each such vertex v, denote by p_v the probability that a random walk starting from v reaches a vertex in the last layer labeled with σ, and for $s \in \{0,1\}^\ell$ denote by $v(s)$ the vertex reached when starting from v and walking according to s. (For vertices v in the last layer we will only care about p_v, which is either 0 or 1.)

Note that $p_v = \mathbb{E}_{s \in \{0,1\}^\ell}\left[p_{v(s)}\right]$, and hence (by Markov's inequality)

$$\Pr_s\left[p_{v(s)} \geq p_v \cdot (2nw/\epsilon)\right] \leq \epsilon/(2nw) .$$

By a union-bound over the $(n+1) \cdot w/\ell < 2nw$ vertices in the relevant layers, with probability more than $1-\epsilon$ over choice of $s \in \{0,1\}^\ell$, for every vertex in these layers we have that $p_{v(s)} < p_v \cdot (2nw/\epsilon)$. In this case, when starting from the initial vertex v_0 in the ROBP and walking according to the n-bit string $(s, s, s, ..., s)$ we pass through vertices $v_1, v_\ell, ...$ and reach a vertex v_n, and by induction for each $i \in [n/\ell]$ we have

$$p_{v_i} = p_{v_{i-1}(s)} < p_{v_{i-1}} \cdot (2nw/\epsilon) < ... < p_{v_1} \cdot (2nw/\epsilon)^i .$$

In particular, applying the above for $i = n$ and recalling that $p_{v_1} \leq B/2^n$, we have that $p_{v_n} < p_{v_1} \cdot (2nw/\epsilon)^{n/\ell} \leq B/2^n \cdot (2nw/\epsilon)^{n/\ell} < 1$, where the last inequality relied on our choice of ℓ. Hence, v_n is labeled with the more likely output $\neg\sigma$ of the ROBP.

The above proves that with probability at least $1-\epsilon$ over choice of seed s for the PRG, the ROBP evaluates to its more likely output. The pseudorandomness of this PRG follows because the probability over a uniform input that the ROBP evaluates to its more likely output is at least $1 - B/2^n \geq 1 - \epsilon$. ■

The proof above (as well as Saks' original proof) is elementary, and does not rely on the vast literature concerning derandomization of logspace (or of ROBP). Nevertheless, improving on the result that it yields is still an open problem:

Open Problem 9: Quantified derandomization of logspace with $B(T) = 2^{(1-o(1))\cdot T}$. *Strengthen Theorem D.1 to work with $B(T) = 2^{T-s(T)}$ for some sub-linear function s, or show that such an improvement implies that $\mathcal{BPL} = \mathcal{L}$.*

D.2 Quantified derandomization of Merlin-Arthur protocols

We are interested in derandomizing Merlin-Arthur protocols, and particularly in derandomizing $\mathcal{MA}$ and $\mathcal{AM}$ (see, e.g., [AB09, Section 8.2] for the standard definitions of these classes). Recall that assuming sufficiently strong lower bounds, both of these classes can be derandomized and equal $\mathcal{NP}$ (see [KM02]).

Goldreich and Wigderson asked if derandomizing $\mathcal{MA}$ or $\mathcal{AM}$ becomes easier when the verifier is extremely unlikely to err (i.e., to accept an incorrect proof or to reject a correct proof). They showed two complementary results, the first of which is the following quantified derandomization algorithm for a subclass of $\mathcal{MA}$.

