Electronic Colloquium on Computational Complexity, Report No. 169 (2019)

On Exponential-Time Hypotheses, Derandomization, and Circuit Lower Bounds

Lijie Chen[*] Ron D. Rothblum[†] Roei Tell[‡] Eylon Yogev[§]

November 21, 2019

Abstract

The Exponential-Time Hypothesis (ETH) is a strengthening of the $\mathcal{P} \neq \mathcal{NP}$ conjecture, stating that 3-SAT on n variables cannot be solved in time $2^{\epsilon \cdot n}$, for some $\epsilon > 0$. In recent years, analogous hypotheses that are "exponentially-strong" forms of other classical complexity conjectures (such as $\mathcal{NP} \not\subseteq \mathcal{BPP}$ or $co\text{-}\mathcal{NP} \not\subseteq \mathcal{NP}$) have also been considered. These Exponential-Time Hypotheses have been widely influential across different areas of complexity theory. However, their connections to *derandomization and circuit lower bounds* have yet to be systematically studied. Such study is indeed the focus of the current work, and we prove a sequence of results demonstrating that *the connections between exponential-time hypotheses, derandomization, and circuit lower bounds are remarkably strong*.

First, we show that if 3-SAT (or even TQBF) cannot be solved by probabilistic algorithms that run in time $2^{n/\text{polylog}(n)}$, then $\mathcal{BPP}$ can be deterministically simulated "on average case" in (nearly-)polynomial-time (i.e., in time $n^{\text{polyloglog}(n)}$). This result addresses a long-standing lacuna in uniform "hardness-to-randomness" results, which did not previously extend to such parameter settings. Moreover, we extend this result to support an "almost-always" derandomization conclusion from an "almost-always" lower bound hypothesis.

Secondly, we show that *disproving* certain exponential-time hypotheses requires proving breakthrough circuit lower bounds. In particular, if CircuitSAT for circuits over n bits of size $\text{poly}(n)$ can be solved by *probabilistic algorithms* in time $2^{n/\text{polylog}(n)}$, then $\mathcal{BPE}$ does not have circuits of quasilinear size. The main novel feature of this result is that we only assume the existence of a *randomized* circuit-analysis algorithm, whereas previous similar results crucially relied on the hypothesis that the circuit-analysis algorithm does not use randomness.

Thirdly, we show that a very weak exponential-time hypothesis is closely-related to the classical question of whether derandomization and circuit lower bounds are *equivalent*. Specifically, we show two-way implications between the hypothesis that the foregoing equivalence holds and the hypothesis that $\mathcal{E}$ cannot be decided by "small" circuits that are *uniformly generated* by relatively-efficient non-deterministic machines. This highlights a sufficient-and-necessary path for progress towards proving that derandomization and circuit lower bounds are indeed equivalent.

[*]Massachusetts Institute of Technology. Email: lijieche@mit.edu.
[†]Technion. Email: rothblum@cs.technion.ac.il.
[‡]Weizmann Institute of Science. Email: roei.tell@weizmann.ac.il.
[§]Simons Institute for the Theory of Computing. Email: eylony@gmail.com.

ISSN 1433-8092

Contents

1 Introduction **1**
1.1 Bird's eye view of our results 2
1.2 rETH and pseudorandom generators for uniform circuits 2
1.3 The negation of rETH and circuit lower bounds 5
1.4 NETH and an equivalence between derandomization and circuit lower bounds 7

2 Technical overview **9**
2.1 Near-optimal uniform hardness-to-randomness results for `TQBF` 10
2.2 Circuit lower bounds from randomized `CircuitSAT` algorithms 14
2.3 $\mathcal{NTIME}$-uniform circuits for $\mathcal{E}$ and the equivalence between derandomization and circuit lower bounds 16

3 Preliminaries **18**

4 rETH and near-optimal uniform hardness-to-randomness **22**
4.1 Construction of a well-structured function 22
4.2 PRGs for uniform circuits with almost-exponential stretch 35
4.3 Proofs of Theorems 1.1 and 1.2 45

5 NOT-rETH and circuit lower bounds from randomized algorithms **46**
5.1 Randomized `CircuitSAT` algorithms imply $\mathcal{BPE}$ circuit lower bounds . 48
5.2 Randomized Σ_2-`SAT`$[n]$ algorithms imply $\mathcal{BPE}$ circuit lower bounds . . 49

6 NETH and the equivalence of derandomization and circuit lower bounds **51**
6.1 A refined Karp-Lipton-style result 51
6.2 Proof of Theorems 1.4 and 1.5 59
6.3 Proof of Theorem 1.6 60

A Low-degree polynomials are sample-aided worst-case to average-case reducible **67**

B When even Merlin and Arthur encounter difficulties **68**

1 Introduction

The Exponential-Time Hypothesis (ETH), introduced by Impagliazzo and Paturi [IP01] (and refined in [IPZ01]), conjectures that 3-SAT with n variables and $m = O(n)$ clauses cannot be deterministically solved in time less than $2^{\epsilon \cdot n}$, for some constant $\epsilon = \epsilon_{m/n} > 0$. The ETH may be viewed as a "very strong version" of $\mathcal{P} \neq \mathcal{NP}$, since it conjectures that a specific problem in non-deterministic linear time (or quasi-linear time, depending on the machine model) requires essentially exponential time to solve.

Since the introduction of ETH many related variants, which are also "stronger versions" of classical complexity-theoretic conjectures, have also been introduced. For example, the Randomized Exponential-Time Hypothesis (rETH), introduced in [Del+14], conjectures that the same lower bound holds also for *probabilistic* algorithms (i.e., it is a strong version of $\mathcal{NP} \not\subseteq \mathcal{BPP}$). The Non-Deterministic Exponential-Time Hypothesis (NETH), introduced (implicitly) in [Car+16], conjectures that *co*-3SAT (with n variables and $O(n)$ clauses) cannot be solved by non-deterministic machines running in time $2^{\epsilon \cdot n}$ for some constant $\epsilon > 0$ (i.e., it is a strong version of *co*-$\mathcal{NP} \not\subseteq \mathcal{NP}$). The variations MAETH and AMETH are defined analogously (see [Wil16][1]), and other variations conjecture similar lower bounds for seemingly-harder problems (e.g., for #3-SAT; see [Del+14]). Each of these hypotheses also has a "strong" variant that conjectures a lower bound of $2^{(1-\epsilon) \cdot n}$, where $\epsilon > 0$ is arbitrarily small, for solving a corresponding problem (e.g., for solving SAT, *co*-SAT, or #SAT; for precise definitions see, e.g., [Wil18]). However, in this paper we focus only on the "non-strong" variants that conjecture lower bounds of $2^{\epsilon \cdot n}$ for some $\epsilon > 0$.

These Exponential-Time Hypotheses have been widely influential across different areas of complexity theory. Among the numerous fields to which they were applied so far are structural complexity (i.e., showing classes of problems that, conditioned on exponential-time hypotheses, are "exponentially-hard"), parameterized complexity, communication complexity, and fine-grained complexity; for relevant surveys see, e.g, [Woe03; LMS11; Wil15; Wil18]. However, the connections of exponential-time hypotheses to *derandomization and circuit lower bounds* have yet to be systematically studied.

In this work we indeed focus on the connections between exponential-time hypotheses, derandomization, and circuit lower bounds, and we prove a sequence of results demonstrating that the foregoing connections are remarkably strong. Loosely speaking, we show that proving (or disproving) certain relatively-weak exponential-time hypotheses requires proving the existence of strong derandomization algorithms, or breakthrough lower bounds for non-uniform circuit families, or strong connections (i.e., an equivalence) between the former and the latter. We will mainly focus on weak variants of rETH and of NETH, mentioned above, which refer to lower bounds for solving CircuitSAT, TQBF, or $\mathcal{E}$.

Our work follows a recent line-of-work that studies the implications of hypotheses

[1] In [Wil16], the introduction of these variants is credited to a private communication from Carmosino, Gao, Impagliazzo, Mihajlin, Paturi, and Schneider [Car+16].

in *fine-grained complexity* on derandomization and circuit lower bounds. For example, Abboud *et al.* [Abb+16] showed that very mild improvements in known polynomial-time algorithms for certain problems would yield circuit lower bounds; and Carmosino, Impagliazzo, and Sabin [CIS18] showed that fine-grained hardness hypotheses imply strong derandomization algorithms. The latter hypotheses are implied by the **Strong** Randomized Exponential-Time Hypothesis (rSETH); however, as already mentioned, in this work we will focus only on weaker ("non-strong") variants.

1.1 Bird's eye view of our results

Let us now give a brief overview of our main results, before describing them in more detail in Sections 1.2, 1.3 and 1.4.

First, in Section 1.2, we will show that the hypothesis rETH implies a strong *average-case derandomization* of $\mathcal{BPP}$. We will in fact consider a weaker hypothesis, which conjectures that the $\mathcal{PSPACE}$-complete problem Totally Quantified Boolean Formula (TQBF) cannot be solved in probabilistic time $2^{n/\mathrm{polylog}(n)}$.[2] Under this hypothesis, we show that $\mathcal{BPP}$ can be decided, in average-case and infinitely-often, by deterministic algorithms that run in time $n^{\mathrm{polyloglog}(n)}$ (see Theorem 1.1). This addresses a long-standing lacuna in *uniform "hardness-to-randomness"* results, which did not previously extend to such parameter settings. We also extend this result to deduce an "almost-always" derandomization of $\mathcal{BPP}$ from an "almost-always" hypothesized lower bound (see Theorem 1.2).

Secondly, in Section 1.3 we show that *disproving* a conjecture similar to rETH requires proving breakthrough circuit lower bounds. Specifically, we show that if there exists a *probabilistic algorithm* that solves CircuitSAT for circuits with n input bits and of size $\mathrm{poly}(n)$ in time $2^{n/\mathrm{polylog}(n)}$, then non-uniform circuits of quasilinear size cannot decide $\mathcal{BPE} \stackrel{\mathrm{def}}{=} \mathcal{BPTIME}[2^{O(n)}]$ (see Theorem 1.3). This result is analogous to well-known theorems stating that "non-trivial" circuit-analysis algorithms imply circuit lower bounds, but has the key novel feature that the hypothesized circuit-analysis algorithm is allowed to use *randomness*; see Section 1.3 for details.

Lastly, in Section 1.4 we show that a very weak version of NETH is closely-related to the question of whether (worst-case) derandomization of $\mathcal{BPP}$ and circuit lower bounds are fully equivalent. That is, we show two-way implications between a very weak version of NETH and the hypothesis that derandomization of $\mathcal{BPP}$ and circuit lower bounds are indeed equivalent (see Theorems 1.4, 1.5, and 1.6).

1.2 rETH and pseudorandom generators for uniform circuits

The first hypothesis that we study is rETH, which (slightly changing notation from above) asserts that probabilistic algorithms cannot decide if a given 3-SAT formula with v variables and $O(v)$ clauses is satisfiable in time less than $2^{\epsilon \cdot v}$, for some constant

[2] Recall that TQBF is the set of 3-SAT formulas φ over variables $w_1, ..., w_v$ such that $\forall w_1 \exists w_2 \forall w_3 ..., \varphi(w_1, ..., w_v) = 1$ (see Definition 4.6), and that 3-SAT reduces to TQBF in linear time.

$\epsilon > 0$. Note that such a formula can be represented with $n = O(v \cdot \log(v))$ bits, and therefore the conjectured lower bound as a function of the input length is $2^{\epsilon \cdot (n/\log(n))}$.

Recall that in a classical work, Impagliazzo and Wigderson [IW98] showed that if $\mathcal{BPP} \neq \mathcal{EXP}$ – in other words, if probabilistic algorithms not extremely strong – then there exists a pseudorandom generator with polynomial stretch that "fools" $\mathcal{BPP}$-uniform circuits. Specifically, there exists an algorithm G that stretches $n^{.01}$ bits to n bits such that every probabilistic algorithm that runs in time $n^{O(1)}$ and tries to output a circuit that distinguishes the output of G from uniform fails, with high probability. We call such an algorithm a pseudorandom generator (PRG) for uniform circuits (see Definition 3.8), and the existence of such a PRG implies what is typically referred to as an "average-case" (or "effective") derandomization of $\mathcal{BPP}$.[3] Since their PRG only has polynomial stretch, the original result of [IW98] only yields an average-case *sub-exponential time* derandomization of $\mathcal{BPP}$.

Now, since rETH conjectures a much stronger lower bound for probabilistic algorithms than $\mathcal{BPP} \neq \mathcal{EXP}$, one naturally expects it to yield a much stronger consequence: Namely, as the lower bound is near-exponential (i.e., $2^{\Omega(n/\log(n))}$), one would expect to obtain a PRG for uniform circuits with near-exponential stretch. The key problem, which is well-known, is that the proof framework of [IW98] does not scale well; in other words, the known *uniform* "hardness-to-randomness" tradeoffs are sub-optimal (see Section 2.1 for details). In particular, the current state-of-the-art tradeoff by Trevisan and Vadhan [TV07] only yields a PRG with *sub-exponential* stretch from such hardness. Previous works have bypassed this difficulty by constructing PRGs for uniform circuits from different hypotheses, which are arguably stronger (or, in some cases strictly stronger): For example, the hypothesis that $pr\mathcal{BPP} = pr\mathcal{P}$ (see [Gol11]), or hypotheses in fine-grained complexity (see [CIS18]).

Our first result directly addresses the problem described above, obtaining a near-optimal uniform hardness-to-randomness tradeoff for the "high-end" parameter setting. Specifically, we start from the hypothesis that the Totally Quantified Boolean Formula (`TQBF`) problem cannot be solved by probabilistic algorithms that run in time $2^{n/\text{polylog}(n)}$; this hypothesis is *weaker* than rETH (recall that 3-`SAT` can be reduced to `TQBF` with a linear overhead). Under this hypothesis, we show that there exists a PRG for uniform circuits with *almost-exponential* stretch (i.e., with seed length $\widetilde{O}(\log(n))$).

Theorem 1.1 (rETH $\Rightarrow$ PRG with almost-exponential stretch for uniform circuits; informal)**.** *Suppose that there exists $T(n) = 2^{n/\text{polylog}(n)}$ such that* `TQBF` $\notin \mathcal{BPTIME}[T]$. *Then, for every $t(n) = n^{\text{polyloglog}(n)}$, there exists a PRG that has seed length $\widetilde{O}(\log(n))$, runs in time $n^{\text{polyloglog}(n)}$, and is infinitely-often $(1/t)$-pseudorandom for every distribution over circuits that can be sampled in time t with $\log(t)$ bits of non-uniform advice.*[4]

[3]Specifically, for every $L \in \mathcal{BPP}$ there exists an efficient deterministic algorithm D that satisfies the following: Every probabilistic algorithm that gets input 1^n and tries to find an input $x \in \{0,1\}^n$ such that $D(x) \neq L(x)$ only has a small probability of success (see, e.g., [Gol11, Prop. 4.4]).

[4]Throughout the paper, when we say that the PRG is ϵ-pseudorandom for a distribution of circuits, we mean that the probability over choice of circuit that the circuit distinguishes the output of the PRG from uniform with advantage more than ϵ is at most ϵ (see Definitions 3.7 and 3.8).

The proof of Theorem 1.1 is based on a careful adaptation of the proof framework of [IW98], using new technical tools that we construct. The latter tools significantly refine and strengthen the technical tools that were used by [TV07] to obtain the previously-best uniform hardness-to-randomness tradeoff. For high-level overviews of the proof of Theorem 1.1 (and of the new constructions), see Section 2.1.

Overcoming the "infinitely-often" barrier. The hypothesis in Theorem 1.1 is that any probabilistic algorithm that runs in time $2^{n/\mathrm{polylog}(n)}$ fails to compute `TQBF` *infinitely-often*, and the corresponding conclusion is that the PRG "fools" uniform circuits only *infinitely-often*. This is identical to all previous uniform "hardness-to-randomness" results that used the [IW98] proof framework, which was not previously known to support an "almost-always" conclusion from an "almost-always" hypothesis.[5]

In the following result, we show how to partially overcome this challenge in our parameter setting. Specifically, we will assume that every probabilistic algorithm that runs in time $2^{n/\mathrm{polylog}(n)}$ fails to decide `TQBF` on almost all input lengths; that is, we assume that $\mathtt{TQBF} \notin \mathtt{i.o.}\mathcal{BPTIME}[2^{n/\mathrm{polylog}(n)}]$. From this hypothesis we will deduce two incomparable conclusions: First, that there exists a Hitting-Set Generator (HSG) for uniform circuits that works almost-always, and thus derandomizes $\mathcal{RP}$ in average-case almost-always;[6] and secondly, that $\mathcal{BPP}$ can also be derandomized in average-case almost-always, albeit for a weaker notion of "average-case" derandomization (see below) and using a tiny amount of non-uniform advice (i.e., $O(\mathrm{logloglog}(n))$ bits).

Theorem 1.2 (almost-always-rETH $\Rightarrow$ almost-always derandomization in time $n^{\mathrm{polyloglog}(n)}$; informal)**.** *Suppose that there exists $T(n) = 2^{n/\mathrm{polylog}(n)}$ such that $\mathtt{TQBF} \notin \mathtt{i.o.}\mathcal{BPTIME}[T]$. Then, for every $t(n) = n^{\mathrm{polyloglog}(n)}$,*

1. ("Black-box" almost-always derandomization of $\mathcal{RP}$:) *There exists a HSG for uniform circuits that has seed length $\widetilde{O}(\log(n))$, runs in time $n^{\mathrm{polyloglog}(n)}$, and for every distribution over circuits that can be sampled in time t with $\log(t)$ bits of non-uniform advice,* for almost all input lengths $n \in \mathbb{N}$, *the HSG "hits" circuits with n input bits from the distribution that accept at least $1/t(n)$ of their inputs.*

2. ("White-box" almost-always derandomization of $\mathcal{BPP}$ using short advice:) *For every $L \in \mathcal{BPTIME}[t]$ and every distribution ensemble $\mathcal{X} = \{\mathcal{X}_n \subset \{0,1\}^n\}$ such that $x \sim \mathcal{X}_n$ can be sampled in probabilistic time $t(n)$, there exists a deterministic algorithm $D = D_{\mathcal{X}}$ that runs in time $n^{\mathrm{polyloglog}(n)}$ and uses $O(\mathrm{logloglog}(n))$ bits of non-uniform advice such that for* almost all input lengths $n \in \mathbb{N}$ *it holds that $\Pr_{x\sim\mathcal{X}_n}[D(x) \neq L(x)] < 1/t(n)$.*

[5]In contrast, other proof strategies (which use different hypotheses) were able to support an "almost-always" conclusion, albeit not necessarily a PRG, from an "almost-always" hypothesis (see [GSTS03; CIS18]).

[6]A Hitting-set generator (HSG) for uniform circuits is defined analogously to PRG for uniform circuits, and implies an average-case derandomization of $\mathcal{RP}$ (see Definition 3.9 and Claim 3.10). We note that generic transformations of HSGs for *non-uniform* circuits to algorithms for *worst-case* derandomization of $\mathcal{BPP}$ are known (see, e.g., [ACR98; BF99; GVW11]), but these transformations do not seem to work for HSGs for uniform circuits and average-case derandomization of $\mathcal{BPP}$.

The notions of "average-case" derandomization in Items (1) and (2) of Theorem 1.2 are different. Specifically, the derandomization of $\mathcal{RP}$ in Item (1) implies that for every $L \in \mathcal{RP}$ there exists a single deterministic algorithm D such that for every efficiently-samplable distribution $\mathcal{X}$ it holds that D agrees with L with high probability over $\mathcal{X}$ (see Claim 3.10). The derandomization of $\mathcal{BPP}$ in Item (2), however, only implies that for every $L \in \mathcal{BPP}$ *and every efficiently-samplable distribution* $\mathcal{X}$ there exists a corresponding deterministic $D_{\mathcal{X}}$ that agrees with L with high probability over $\mathcal{X}$. For a high-level description of the proof of Theorem 1.2, see Section 2.1.

Non-deterministic extensions. Note that "scaled-up" versions of Theorems 1.1 and 1.2 for *non-deterministic settings* essentially follow from known results; that is, assuming lower bounds for non-deterministic uniform algorithms, we can deduce strong derandomization of corresponding non-deterministic classes. First, from the hypothesis MAETH [7] we can deduce strong circuit lower bounds, and hence also *worst-case* derandomization of $pr\mathcal{BPP}$ and of $pr\mathcal{MA}$ (this uses relatively standard Karp-Lipton-style arguments, following [Bab+93]; see Appendix B for details and for a related result). Similarly, as shown by Gutfreund, Shaltiel, and Ta-Shma [GSTS03], a suitable variant of AMETH implies an average-case derandomization of $\mathcal{AM}$.

1.3 The negation of rETH and circuit lower bounds

The results in Section 1.2 show that rETH implies strong average-case derandomization of $\mathcal{BPP}$ (indeed, this holds even for a weaker hypothesis that refers to TQBF). It is not clear, though, if rETH implies circuit lower bounds; this is since rETH only conjectures a lower bound for uniform probabilistic algorithms, rather than for non-uniform circuits. Loosely speaking, in this section, we show that a hypothesis that is a bit stronger than the *negation of* rETH implies breakthrough circuit lower bounds. Thus, intuitively (and being slightly inaccurate), rETH implies derandomization of $\mathcal{BPP}$ whereas (a form of) "not rETH" implies circuit lower bounds.

The main challenge in proving that "not rETH" implies circuit lower bounds is that our hypothesis only means that there exists a *randomized* "non-trivial" circuit-analysis algorithm. Recall that known arguments that deduce circuit lower bounds from "non-trivial" circuit-analysis algorithms, following the celebrated result of Williams [Wil13] (see, e.g., [SW13; BSV14; MW18; Che19; CW19]), crucially rely on the hypothesis that the circuit-analysis algorithm does not use randomness. It is a well-known challenge to obtain similar results for randomized algorithms; the best previous related result that we are aware of is by Oliveira and Santhanam [OS17], who deduced circuit lower bounds from randomized *subexponential-time* algorithms for *learning* with membership queries (see Section 2.2 for details). [8]

[7]Note that indeed a non-deterministic analogue of rETH is MAETH (or, arguably, AMETH), rather than NETH, due to the use of randomness. Also recall that, while the "strong" version of MAETH is false (see [Wil16]), there is currently no evidence against the "non-strong" version MAETH.