Definition D.2 ($\mathcal{MA}$ with restricted verifiers)**.** *For a circuit class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$, we say that $L \subseteq \{0,1\}^*$ can be decided by an $\mathcal{MA}$ protocol with $\mathcal{C}$-verifiers if there exists an $\mathcal{MA}$ verifier V that decides L such that the following holds: For every input $x \in \{0,1\}^*$ and proof $w \in \{0,1\}^{\mathrm{poly}(|x|)}$, the decision of V at x with proof w as a function of the $m = \mathrm{poly}(n)$ random coins can be computed by a circuit in $\mathcal{C}_m$.*

Theorem D.3 (quantified derandomization of $\mathcal{MA}$ with $\mathcal{AC}^0$ verifiers; see [GW14, Theorem 7.3 in the Full Version])**.** *Assume that $L \subseteq \{0,1\}^*$ can be decided by an $\mathcal{MA}$ protocol with $\mathcal{AC}^0$ verifiers such that the verifier always errs on at most $B(T) = 2^{T^{1-\epsilon}}$ random choices. Then $L \in \mathcal{NP}$.*

Theorem D.3 may appear weak, because it only refers to $\mathcal{MA}$ verifiers whose decision as a function of the random coins is an $\mathcal{AC}^0$ circuit. However, if an analogous result holds for $\mathcal{AM}$ verifiers, then $\mathcal{AM} = \mathcal{NP}$! In fact, this conclusion holds even if the verifier's decision is only a CNF, and even for smaller values of $B(T) = 2^{T^\epsilon}$.

Definition D.4 ($\mathcal{AM}$ with restricted verifiers)**.** *For a circuit class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$, we say that $L \subseteq \{0,1\}^*$ can be decided by an $\mathcal{AM}$ protocol with $\mathcal{C}$-verifiers if there exists a deterministic procedure V and a polynomial $p\colon \mathbb{N} \to \mathbb{N}$ such that the following holds:*

- *For every $x \in L$ it holds that $\Pr_{r\in\{0,1\}^{p(n)}}[\exists w \in \{0,1\}^{p(n)}, V(x,w,r) = 1] \geq 2/3$.*
- *For every $x \notin L$ it holds that $\Pr_{r\in\{0,1\}^{p(n)}}[\forall w \in \{0,1\}^{p(n)}, V(x,w,r) = 0] \geq 2/3$.*
- *On n-bit inputs V can be computed by a circuit from $\mathcal{C}_{n+2p(n)}$.*

Theorem D.5 (threshold values for quantified derandomization of $\mathcal{AM}$ with CNF verifiers; see [GW14, Theorem 7.4 in the Full Version])**.** *Assume that for some $\epsilon \in (0,1)$ the following holds: For any $L \subseteq \{0,1\}^*$ that can be decided by an $\mathcal{AM}$ protocol with* CNF *verifiers such that the verifier always errs on at most $B(T) = 2^{T^\epsilon}$ random choices, we have that $L \in \mathcal{NP}$. Then $\mathcal{AM} = \mathcal{NP}$.*

To make sense of Theorems D.3 and D.5, recall that derandomization of $\mathcal{AM}$ is in general a harder problem than derandomization of $\mathcal{MA}$ (since $\mathcal{AM} \supseteq \mathcal{MA}$). Nevertheless, the contrast between the two results is still striking.

The proof of Theorem D.3 amounts to applying the quantified derandomization algorithm of Theorem 6.3 to the verifier's residual decision as a function of the random coins, when the input and the proof are fixed. Similar results can be obtained for analogous classes of $\mathcal{MA}$ with restricted verifiers (such as verifiers computable by formulas) using Theorems 6.4, 6.10, 6.12, and 6.15. However, this approach does not use the power of interaction for the quantified derandomization algorithm, but rather only applies a known quantified derandomization algorithm to the verifier's decision.

Open Problem 10: Quantified derandomization of $\mathcal{MA}$ using the power of interaction. *For any class $\mathcal{C}$, let $\mathcal{MA}^{\mathcal{C}}$ be the set of problems solvable by $\mathcal{MA}$ protocols in which the verifier's decision as a function of the random coins is computable in $\mathcal{C}$. Can we construct a quantified derandomization algorithm for $\mathcal{MA}^{\mathcal{C}}$ with better parameters than the known quantified derandomization algorithm for $\mathcal{C}$, using the interaction with the prover?*

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il