[8]Another known result, which was communicated to us by Igor Oliveira, asserts that if CircuitSAT

We are able to prove a significantly more refined result, with parameters that are closer to the ones in the known results for deterministic algorithms. Specifically, the first hypothesis that we consider is that there exists a probabilistic algorithm that solves `CircuitSAT` for circuits with n input bits and of size $\mathrm{poly}(n)$ in time $2^{n/\mathrm{polylog}(n)}$, for a sufficiently large polylogarithmic function. Under this hypothesis, we show that $\mathcal{BPE} \overset{\mathrm{def}}{=} \mathcal{BPTIME}[2^{O(n)}]$ cannot be decided by non-uniform circuits of quasilinear size; indeed, these lower bounds might seem weak (compared, say, to lower bounds for $\mathcal{P}/\mathrm{poly}$), but they would nevertheless be a major breakthrough in circuit complexity.[9] That is:

Theorem 1.3 (circuit lower bounds from non-trivial randomized `CircuitSAT` algorithms)**.** *For any constant $c \in \mathbb{N}$ there exists a constant $c' \in \mathbb{N}$ such that the following holds. If* `CircuitSAT` *for circuits over n variables and of size $n^2 \cdot (\log n)^{c'}$ can be solved in probabilistic time $2^{n/(\log n)^{c'}}$, then $\mathcal{BPE} \not\subset \mathcal{SIZE}[n \cdot (\log n)^c]$.*

Note that Theorem 1.3 is not a strict strengthening of the results of [Wil13; MW18], despite the fact that it only assumes the existence of a randomized algorithm rather than a deterministic one. This is since the results of [Wil13; MW18] yield some lower bound even if the running time of the algorithm is $2^n/n^{\omega(1)}$ (e.g., the bound $\mathcal{NEXP} \not\subseteq \mathcal{P}/\mathrm{poly}$); and from an algorithm that runs in time $2^{n/\mathrm{polylog}(n)}$ as in Theorem 1.3, their results yield the lower bound $\mathcal{NP} \not\subset \mathcal{SIZE}[n^k]$ for every fixed $k \in \mathbb{N}$. In comparison, in Theorem 1.3 we can only deduce a lower bound if the running time is $2^{n/\mathrm{polylog}(n)}$, and the deduced lower bound is incomparable to $\mathcal{NP} \not\subseteq \mathcal{SIZE}[n^k]$ (since $\mathcal{BPE}$ and $\mathcal{NP}$ are incomparable). Nevertheless, as far as we are aware, Theorem 1.3 is the first result that deduces circuit lower bounds from near-exponential-time *randomized* algorithms for a natural circuit-analysis task.

The proof of Theorem 1.3 uses techniques that are very different from the ones used in previous results (e.g., from "easy-witness" techniques or from $\mathcal{MA}$ lower bounds; see Section 2.2). Loosely speaking, the proof is based on the connection shown in [OS17] (following [FK09; HH13; KKO13]) between learning algorithms and circuit lower bounds, while crucially leveraging the technical tools that we develop as part of the proof of Theorems 1.1 and 1.2. See Section 2.2 for a description of the proof.

In addition, we show that a hypothesis similar to rETH implies an average-case derandomization of $\mathcal{BPP}$, whereas its negation implies circuit lower bounds. We deduce that, unconditionally, at least one of the following statements is true: (1) $\mathcal{BPP}$ can be derandomized, in average-case and infinitely-often, in time $n^{\mathrm{polyloglog}(n)}$; (2) $\mathcal{BPE}$ is

for circuits over n variables and of size $\mathrm{poly}(n)$ can be solved in probabilistic sub-exponential time $2^{n^{o(1)}}$, then $BPTIME[2^{O(n)}] \not\subset \mathcal{P}/\mathrm{poly}$. This result can be seen as a "high-end" form of our result (i.e., of Theorem 1.3), where the latter will use a weaker hypothesis but deduce a weaker conclusion.

[9]For context, let us recall the current state-of-the-art lower bounds against circuits of quasilinear size. There exist "hard" functions for such circuits that are computable by *exponential-time Merlin-Arthur* verifiers (see [BFT98]), or computable by brute-force diagonalization (which requires time exponential in the circuit-size, or several non-deterministic alterations), or computable by polynomial-time Merlin-Arthur verifiers that use *one bit of non-uniform advice* (see [San09; MW18]).

hard for circuits of quasilinear size. (See Corollary 5.6 for a precise statement.) We comment that we do not know how to prove this result by applying a "win-win" argument to rETH itself, because we do not know if the statement "rETH is false" implies circuit lower bounds (this is since the hypothesis of Theorem 1.3 is *stronger* than the negation of rETH).[10] Instead, we base our "win-win" argument on other hypotheses, which are similar to rETH (i.e., on hypotheses that refer either to Σ_2-$\mathtt{SAT}[O(n)]$ or to $\mathtt{TQBF}$; see Section 5.2 for details).

1.4 NETH and an equivalence between derandomization and circuit lower bounds

In this section we consider a different notion of derandomization of $\mathcal{BPP}$, namely *worst-case* derandomization rather than average-case derandomization. For example, we will be interested in worst-case derandomizations of $pr\mathcal{BPP}$ such as $pr\mathcal{BPP} = pr\mathcal{P}$ or $pr\mathcal{BPP} \subseteq pr\mathcal{NP}$. Recall that the Non-Deterministic Exponential-Time Hypothesis (NETH) conjectures that co-$3\mathtt{SAT}$ (with n variables and $O(n)$ clauses) cannot be solved by non-deterministic machines running in time $2^{\epsilon \cdot n}$ for some $\epsilon > 0$. The motivating observation for our results in this section is that NETH has an unexpected consequence to the long-standing question of whether *worst-case derandomization of $pr\mathcal{BPP}$ is equivalent to circuit lower bounds against $\mathcal{E}$*.

Specifically, recall that if $\mathcal{E}$ does not have "small" non-uniform circuits, then there exist efficient pseudorandom generators, which allow for worst-case derandomization of $pr\mathcal{BPP}$ (this was proved in the sequence of works that began with [NW94; IW99] and culminated in [Uma03]). On the other hand, efficient deterministic algorithms for $pr\mathcal{BPP}$ imply circuit lower bounds, albeit only against non-deterministic classes rather than against $\mathcal{E}$ (see [MW18] and the follow-up [Tel19], following [Wil13] and the sequence of works that began with [BFT98; IKW02]). It is a long-standing question of whether the foregoing results can be strengthened to show a *full equivalence* between worst-case derandomization of $pr\mathcal{BPP}$ and circuit lower bounds against $\mathcal{E}$; one well-known implication of such an equivalence would be that any derandomization of *$pr\mathcal{BPP}$ necessitates* the construction of PRGs that "fool" non-uniform circuits.[11]

Our main contribution is in showing that, loosely speaking (and being slightly inaccurate), proving a *very weak form of* NETH is both sufficient and necessary in order to answer the question of equivalence in the affirmative. Towards presenting this very weak form of NETH, let us say that $L \subseteq \{0,1\}^*$ has $\mathcal{NTIME}[T]$-uniform circuits if there exists a non-deterministic machine M that gets input 1^n, runs in time $T(n)$, and

[10]The hypothesis in Theorem 1.3 is stronger both because it refers to running-time $2^{n/\mathrm{polylog}(n)}$ (rather than $2^{\epsilon \cdot n}$), and because it requires deciding satisfiability of n-bit circuits of size $\mathrm{poly}(n)$, whereas rETH refers to deciding satisfiability of an n-bit formula of size $O(n)$.

[11]The question of equivalence is mostly "folklore", but was mentioned several times in writing. It was raised as a hypothetical possibility in [TV07] (as a "super-Karp-Lipton theorem"), and was (implicitly) referred to again in [Gol11], which focuses on average-case derandomization and whose main goal was to *bypass* this question and obtain pseudorandom generators without circuit lower bounds. The question was recently raised explicitly (as a conjecture) in [Tel19], following the results of [MW18].

satisfies the following: For some non-deterministic choices M outputs a *single circuit* $C : \{0,1\}^n \to \{0,1\}$ that *decides* L *on all inputs* $x \in \{0,1\}^n$, and whenever M does not output such a circuit, it outputs $\perp$. Moreover, we will sometimes quantify the *size* of the circuit that the machine outputs, in cases where the machine runs in time $T(n)$ but outputs a circuit of much smaller size $S(n) \ll T(n)$.

Indeed, the assumption that L has $\mathcal{NTIME}[T]$-uniform circuits (let alone of size $S < T$) is much stronger than the assumption $L \subseteq \mathcal{NTIME}[T]$, since the former means that we can uniformly generate in time $T(n)$ (using non-determinism) a *concise description of* $L \cap \{0,1\}^n$, in the form of a single circuit that decides L on all n-bit inputs. (In particular, it means that $L \in \mathcal{NTIME}[T] \cap \mathcal{SIZE}[S]$.) Thus, the assumption that $\mathcal{E}$ does *not* have $\mathcal{NTIME}[T]$-uniform circuits of size S, for a small T and perhaps smaller S, is *much weaker* than NETH (which asserts that *co*-3SAT $\notin \mathcal{NTIME}[T]$ for $T(n) = 2^{\epsilon \cdot (n/\log(n))}$). The fact that such a weak assumption (i.e., for small values of T and S that will be specified below) suffices to deduce that derandomization and circuit lower bounds are equivalent can be seen as appealing evidence that the equivalence indeed holds.

Our first result asserts that if $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of arbitrary polynomial size (for some $\delta > 0$), then derandomization of $pr\mathcal{BPP}$ in sub-exponential time is equivalent to lower bounds for polynomial-sized circuits against $\mathcal{E}$. Moreover, we show that under this hypothesis, even *non-deterministic derandomization* of $pr\mathcal{BPP}$ (in sub-exponential time) is equivalent to such lower bounds.

Theorem 1.4 (NETH $\Rightarrow$ circuit lower bounds are equivalent to derandomization; "low-end" setting)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of arbitrary* polynomial size*. Then, the following statements are equivalent:*

1. *$pr\mathcal{BPP} \subseteq$ i.o.$pr\mathcal{NSUBEXP}$.*
2. *$pr\mathcal{BPP} \subseteq$ i.o.$pr\mathcal{SUBEXP}$.*
3. *$\mathcal{DTIME}[2^n] \not\subseteq \mathcal{P}/\text{poly}$.*

where $pr\mathcal{NSUBEXP} = \cap_{\epsilon>0}$i.o.$pr\mathcal{NTIME}[2^{n^\epsilon}]$ and $pr\mathcal{SUBEXP} = \cap_{\epsilon>0}$i.o.$pr\mathcal{DTIME}[2^{n^\epsilon}]$.

Recall that the circuit lower bound in Item (3) of Theorem 1.4 *unconditionally* implies the derandomization results in Items (1) and (2) (see [Bab+93; Uma03]). The point of Theorem 1.4 is that, under our hypothesis, the derandomization results also imply the corresponding circuit lower bounds. We also note that the conditional equivalence between *non-deterministic* derandomization and circuit lower bounds counters a common intuition for why the known "derandomization implies lower bounds" results fail to show an equivalence as in the conclusion of Theorem 1.4; see the comment in the end of Section 2.3 for details.

Next, we present a version of Theorem 1.4 for "high-end" parameter settings (i.e., an equivalence between fast derandomization and lower bounds for large circuits). For this "high-end" result we will need the stronger hypothesis that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\Omega(n)}]$-uniform circuits, and we prove the weaker conclusion that circuit lower bounds are only equivalent to simulation of $pr\mathcal{BPP}$ in small *deterministic* (rather than non-deterministic) time. On the other hand, the "high-end" result also extends to an "almost-always" version (i.e., if the hypothesis is an "almost-always" lower bound, then we can deduce an "almost-always" derandomization; for simplicity, the statement refers only to the "almost-always" version). Specifically:

Theorem 1.5 (NETH $\Rightarrow$ circuit lower bounds are equivalent to derandomization; "high--end" setting)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\delta \cdot n}]$-uniform circuits even infinitely-often. Then:*

$$pr\mathcal{BPP} = pr\mathcal{P} \iff \exists \epsilon > 0 : \mathcal{DTIME}[2^n] \not\subset \mathtt{i.o.}\mathcal{SIZE}[2^{\epsilon \cdot n}]\ ,$$

and furthermore, for every constant $c > 1$ we have that

$$pr\mathcal{BPP} \subseteq pr\mathcal{DTIME}\left[2^{O(\log n)^c}\right] \iff \exists \epsilon > 0 : \mathcal{DTIME}[2^n] \not\subset \mathtt{i.o.}\mathcal{SIZE}[\epsilon \cdot 2^{n^{1/c}}]\ .$$

Remarkably, as mentioned above, a hypothesis similar to the ones in Theorems 1.4 and 1.5 is also *necessary* in order to prove that derandomization and circuit lower bounds are equivalent. Specifically, we show that if the latter equivalence is true, then $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[T]$-uniform circuits for a function $T(n) = 2^{2^{\mathrm{polyloglog}(n)}} = 2^{\mathrm{quasipolylog}(n)}$ that is "in between" quasipolynomial and subexponential:

Theorem 1.6 (a partial converse to Theorems 1.4 and 1.5; informal)**.** *There exist a class $\mathcal{T}$ of time functions and a class $\mathcal{S}$ of size functions that are both "in between" quasipolynomial and subexponential such that the following holds. If*

$$pr\mathcal{BPP} \subset \mathtt{i.o.} \cup_{T \in \mathcal{T}} pr\mathcal{NTIME}[T] \iff \mathcal{DTIME}[2^n] \not\in \cup_{S \in \mathcal{S}} \mathcal{SIZE}[S]\ ,$$

then, for every $c \in \mathbb{N}$, it holds that $\mathcal{DTIME}[2^n]$ does not have $\mathcal{NTIME}[T]$-uniform circuits, where $T(n) = 2^{2^{(\log\log n)^c}}$.

The hypothesis of Theorem 1.6 is implied by the assumption that $\mathcal{E}$ does not have $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits for some $\delta > 0$. By a more careful optimization of the underlying argument we also show a tighter connection: That is, assuming that $\mathcal{E}$ does not have $\mathcal{NTIME}[T]$-uniform circuits, we deduce an equivalence between derandomization and circuit lower bounds, which in turn implies that that $\mathcal{E}$ does not have $\mathcal{NTIME}[T']$-uniform circuits, for functions $T' < T$ that are closer (and both T and T' are "in between" quasipolynomial and subexponential; specific statements appear in Section 6.3).

2 Technical overview

In this section we provide high level overviews for the proofs of our main results, which were stated in Section 1. In Section 2.1 we describe the proofs of Theorems 1.1 and 1.2. In Section 2.2 we describe the proof of Theorem 1.3, which relies on the proofs from Section 2.1. And in Section 2.3 we describe the proofs of Theorems 1.5 and 1.6.

2.1 Near-optimal uniform hardness-to-randomness results for TQBF

Let us begin by describing the proof of Theorem 1.1, which will serve as a basis for the proof of Theorem 1.2. In high-level, our proof strategy follows the classic approach of Impagliazzo and Wigderson [IW98]. The starting-point of this approach is a well-structured function $f^{\mathtt{ws}}\colon \{0,1\}^* \to \{0,1\}^*$; the meaning of the term "well-structured" differs across different follow-up works to [IW98], and in the current work it will also take on a new meaning, but for now let us intuitively think of $f^{\mathtt{ws}}$ as downward self-reducible and as having properties akin to random self-reducibility. Instantiating the Nisan-Wigderson pseudorandom generator with a suitable encoding $\mathtt{ECC}(f^{\mathtt{ws}})$ of $f^{\mathtt{ws}}$ as the “hard” function (again, the precise requirements from ECC differ across works), the proof framework of [IW98] aims to show that if the PRG with stretch $t(n)$ does not “fool” uniform distinguishers even infinitely-often, then $f^{\mathtt{ws}}$ is computable in probabilistic time $t'(n) > t(n)$. As a contrapositive, if probabilistic algorithms cannot compute $f^{\mathtt{ws}}$ in time t', then the PRG “fools” uniform distinguishers infinitely-often.

The key challenge underlying this proof approach is the *significant overheads* in the proof, which increase the time complexity t' of computing $f^{\mathtt{ws}}$. In the original proof of [IW98] this time was roughly $t'(n) \approx t(t(n))$, and the state-of-the-art prior to the current work, by Trevisan and Vadhan [TV07] (following [CNS99]), yielded time $t'(n) = \mathrm{poly}(t(\mathrm{poly}(n)))$. Since the relevant functions $f^{\mathtt{ws}}$ in all works are computable in $\mathcal{E} = \mathcal{DTIME}[2^{O(n)}]$, proofs with such overhead can yield at most a sub-exponential stretch $t(n) = 2^{n^{\Omega(1)}}$. As mentioned in Section 1.2, Goldreich [Gol11] bypassed this difficulty by using the stronger hypothesis $pr\mathcal{BPP} = pr\mathcal{P}$, whereas Carmosino, Impagliazzo, and Sabin [CIS18] bypassed this difficulty by using hypotheses from fine-grained complexity (that are implied by the “strong” version of rETH, i.e. by rSETH). In contrast, we take a brute-force approach: We replace *all* of the polynomial overheads in the input length in the [IW98; TV07] proof with only *polylogarithmic overheads* in the input length. That is, we will show that for carefully-constructed $f^{\mathtt{ws}}$ and suitably-chosen ECC (and with some variations in the proof itself), if the PRG instantiated with $\mathtt{ECC}(f^{\mathtt{ws}})$ for stretch t does not “fool” uniform distinguishers infinitely-often, then $f^{\mathtt{ws}}$ can be computed in time $t'(n) = t(\widetilde{O}(n))^{O(1)}$.

2.1.1 The well-structured function $f^{\mathtt{ws}}$

Following an idea of Trevisan and Vadhan [TV07], we take $f^{\mathtt{ws}}$ to be an artificial problem that we carefully construct. Our function $f^{\mathtt{ws}}$ will satisfy the following requirements. First, we require that $f^{\mathtt{ws}}$ will be computable in linear space. Secondly, we require that TQBF can be reduced to $f^{\mathtt{ws}}$ in *quasilinear time*. Thirdly, we require that $f^{\mathtt{ws}}$ will be downward self-reducible, yet only in time $2^{n/\mathrm{polylog}(n)}$.[12] And lastly, we require that $f^{\mathtt{ws}}$ will be sample-aided worst-case to δ-average-case reducible, for $\delta(n) = 2^{-n/\mathrm{polylog}(n)}$; this property, which is implicit in many works and was recently

[12]That is, there exists an algorithm that computes $f^{\mathtt{ws}}_n$ (i.e., $f^{\mathtt{ws}}$ on inputs of length n) in time $2^{n/\mathrm{polylog}(n)}$ when given oracle access to $f^{\mathtt{ws}}_{n-1}$.

made explicit by Goldreich and G. Rothblum [GR17], asserts the following: There exists a uniform algorithm T that gets as input a circuit $C : \{0,1\}^n \to \{0,1\}^*$ that agrees with $f_n^{\mathtt{ws}}$ on at least $\delta(n)$ of the inputs, and *labeled examples* $(x, f^{\mathtt{ws}}(x))$ where $x \in \{0,1\}^n$ is uniformly-chosen, runs in time $2^{n/\mathrm{poly}\log(n)}$ and with high probability outputs a circuit $C' : \{0,1\}^n \to \{0,1\}^*$ that computes $f_n^{\mathtt{ws}}$ on all inputs (see Definition 4.2 for intuition and details).

The construction of $f^{\mathtt{ws}}$ is the main technical part of the proof of Theorem 1.1; the main challenge in constructing $f^{\mathtt{ws}}$ is not to obtain each of the required properties separately, but to obtain all of them *simultaneously*. We first explain (very briefly) the key points underlying the construction; a detailed overview is presented in Section 4.1. We will later show how $f^{\mathtt{ws}}$ can be used to prove Theorem 1.1.

Our construction of $f^{\mathtt{ws}}$ is a significant refinement of the construction of the well-known $\mathcal{PSPACE}$-complete set constructed by Trevisan and Vadhan [TV07]. Our starting point is their previous construction. Loosely speaking, they first reduce TQBF to an intermediate problem TQBF$'$ that is amenable to arithmetization, and then reduce TQBF$'$ to an arithmetic problem. Now, for every $n \in \mathbb{N}$ they construct a collection of $\mathrm{poly}(n)$ low-degree polynomials that are randomly self-reducible (since they have low degree) and also satisfy a property akin to downward self-reducibility; loosely speaking, these polynomials arise from applying the protocol underlying the proof of $\mathcal{IP} = \mathcal{PSPACE}$ to TQBF$'$ [Lun+92; Sha92]. Finally they "combine" these polynomials to a Boolean function $f^{\mathtt{ws}}$ that "inherits" their useful properties, and is thus well-structured; this is done by mapping each $n \in \mathbb{N}$ to an interval of $\mathrm{poly}(n)$ input lengths corresponding to the collection of polynomials.

The main problem for us is that *each* of the four foregoing steps entails a *polynomial overhead* in the input length; again, our challenge is to eliminate all of of the overheads *simultaneously*. Loosely speaking, we do so by first reducing TQBF to another problem, denoted $\mathtt{TQBF}^{\mathtt{loc}}$, which is both amenable to arithmetization and reducible from TQBF in quasilinear time (see Claim 4.7.1). We then move to an arithmetic setting that will later on support the strong random self-reducibility that we want (namely, low-degree polynomials in this setting will be sample-aided worst-case to δ-average-case reducible), and show how to arithmetize $\mathtt{TQBF}^{\mathtt{loc}}$ in this setting (see Claim 4.7.2). Then, we construct a *much smaller* collection of only $\mathrm{polylog}(n)$ low-degree polynomials (instead of $\mathrm{poly}(n)$ polynomials), roughly corresponding to a version of $\mathcal{IP} = \mathcal{PSPACE}$ with $\mathrm{polylog}(n)$ rounds and relatively-high verification times (see Claim 4.7.3); these polynomials are sample-aided worst-case to δ-average-case reducible (see Proposition A.1), and also posses a property similar to downward self-reducibility. Finally, we "combine" the polynomials (for every $n \in \mathbb{N}$) to obtain a single function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$, where the small number of polynomials allows us to reduce $\mathtt{TQBF}^{\mathtt{loc}}$ to $f^{\mathtt{ws}}$ in only quasilinear time. Indeed, we are "paying" for these optimizations, by the fact that all of the underlying algorithms (e.g., for downward self-reducibility and for sample-aided worst-case to δ-average-case reducibility) now run in time $2^{n/\mathrm{polylog}(n)}$, instead of in polynomial time; but we are indeed able to afford this in our proof (since eventually we only need to solve TQBF in time $2^{n/\mathrm{polylog}(n)}$).

2.1.2 Instantiating the [IW98] proof framework with the function $f^{\texttt{ws}}$

Given this construction, we can now instantiate a relatively-standard variant of the [IW98] proof framework with $f^{\texttt{ws}}$, as follows (for simplicity, we show how to "fool" polynomial-time distinguishers that do not use advice). Let $\texttt{ECC}$ be the Goldreich-Levin [GL89] (i.e., Hadamard) encoding $\texttt{ECC}(f^{\texttt{ws}})(x, r) = \oplus_i f^{\texttt{ws}}(x)_i \cdot r_i$. The argument of [IW98] (following [NW94]) shows that if for input length n there exists a uniform $\text{poly}(n)$-time distinguisher A for the Nisan-Wigderson PRG (instantiated with $\texttt{ECC}(f^{\texttt{ws}})$) that succeeds with advantage $1/n$, then for input length $\ell = \widetilde{O}(\log(n))$ (corresponding to the set-size in the underlying combinatorial design) there is a *weak learner* for $\texttt{ECC}(f^{\texttt{ws}})$: That is, there exists an algorithm that gets oracle access to $\texttt{ECC}(f^{\texttt{ws}})$, runs in time $\text{poly}(n) \approx 2^{\ell/\text{polylog}(\ell)}$, and outputs a small circuit that agrees with $\texttt{ECC}(f^{\texttt{ws}})$ on approximately $1/2 + 1/n^2 \approx 1/2 + \delta_0(\ell)$ of the ℓ-bit inputs, where $\delta_0(\ell) = 2^{-\ell/\text{polylog}(\ell)}$.

Assuming that there exists a distinguisher for the PRG as above for every $n \in \mathbb{N}$, we deduce that a weak learner exists for every $\ell \in \mathbb{N}$. Following [IW98], for each input length $i = 1, ..., \ell$ we construct a circuit of size $2^{i/\text{polylog}(i)}$ for $f^{\texttt{ws}}_i$. Specifically, in iteration i we run the learner for $\texttt{ECC}(f^{\texttt{ws}})$ on input length $2i$, and answer its oracle queries using the downward self-reducibility of $f^{\texttt{ws}}$, the circuit that we have for $f^{\texttt{ws}}_{i-1}$, and the fact that $\texttt{ECC}(f^{\texttt{ws}})_{2i}$ is easily computable given access to $f^{\texttt{ws}}_i$. The learner outputs a circuit of size $2^{2i/\text{polylog}(2i)}$ that agrees with $\texttt{ECC}(f^{\texttt{ws}})$ on approximately $1/2 + \delta_0(2i)$ of the $2i$-bit inputs, and the argument of [GL89] allows to efficiently transform this circuit to a circuit of similar size that computes $f^{\texttt{ws}}$ on a approximately $\delta(i) = \text{poly}(\delta_0(2i))$ of the i-bit inputs. Our goal now is to transform this circuit to a circuit of similar size that computes $f^{\texttt{ws}}$ on all i-bit inputs. Recall that in general, performing such transformations by a *uniform* algorithm is challenging (intuitively, if $f^{\texttt{ws}}$ is a codeword in an error-correcting code, this corresponds to uniform list-decoding of a "very corrupt" version of $f^{\texttt{ws}}$). However, in our specific setting we can produce random *labeled samples* for $f^{\texttt{ws}}$, using its downward self-reducibility and the circuit that we have for $f^{\texttt{ws}}_{i-1}$. Relying on the *sample-aided worst-case to average-case reducibility* of $f^{\texttt{ws}}$, we can transform our circuit to a circuit of similar size that computes $f^{\texttt{ws}}_i$ on all inputs.

Finally, since $\texttt{TQBF}$ is reducible with quasilinear overhead to $f^{\texttt{ws}}$, if we can compute $f^{\texttt{ws}}$ in time $2^{n/\text{polylog}(n)}$ then we can compute $\texttt{TQBF}$ in such time. Moreover, since $f^{\texttt{ws}}$ is computable in space $O(\ell) = \tilde{O}(\log(n))$ (and thus in time $n^{\text{polyloglog}(n)}$), the pseudorandom generator is computable in time $n^{\text{polyloglog}(n)}$.

2.1.3 The "almost-always" version: Proof of Theorem 1.2

The proof framework of [IW98], which underlies Theorem 1.1, was previously only known to yield PRGs that "fool" any uniform distinguisher *infinitely-often*, rather than an "almost-always" derandomization. We now explain how we can adapt the proof above framework in order to get an "almost-always" conclusion, in our specific setting of PRGs with near-exponential stretch.

For starters, we will need a stronger property of $f^{\texttt{ws}}$. Specifically, we construct a function $f^{\texttt{ws}}$ that is downward self-reducible in a polylogarithmic number of steps, which means that for every input length ℓ there exists an input length $\ell_0 \geq \ell - \text{polylog}(\ell)$ such that $f^{\mathtt{ws}}$ is efficiently-computable at input length ℓ_0 (i.e., $f^{\mathtt{ws}}_{\ell_0}$ is computable in time $2^{\ell_0/\text{polylog}(\ell_0)}$ without a "downward" oracle; see Section 4.1.1 for intuition and details).

Now, observe that the transformation of a probabilistic time-t algorithm A (that distinguishes G from uniform) to a probabilistic time-t' algorithm F (that computes $f^{\mathtt{ws}}$) actually gives a "point-wise" guarantee: For every input length $n \in \mathbb{N}$, if A distinguishes the PRG on a corresponding set of input lengths S_n, then F computes $f^{\mathtt{ws}}$ correctly at input length $\ell = \ell(n) = \tilde{O}(\log(n))$; specifically, S_n is the set of input lengths at which we need a distinguisher for G, in order to obtain a weak learner for $\mathtt{ECC}(f^{\mathtt{ws}})$ at smaller input lengths, and use the downward self-reducibility argument for $f^{\mathtt{ws}}$ at input lengths $\ell, \ell-1, ..., \ell_0$. Moreover, since $f^{\mathtt{ws}}$ is downward self-reducible in polylog steps, we will only need weak learners at inputs $\ell, \ell-1, ..., \ell_0 = \ell - \text{polylog}(\ell)$; hence, we can show that S_n is a set of $\text{polylog}(\ell) = \text{polyloglog}(n)$ input lengths in the interval $[n, n^2]$ (see Lemma 4.9 for the precise calculation). Taking the contrapositive, if $f^{\mathtt{ws}}$ cannot be computed by F on almost all ℓ's, then for every $n \in \mathbb{N}$ there exists an input length $m \in S_n \subset [n, n^2]$ such that G fools A at input length m.[13]

It is now relatively straightforward to obtain an "almost-always" HSG. Given input 1^n, the HSG H randomly chooses $m \in S_n$ and emulates $G(1^m)$ using its random seed, truncating the output of $G(1^m)$ to $n \leq m$ bits. To see why this works, assume towards a contradiction that a uniform distinguisher $A(1^n)$ outputs, with probability at least $1/n$, a circuit with acceptance probability at least $1/n$ that rejects all the outputs of $H(1^n)$. Note that for every choice of $m \in S_n$ by H, the output distribution of H conditioned on this choice of m is simply the output distribution of $G(1^m)$, truncated to n bits. Then, we can construct a distinguisher A' for G that for every $m \in S_n$ emulates $A(1^n)$, and thus outputs with probability at least $1/n \geq 1/m$ a circuit with acceptance probability at least $1/n \geq 1/m$ that rejects all the outputs of G; this contradicts our assumption that for almost all $n \in \mathbb{N}$ it holds that G fools the distinguisher A' on *some* input length $m \in S_n$.[14] The seed length of H is $\log(|S_n|) + \tilde{O}(\log(m)) = \tilde{O}(\log(n))$.

The second item of Theorem 1.2 deduces an "almost-always" derandomization of $\mathcal{BPP}$ with $O(\text{logloglog}(n))$ bits of advice. In contrast to the HSG construction above, given input 1^n we cannot now simply choose a random $m \in S_n$ and hope to "hit" a good m (as that would critically deteriorate the "two-sided error" pseudorandom-

[13]Actually, since $f^{\mathtt{ws}}$ is downward self-reducible in polylog steps, it can be computed relatively-efficiently on infinitely-many input lengths, and thus cannot be "hard" for almost all ℓ's. However, since $\mathtt{TQBF}$ can be reduced to $f^{\mathtt{ws}}$ with quasilinear overhead, if $\mathtt{TQBF}$ is "hard" almost-always then for every $\ell(n)$ there exists $\ell' \leq \tilde{O}(\ell(n))$ such that $f^{\mathtt{ws}}$ is "hard" on ℓ', which allows our argument to follow through, with a similar set $\overline{S_n} \subset [n, n^{\text{polyloglog}(n)}]$ (i.e., with only a minor loss in parameters; see Proposition 4.11 for details). For simplicity, we ignore this issue in the overview.

[14]There is a slight complication here, since the distinguisher A' needs to emulate $A(1^n)$ for the "good" input length n, and it is not clear how A' can find the "good" input length n. One option is for A' to simply randomly guess $n < m$; another option, which we use in our proof, is to show that G also "fools" distinguishers that use a small amount of non-uniform advice (the proof uses an idea of Oliveira and Santhanam [OS17, Sec. 6]), in which case A' can get the "good" input length n as advice. See Proposition 4.11 for details.

ness guarantee). Thus, our derandomization algorithm gets input 1^n and also gets the "good" input length $m \in S_n$ *as non-uniform advice*; it then simulates $G(1^m)$ and truncates the output to n bits. (Similarly to the HSG argument above, we can show that truncating the output of our PRG preserves its pseudorandomness in this uniform setting; see Proposition 4.12 for details.) Indeed, since $|S_n| = \text{polyloglog}(n)$, the advice length is $O(\text{logloglog}(n))$. Note, however, that for every potential distinguisher A there exists a *different* input length $m \in S_n$ such that G is pseudorandom for A on m. Hence, our derandomization algorithm (or, more accurately, the advice that it needs) depends on the distinguisher that it wants to "fool". Thus, for every $L \in \mathcal{BPP}$ decided by algorithm M, and every efficient probabilistic algorithm that "samples" inputs for M according to a distribution $\mathcal{X}$, there exists a corresponding "almost-always" derandomization algorithm $D_{\mathcal{X}}$ (see Proposition 4.12).

Remark: Downward self-reducibility in "few" steps is not crucial. We note that weaker versions of the "almost-always" results above can be obtained without relying on the fact that $f^{\texttt{ws}}$ is downward self-reducible in polylog steps, relying instead only on the fact that we are dealing with *the "high-end" parameter setting* (i.e., small seed length of the PRG and high running time of the algorithm for $f^{\texttt{ws}}$).

Specifically, when transforming A into F (i.e., a distinguisher into an algorithm for $f^{\texttt{ws}}$), in the downward self-reducibility argument, instead of taking $\ell_0 = \ell - \text{polylog}(\ell)$ as the "base case" at which $f^{\texttt{ws}}$ is efficiently-computable, we take the "base case" $\ell_0 = \ell/\text{polylog}(\ell)$, at which we can construct a circuit for $f^{\texttt{ws}}$ by "brute-force" (i.e., in time $2^{\ell_0} = 2^{\ell/\text{polylog}(\ell)}$, which we are allowed). We obtain a set $S_n \subset [n^{1/\text{polyloglog}(n)}, n^2]$ analogous to the set S_n above, and hence under an "almost-everywhere" hypothesized lower bound, for every $n \in \mathbb{N}$, and for a corresponding $\overline{n} = n^{\text{polyloglog}(n)}$, there exists some input length $m \in S_{\overline{n}} \subset [n, n^{\text{polyloglog}(n)}]$ such that G fools A at input length m, and the argument follows through.

This alternative approach suffices to obtain an "almost-always" hitting-set generator with seed length $\tilde{O}(\log(n))$. However, since the set S_n in this approach is of size $\Omega(\ell) = \tilde{O}(\log(n))$, this approach only yields "almost-always" derandomization of $\mathcal{BPP}$ with $O(\text{loglog}(n))$ bits of advice, instead of $O(\text{logloglog}(n))$ as in Theorem 1.2.

2.2 Circuit lower bounds from randomized `CircuitSAT` algorithms

Our proof strategy for Theorem 1.3 is very different from previous proof strategies that deduce circuit lower bounds from "non-trivial" circuit-analysis algorithms(e.g., from the "easy-witness" proof strategy [IKW02; Wil13; MW18; Che19], or from proofs that rely on $\mathcal{MA}$ lower bounds [IKW02, Rmk. 26], [San09; Tel19]). In high-level, to prove our result we exploit the connection between *randomized learning algorithms* and *circuit lower bounds*, which was recently discovered by Oliveira and Santhanam [OS17, Sec. 5] (following [FK09; HH13; KKO13]). Loosely speaking, their connection relies on the classical results of [IW98], and we are able to significantly refine this connection, using our refined version of the [IW98] argument that was detailed in Section 2.1.

Our starting point is the observation that CircuitSAT algorithms yield learning algorithms. Specifically, fix $k \in \mathbb{N}$, and assume (for simplicity) that CircuitSAT for polynomial-sized n-bit circuits can be solved in probabilistic time $2^{n/\mathrm{polylog}(n)}$ for an arbitrarily large polylogarithmic function. We show that in this case, any function that is computable by circuits of size $n \cdot (\log n)^k$ can be learned (approximately) using membership queries in time $2^{n/\mathrm{polylog}(n)}$ (we explain below how to prove this).[15] Now, let $f^{\mathtt{ws}}$ be the well-structured function from Section 2.1, and recall that $f^{\mathtt{ws}}$ is computable in linear space, and hard for linear space under quasilinear-time reductions. Then, exactly one of two cases holds:

1. The function $f^{\mathtt{ws}}$ does not have circuits of size $n \cdot (\log n)^k$. In this case a Boolean version of $f^{\mathtt{ws}}$ also does not have circuits of such size, and since this Boolean version is in $\mathcal{SPACE}[O(n)] \subseteq \mathcal{BPE}$, we are done.

2. The function $f^{\mathtt{ws}}$ has circuits of size $n \cdot (\log n)^k$. Hence, $f^{\mathtt{ws}}$ is also learnable (as we concluded above), and so the argument of [IW98] can be used to show that $f^{\mathtt{ws}}$ is computable by an efficient probabilistic algorithm.[16] Now, by a diagonalization argument, there exists $L^{\mathsf{diag}} \in \Sigma_4[n \cdot (\log n)^{2k}]$ that cannot be computed by circuits of size $n \cdot (\log n)^k$. We show that $L^{\mathsf{diag}} \in \mathcal{BPE}$ by first reducing L^{diag} to $f^{\mathtt{ws}}$ in time $\widetilde{O}(n)$, and then computing $f^{\mathtt{ws}}$ (using the efficient probabilistic algorithm).

Thus, in both cases we showed a function in $\mathcal{BPE} \setminus \mathcal{SIZE}[n \cdot (\log n)^k]$. The crucial point is that in the second case, our new and efficient implementation of the [IW98] argument (which was described in Section 2.1) yields a probabilistic algorithm for $f^{\mathtt{ws}}$ with very little overhead, which allows us to indeed show that $L^{\mathsf{diag}} \in \mathcal{BPE}$. Specifically, our implementation of the argument (with the specific well-structured function $f^{\mathtt{ws}}$) shows that $f^{\mathtt{ws}}$ can be learned in time $T(n) = 2^{n/\mathrm{polylog}(n)}$, then $f^{\mathtt{ws}}$ can be computed in similar time $T'(n) = 2^{n/\mathrm{polylog}(n)}$ (see Corollary 4.10).

We thus only need to explain how a CircuitSAT algorithm yields a learning algorithm with comparable running time. The idea here is quite simple: Given oracle access to a function $f^{\mathtt{ws}}$, we generate a random sample of $r = \mathrm{poly}(n)$ labeled examples $(x_1, f^{\mathtt{ws}}(x_1)), ..., (x_r, f^{\mathtt{ws}}(x_r))$ for $f^{\mathtt{ws}}$, and we use the CircuitSAT algorithm to construct, bit-by-bit, a circuit of size $n \cdot (\log n)^k$ that agrees with $f^{\mathtt{ws}}$ on the sample. Note that the input for the CircuitSAT algorithm is a circuit of size $\mathrm{poly}(n)$ over only $n' \approx n \cdot (\log n)^{k+1}$ bits (corresponding to the size of the circuit that we wish to construct). Hence, the CircuitSAT algorithm runs in time $2^{n'/\mathrm{polylog}(n')} = 2^{n/\mathrm{polylog}(n)}$. And if the sample size $r = \mathrm{poly}(n)$ is large enough, then with high probability *any*

[15]That is, there exists a probabilistic algorithm that gets input 1^n and oracle access to the function f, and with high probability outputs an n-bit circuit of size $n \cdot (\log n)^k$ that agrees with f on almost all inputs.

[16]Actually, our implementation of the [IW98] argument shows that if the function $\mathtt{ECC}(f^{\mathtt{ws}})$ (where ECC is defined as in Section 2.1) can be learned, then the function $f^{\mathtt{ws}}$ can be efficiently computed. For simplicity, we ignore the difference between $f^{\mathtt{ws}}$ and $\mathtt{ECC}(f^{\mathtt{ws}})$ in the current high-level decription.

circuit of size $n \cdot (\log n)^k$ that agrees with $f^{\mathtt{ws}}$ on the sample also agrees with $f^{\mathtt{ws}}$ on almost all inputs(i.e., by a union-bound over all circuits of such size).

2.3 $\mathcal{NTIME}$-uniform circuits for $\mathcal{E}$ and the equivalence between derandomization and circuit lower bounds

The proofs that we describe in the current section are technically much simpler than the proofs described in Sections 2.1 and 2.2. We first describe the proof idea for Theorems 1.4 and 1.5, which assert that if $\mathcal{E}$ does not have $\mathcal{NTIME}[T]$-uniform circuits (for a relatively-small T such as $T(n) = 2^{n^{\epsilon}}$), then derandomization is equivalent to lower bounds. Recall that by known non-uniform "hardness-to-randomness" results (specifically, using Umans' [Uma03] pseudorandom generator; see Corollary 3.4), the circuit lower bounds in the conclusions of the theorems *unconditionally* imply the corresponding derandomization results. Therefore, it suffices to show that under our hypothesis, each derandomization result implies the corresponding circuit lower bound.

For concreteness, let us focus on the setting of Theorem 1.4. The main underlying result is a refinement of classical "Karp-Lipton-style" theorems. Specifically, for "nice" functions S and T, assume that $pr\mathcal{BPP} \subseteq pr\mathcal{NTIME}[T]$ and assume (towards a contradiction) that $\mathcal{E}$ "collapses" to $\mathcal{SIZE}[S]$. The classical result of [Bab+93] asserts that in this case $\mathcal{E} \subseteq \mathcal{NTIME}[T']$, for $T'(n) \approx T(S(n))$.[17] We prove the stronger conclusion that $\mathcal{E}$ can be decided by *$\mathcal{NTIME}[T']$-uniform circuits of size approximately* $S(n)$, where $T'(n) \approx T(S(S(n)))$. (See Proposition 6.6.)

To do so, fix a proof system for $\mathcal{E}$ with a polynomial-time verifier and a prover that is computable in $\mathcal{E}$; concretely, we use an instance checker for $\mathcal{E}$-complete sets (see Proposition 6.4). Recall that in the argument of [Bab+93], an $\mathcal{MA}$ verifier gets input x and non-deterministically guesses a *concise proof* that $x \in L$ (or that $x \notin L$), where the latter proof exists by applying the "collapse" hypothesis to the $\mathcal{E}$-computable prover in the foregoing proof system. Our proof follows from two key observations. The first observation is that the hypothesis actually implies that there exists a *single concise representation of a prover* (by a small circuit) for all inputs x, rather than a separate concise proof for each x. Upon receiver this "prover-circuit", the $\mathcal{MA}$ verifier can check (with high probability) that for almost all inputs x, the "prover-circuit" convinces the verifier (in the underlying proof system) whether x is in L or not. Then, assuming that L is also randomly self-reducible (which may be assumed without loss of generality – see Proposition 6.4), the verifier can construct a small probabilistic circuit that gets input x, invokes the random self-reducibility algorithm for L, and answers each of the queries of this algorithm by simulating the proof system using the concise prover.

The second observation is that this construction can be derandomized, relying both on the derandomization hypothesis and on the collapse hypothesis. First, since we assumed that $pr\mathcal{BPP} \subseteq pr\mathcal{NTIME}[T]$, the verifier can replace its initial probabilistic verification of the "prover-circuit" with non-deterministic verification (see Step 2 in the proof of Proposition 6.6). Secondly, to derandomize the resulting probabilistic circuit,

[17] This follows from a straightforward adaptation of their argument; see Proposition 6.5.

assume (by error-reduction) that there exists a random string that causes the circuit to correctly decide L on all inputs. The verifier then non-deterministically constructs such a string, bit-by-bit. Note that in each iteration we only need to solve a problem in $\mathcal{E}$: This is since in each iteration we have an input $(\langle C \rangle, \sigma)$ where σ is a prefix of the $|C|$-bit random string, and we want to decide whether or not we can extend $(\sigma, 0)$ and $(\sigma, 1)$ to a $|C|$-bit string in a way that will cause C to correctly compute $L \in \mathcal{E}$ on all 2^n inputs. The point is that now we can use the *classical* result of [Bab+93], which (as mentioned above) asserts that under our joint hypotheses we have that $\mathcal{E} \subseteq \mathcal{NTIME}[T']$ for $T' \approx T \circ S$. This allows us to non-deterministically construct a "good" random string in time approximately $T'(|C|) \approx T(S(S(n)))$. Moreover, despite the overhead in our running time, the resulting circuit is still of size $|C| \approx S(n)$.

Being a bit more careful in our construction of the probabilistic circuit C, we can show a construction that uses only $O(n)$ random bits (this relies on randomness-efficient error-reduction and on re-using certain random strings). Relying on this, we show that the hypothesis $pr\mathcal{BPP} \subseteq pr\mathcal{NTIME}[T]$ can actually be relaxed, assuming only that the circuit acceptance probability problem *for circuits of size approximately* $S(n)$ can be solved in non-deterministic time T.[18]

Proof of Theorem 1.6. Recall that Theorem 1.6 asserts that if non-deterministic derandomization is equivalent to circuit lower bounds, then $\mathcal{E}$ does not have $\mathcal{NTIME}[T]$-uniform circuits, for a relatively-small function T. Let us describe the general proof idea, which is very simple, without being specific regarding the parameters.

Assume towards a contradiction that $\mathcal{E}$ has $\mathcal{NTIME}[T]$-uniform circuits. Then, we can construct an efficient algorithm for non-deterministic derandomization: Specifically, since our assumption implies that $\mathcal{E} \subseteq \mathcal{NTIME}[T] \cap co\mathcal{NTIME}[T]$, we can non-deterministically guess-and-verify in time $T(\text{poly}(n))$ the truth-table of the lex-first function on $O(\log(n))$ bits that is "hard" for circuits of size $O(n)$ (see Claim 6.10). We can then use this function to instantiate Umans' [Uma03] PRG, and deduce that $pr\mathcal{BPP} \subseteq \cup_{c \in \mathbb{N}} pr\mathcal{NTIME}[T(\text{poly}(n))^c]$.

Now, relying on the assumed equivalence between non-deterministic derandomization and circuit lower bounds, we deduce that $\mathcal{E}$ does *not* have *non-uniform* circuits of a corresponding size S. The point is that if $S > T$ then we reach a contradiction: This is since if $\mathcal{E}$ does not have non-uniform circuits of size T, then it certainly does not have uniform circuits, of any kind, of such size. Indeed, we want to choose the maximal value of T such that derandomization of $pr\mathcal{BPP}$ in time $\text{poly}(T(\text{poly}(n)))$ is equivalent to lower bounds for circuits of size S such that $S > T$. Theorem 1.6 follows by showing that this holds for $T(n) = 2^{2^{\text{polyloglog}(n)}}$, and (as mentioned in Section 1) we actually show a much tighter result (see Theorem 6.11).

[18]For a definition of the Circuit Acceptance Probability Problem (CAPP), see Definition 3.1. Recall that the existence of an algorithm that solves CAPP for n-bit circuits of *large* size in time $T(n)$ is a weaker hypothesis than the existence of an algorithm that solves CAPP on general n-bit circuits in time $T(n)$.

An additional implication of the conditional equivalence between *non-deterministic* derandomization of $pr\mathcal{BPP}$ and circuit lower bounds. Recall that the known *unconditional* implications of derandomization only yield lower bounds against non-deterministic classes (e.g., $\mathcal{NEXP}$ or $\mathcal{NP}$, depending on the strength of the hypothesis), rather than against $\mathcal{E}$ (see, e.g., [BFT98; IKW02; KI04; JS12; Wil13; MW18; Tel19; Che19; CW19]). A natural explanation for this shortcoming is that known results follow from the hypothesis that $pr\mathcal{BPP}$ can be simulated *non-deterministically* (i.e., that $pr\mathcal{BPP} \subseteq pr\mathcal{NTIME}[T]$ for some small T), and thus it seems intuitive that the corresponding conclusion only yields circuit lower bounds against *non-deterministic* classes.

However, Theorem 1.4 provides evidence that *counters* this explanation: Under the (very weak) hypothesis of Theorem 1.4, non-deterministic derandomization of $pr\mathcal{BPP}$ already suffices to deduce circuit lower bounds against $\mathcal{E}$.

3 Preliminaries

We denote random variables in boldface. For an alphabet Σ and $n \in \mathbb{N}$, we denote the uniform distribution over Σ^n by $\mathbf{u}_n$, where Σ will be clear from context.

For any set $L \subseteq \{0,1\}^*$ and $n \in \mathbb{N}$, we denote by $L_n = L \cap \{0,1\}^n$ the restriction of L to n-bit inputs. Similarly, for $f : \{0,1\}^* \to \{0,1\}^*$, we denote by $f_n : \{0,1\}^n \to \{0,1\}^*$ the restriction of f to the domain of n-bit inputs.

3.1 Two exponential-time hypotheses

We define two exponential-time hypotheses that we consider in this paper. We note in advance that our actual results refer to various *weaker variants* of these hypotheses.

Hypothesis 1 (rETH; see [Del+14])**.** Randomized Exponential Time Hypothesis (rETH)*: There exists $\epsilon > 0$ and $c > 1$ such that* 3-SAT *on n variables and with $c \cdot n$ clauses cannot be solved by* probabilistic *algorithms that run in time $2^{\epsilon \cdot n}$.*

Hypothesis 2 (NETH; see [Car+16])**.** Non-Deterministic Exponential Time Hypothesis (NETH)*: There exists $\epsilon > 0$ and $c > 1$ such that co-*3-SAT *on n variables and with $c \cdot n$ clauses cannot be solved by* non-deterministic *algorithms that run in time $2^{\epsilon \cdot n}$.*

We also extend the two foregoing hypotheses to stronger versions in which every algorithm (probabilistic or non-deterministic, respectively) fails to compute the corresponding "hard" function on *all but finitely-many* input lengths. These stronger hypotheses are denoted a.a.-rETH, and a.a.-NETH, respectively.

3.2 Worst-case Derandomization and Pseudorandom Generators

We now formally define the circuit acceptance probability problem (or CAPP, in short); this well-known problem is also sometimes called Circuit Derandomization, Approx Circuit Average, and GAP-SAT or GAP-UNSAT.

Definition 3.1 (CAPP)**.** *The* circuit acceptance probability problem *with parameters $\alpha, \beta \in [0,1]$ such that $\alpha > \beta$ and for size $S : \mathbb{N} \to \mathbb{N}$ (or (α,β)-*CAPP$[S]$*, in short) is the following promise problem:*

- *The YES instances are (representations of) circuits over n input bits of size at most $S(n)$ that accept at least an α fraction of their inputs.*
- *The NO instances are (representations of) circuits over n input bits of size at most $S(n)$that accept at most a β fraction of their inputs.*

We define the CAPP$[S]$ *problem (i.e., omitting α and β) as the $(2/3, 1/3)$-CAPP problem. We define* CAPP *to be the problem when there is no restriction on S.*

It is well-known that CAPP is complete for $pr\mathcal{BPP}$ under deterministic polynomial-time reductions; in particular, CAPP can be solved in deterministic polynomial time *if and only if $pr\mathcal{BPP} = pr\mathcal{P}$.*

Proposition 3.2 (CAPP is equivalent to $pr\mathcal{BPP} = pr\mathcal{P}$)**.** *The circuit acceptance probability problem can be solved in deterministic polynomial time* if and only if *$pr\mathcal{BPP} = pr\mathcal{P}$.*

For a proof of Proposition 3.2 see any standard textbook on the subject (e.g. [Vad12, Cor. 2.31], [Gol08, Exer. 6.14]).

We will need the following well-known construction of a pseudorandom generator from a function that is "hard" for non-uniform circuits, by Umans [Uma03] (following the line of works initiated by Nisan and Wigderson [NW94]).

Theorem 3.3 (Umans' PRG; see [Uma03, Thm. 6])**.** *There exists a constant $c > 1$ and an algorithm G such that the following holds. When G is given an n-bit truth-table of a function $f : \{0,1\}^{\log(n)} \to \{0,1\}$ that cannot be computed by circuits of size s, and a random seed of length $\ell(n) = c \cdot \log(n)$, it runs in time n^c, and for $m = s^{1/c}$ outputs an m-bit string that is $(1/m)$-pseudorandom for every size-m circuit over m bits.*

Corollary 3.4 (near-optimal non-uniform hardness-to-randomness using Umans' PRG)**.** *There exists a universal constant $c > 0$ such that the following holds. Let $S : \mathbb{N} \to \mathbb{N}$ be a time-computable function, and assume that $\mathcal{DTIME}[2^n] \not\subset \mathtt{i.o.}\mathcal{SIZE}[S]$. Then* CAPP $\in$ *$pr\mathcal{DTIME}[T]$ where $T(n) = 2^{c \cdot S^{-1}(n^c)}$. Moreover, if $\mathcal{DTIME}[2^n] \not\subset \mathcal{SIZE}[S]$ then* CAPP $\in \mathtt{i.o.}pr\mathcal{DTIME}[T]$.

In addition we will need a suitable construction of an averaging sampler. Recall the standard definition of averaging samplers:

Definition 3.5 (averaging sampler)**.** *A function $Samp : \{0,1\}^{m'} \to (\{0,1\}^m)^D$ is an* averaging sampler with accuracy ϵ and confidence δ *(or (ϵ,δ)-*averaging sampler*, in short) if for every $T \subseteq \{0,1\}^m$, the probability over choice of $x \in \{0,1\}^{m'}$ that $\Pr_{i\in[D]}[Samp(x)_i \in T] \notin |T|/2^m \pm \epsilon$ is at most δ.*

We will specifically use the following well-known construction by Guruswami, Umans, and Vadhan [GUV09]. (The construction in [GUV09] is of an extractor, rather than of an averaging sampler, but the two are well-known to be essentially equivalent; see, e.g., [Gol08, Sec. D.4.1.2] or [Vad12, Cor. 6.24].)

Theorem 3.6 (the near-optimal extractor of [GUV09], instantiated as a sampler and for specific parameters)**.** *Let $c \geq 1$ be an arbitrarily large constant. Then, there exists a polynomial-time algorithm that for every m computes an $(n^{-c}, 2^{-2m})$-averaging sampler $Samp\colon \{0,1\}^{m'} \to (\{0,1\}^m)^D$, where $m' = O(m)$ and $D = \text{poly}(m)$.*

3.3 Average-case Derandomization and Pseudorandom Generators

We now define the notions of "average-case" derandomization of probabilistic algorithms. The first definitions that we need are of circuits that distinguish a distribution from uniform, and of distributions that are pseudorandom for uniform algorithms, as follows:

Definition 3.7 (distinguishing distributions from uniform)**.** *For two functions $\mathtt{str}, \ell : \mathbb{N} \to \mathbb{N}$, let G be an algorithm that gets input 1^n and a random seed of length $\ell(n)$ and outputs a string of length $\mathtt{str}(n)$. Then:*

1. *For $n \in \mathbb{N}$ and $n' \in \mathtt{str}^{-1}(n)$, we say that $D_n : \{0,1\}^n \to \{0,1\}$ ϵ-distinguishes $G(1^{n'}, \mathbf{u}_{\ell(n')})$ from uniform if $\left| \Pr[D_n(G(1^{n'}, \mathbf{u}_{\ell(n')})) = 1] - \Pr[D_n(\mathbf{u}_n) = 1] \right| > \epsilon$.*
2. *For a probabilistic algorithm A, an integer n, and $\epsilon > 0$, we say that $G(1^n, \mathbf{u}_{\ell(n)})$ is ϵ-pseudorandom for A if the probability that $A(1^{\mathtt{str}(n)})$ outputs a circuit that ϵ-distinguishes $G(1^n, \mathbf{u}_{\ell(n)})$ from uniform is at most ϵ.*

When applying this definition without specifying a function $\mathtt{str}$, we assume that $\mathtt{str}$ is the identity function.

We now use Definition 3.7 to define pseudorandom generators for uniform circuits and hitting-set generators for uniform circuits, which are analogous to the standard definitions of PRGs and HSGs for non-uniform circuits:

Definition 3.8 (PRGs for uniform circuits)**.** *For $\ell : \mathbb{N} \to \mathbb{N}$, let G be an algorithm that gets as input 1^n and a random seed of length $\ell(n)$, and outputs strings of length n. For $t, a : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to (0,1)$, we say that G is an ϵ-i.o.-PRG for (t,a)-uniform circuits if for every probabilistic algorithm A that runs in time $t(n)$ and gets $a(n)$ bits of non-uniform advice there exists an infinite set $S_A \subseteq \mathbb{N}$ such that for every $n \in S_A$ it holds that $G(1^n, \mathbf{u}_{\ell(n)})$ is $\epsilon(n)$-pseudorandom for A. If for every such algorithm A there is a set S_A as above that contains all but finitely-many inputs, we say that G is an ϵ-PRG for (t,a)-uniform circuits.*

Definition 3.9 (HSGs for uniform circuits)**.** *For $\ell : \mathbb{N} \to \mathbb{N}$, let H be an algorithm that gets as input 1^n and a random seed of length $\ell(n)$, and outputs strings of length n. For*

*$t, a : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to (0,1)$, we say that H is an ϵ-*HSG *for* (t,a)-uniform circuits *if the following holds. For every probabilistic algorithm A that gets input 1^n and $a(n)$ bits of non-uniform advice, runs in time $t(n)$, and outputs a circuit $D_n : \{0,1\}^n \to \{0,1\}$, and every sufficiently large $n \in \mathbb{N}$, with probability at least $1 - \epsilon(n)$ (over the coin tosses of A) at least one of the following two cases holds:*

1. *There exists $s \in \{0,1\}^{\ell(n)}$ such that $D_n(G(1^n, s)) = 1$.*
2. *The circuit D_n satisfies $\Pr_{x \in \{0,1\}^n}[D_n(x) = 1] \leq \epsilon(n)$.*

As mentioned in Section 1, PRGs for uniform circuits can be used to derandomize $\mathcal{BPP}$ "on average" (see, e.g., [Gol11, Prop. 4.4]). Analogously, *HSGs* for uniform circuits can be used to derandomize $\mathcal{RP}$ "on average". That is, loosely speaking, if there exists an HSG for uniform circuits, then for any $L \in \mathcal{RP}$ there exists a deterministic algorithm D such that for every efficiently-samplable distribution $\mathcal{X}$, the probability over $x \sim \mathcal{X}$ that $D(x) \neq L(x)$ is small. For simplicity, we prove the foregoing claim for HSGs that are computable in polynomial time and have logarithmic seed length:

Claim 3.10 (HSGs for uniform circuits $\Rightarrow$ derandomization of $\mathcal{RP}$ "on average")**.** *For $\epsilon : \mathbb{N} \to (0,1)$ such that $\epsilon(n) \leq 1/3$, assume that for every $k \in \mathbb{N}$ there exists a ϵ-HSG for $(n^k, 0)$-uniform circuits that is polynomial-time computable and that has logarithmic seed length. Then, for every $L \in \mathcal{RP}$ and every $c \in \mathbb{N}$, there exists a deterministic polynomial-time algorithm D such that for every probabilistic algorithm F that runs in time n^c and every sufficiently large $n \in \mathbb{N}$, the probability (over the internal coin tosses of F) that $F(1^n)$ outputs a string $x \in \{0,1\}^n$ such that $D(x) \neq L(x)$ is at most $\epsilon(n)$.*

Proof. Let M be an $\mathcal{RP}$ machine that decides L in time $n^{c'}$, for some $c' \in \mathbb{N}$. The deterministic algorithm D gets input $x \in \{0,1\}^n$, enumerates the seeds of the HSG for output length $m = n^{c'}$ and with the parameter $k = O(1 + c/c')$, and accepts x if and only if there exists an output r of the HSG such that M accepts x with random coins r. Note that D never accepts inputs $x \notin L$ (since M is an $\mathcal{RP}$ machine), and thus we only have to prove that for every algorithm F as in the claim's statement, the probability that $x = F(1^n)$ satisfies both $x \in L$ and $D(x) = 0$ is at most $\epsilon(n)$.

To do so, let F be a probabilistic algorithm that runs in time n^c. Consider the probabilistic algorithm A that, on input 1^m, runs the algorithm F on input 1^n to obtain $x \in \{0,1\}^n$, and outputs a circuit $C_{m,x} : \{0,1\}^m \to \{0,1\}$ that computes the decision of M at input x as a function of M's $m = n^{c'}$ random coins. Note that the algorithm A runs in time at most $m^{O(1+c/c')}$, and also note that the only probabilistic choices that A makes are a choice of $x = F(1^n)$. Thus, by Definition 3.9 for every sufficiently large m, with probability at least $1 - \epsilon(m) > 1 - \epsilon(n)$ over choice of $x = F(1^n)$ (i.e., over the coin tosses of A), if $D(x) = 0$ then $\Pr_r[C_{m,x}(r) = 1] = \Pr[M(x) = 1] \leq \epsilon(n) \leq 1/3$, which means that $x \notin L$. ■

4 rETH and near-optimal uniform hardness-to-randomness

In this section we prove Theorems 1.1 and 1.2. First, in Section 4.1, we define and construct well-structured functions, which are the key technical component in our proof of Theorem 1.1. Then, in Section 4.2 we show how well-structured functions can be used in the proof framework of [IW98] (with minor variations) to construct a PRG that "fools" uniform circuits, assuming that the well-structured function cannot be computed by efficient probabilistic algorithms. Finally, in Section 4.3 we prove Theorems 1.1 and 1.2.

4.1 Construction of a well-structured function

In Section 4.1.1 we present the required properties of well-structured functions and define such functions. Then, in Section 4.1.2 we present a high-level overview of our construction of such functions. Finally, in Section 4.1.3 we present the construction itself in detail.

4.1.1 Well-structured function: Definition

Loosely speaking, we will say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is well-structured if it satisfies three properties. The *first property*, which is not crucial for our proofs but simplifies them a bit, is that f is length-preserving; that is, for every $x \in \{0,1\}^*$ it holds that $|f(x)| = |x|$.

The *second property* is a strengthening of the notion of downwards self-reducibility. Recall that a function $f : \{0,1\}^* \to \{0,1\}^*$ is downwards self-reducible if f_n can be computed by an efficient algorithm that has oracle access to f_{n-1}. First, we quantify the notion of "efficient", in order to also allow for very large running time (e.g., running time $2^{n/\text{polylog}(n)}$). Secondly, we also require that for any $n \in \mathbb{N}$ there exists an input length m that is not much smaller than n such that f_m is efficiently computable *without any "downward" oracle*. That is, intuitively, if we try to compute f on input length n by "iterating downwards" using downward self-reducibility, our "base case" in which the function is efficiently-computable is not input length $O(1)$, but a large input length m that is not much smaller than n. More formally:

Definition 4.1 (downward self-reducibility in few steps). *For $t, s : \mathbb{N} \to \mathbb{N}$, we say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is* downward self-reducible in time t and s steps *if there exists a probabilistic oracle machine A that for any sufficiently large $n \in \mathbb{N}$ satisfies the following.*

1. *When A is given input $x \in \{0,1\}^n$ and oracle access to f_{n-1}, it runs in time at most $t(n)$ and satisfies $\Pr_r[A^{f_{n-1}}(x,r) = f(x)] \geq 2/3$.*

2. *There exists an input length $m \in [n - s(n), n]$ such that A computes f_m in time $t(m)$ without using randomness or oracle queries..*

In the special case that $s(n) = n$, we simply say that f is downward self-reducible in time t.

The third property that we need is a refinement of the notion of random self-reducibility, which is called sample-aided worst-case to average-case reducibility. This notion was recently made explicit by Goldreich and G. Rothblum [GR17], and is implicit in many previous results (see, e.g., the references in [GR17]).

To explain the notion, recall that if a function f is randomly self-reducible, then a circuit $\widetilde{C}$ that computes f on *most* of the inputs can be efficiently transformed to a (probabilistic) circuit C that computes f on *every* input (whp). We want to relax this notion, by allowing the efficient algorithm that transforms $\widetilde{C}$ into C to obtain *random labeled samples for f* (i.e., inputs of the form $(r, f(r))$ where r is chosen uniformly at random). The main advantage in this relaxation is that we will not need to assume that $\widetilde{C}$ computes f on *most* of the inputs, but will be satisfied with the weaker assumption that $\widetilde{C}$ computes f on a *tiny fraction* of the inputs. Specifically:[19]

Definition 4.2 (sample-aided reductions; see [GR17, Def 4.1]). *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a length-preserving function, and let $s : \mathbb{N} \to \mathbb{N}$ and $\delta_0 : \mathbb{N} \to [0,1)$. Let M be a probabilistic oracle machine that gets input 1^n and a sequence of $s(n)$ pairs of the form $(r,v) \in \{0,1\}^n \times \{0,1\}^n$ and oracle access to a function $\tilde{f}_n : \{0,1\}^n \to \{0,1\}^n$, and outputs a circuit $C : \{0,1\}^n \to \{0,1\}^n$ with oracle gates. We say that M is a* sample-aided reduction of computing f in the worst-case to computing f on δ_0 of the inputs using a sample of size s *if for every $\tilde{f}_n : \{0,1\}^n \to \{0,1\}^n$ satisfying $\Pr_{x\in\{0,1\}^n}[\tilde{f}_n(x) = f_n(x)] \geq \delta_0(n)$ the following holds: With probability at least $1 - \delta_0(n)$ over choice of $\bar{r} = r_1, ..., r_{s(n)} \in \{0,1\}^n$ and over the internal coin tosses of M, we have that $M^{\tilde{f}_n}(1^n, (r_i, f_n(r_i))_{i\in[s(n)]})$ outputs a circuit C such that $\Pr[C^{\tilde{f}_n}(x) = f_n(x)] \geq 2/3$ for every $x \in \{0,1\}^n$.*

Definition 4.3 (sample-aided worst-case to average-case reducibility). *For $\delta_0 : \mathbb{N} \to (0,1)$, we say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is* sample-aided worst-case to δ_0-average-case reducible *if there exists a sample-aided reduction M of computing f in worst-case to computing f on δ_0 of the inputs such that M runs in time $\mathrm{poly}(n, 1/\delta_0(n))$ and uses $\mathrm{poly}(1/\delta_0(n))$ samples.*

For high-level intuition of why labeled samples can be helpful for worst-case to average-case reductions, and for a proof that if f is a low-degree multivariate polynomial then it is sample-aided worst-case to average-case reducible, see Appendix A.

We are now ready to define well-structured functions. Fixing a parameter $\delta > 0$, a function $f^{\mathtt{ws}}$ is δ-well-structured if it is length-preserving, downward self-reducible in time $\mathrm{poly}(1/\delta)$, and sample-aided worst-case to δ-average case reducible. That is:

Definition 4.4 (well-structured function). *For $\delta : \mathbb{N} \to (0,1)$ and $s : \mathbb{N} \to \mathbb{N}$, we say that a function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ is* (δ, s)-well-structured *if $f^{\mathtt{ws}}$ is length-preserving,*

[19] Definition 4.2 is actually a slightly modified version of the definition in [GR17]. First, we consider reductions of computing f in the worst-case to computing f in "rare-case", whereas [GR17] both reduce the computation of f to the computation of a possibly different function f', and parametrize the success probability of computing both f and f'. Secondly, we separately account for the success probability of the transformation M and of the final circuit C. And lastly, we also require f to be length-preserving.

downward self-reducible in time $\mathrm{poly}(1/\delta)$ *and* s *steps, and sample-aided worst-case to* δ*-average-case reducible. Also, when* $s(n) = n$ *(i.e.,* $f^{\mathtt{ws}}$ *is simply downward self-reducible in time* $\mathrm{poly}(1/\delta)$*), we say that* $f^{\mathtt{ws}}$ *is* δ-well-structured*.*

In the following definition, we consider reductions from a decision problem $L \subseteq \{0,1\}^*$ to a well-structured function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$. To formalize this we consider both a reduction R, which transforms any input x for L to an input $R(x)$ for $f^{\mathtt{ws}}$, and a "decision algorithm" D, which translates the non-Boolean result $f^{\mathtt{ws}}(R(x))$ into a decision of whether or not $x \in L$.

Definition 4.5 (reductions to multi-output functions)**.** *Let* $L \subseteq \{0,1\}^*$ *and* $f : \{0,1\}^* \to \{0,1\}^*$*. For* $t, b : \mathbb{N} \to \mathbb{N}$*, we say that* L reduces to f in time t with blow-up b *if there exist two deterministic time-t algorithms R and D such that for every* $x \in \{0,1\}^*$ *it holds that* $|R(x)| \le b(|x|)$ *and that* $x \in L$ *if and only if* $D(f(R(x))) = 1$*.*

4.1.2 Overview of our construction

For $\delta = 2^{-n/\mathrm{polylog}(n)}$ and $s = \mathrm{polylog}(n)$, our goal is to construct a (δ, s)-well-structured function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ such that TQBF reduces to $f^{\mathtt{ws}}$ in *quasi-linear time* (and thus with quasilinear blow-up). Throughout the section, assume that an n-bit input to TQBF is simply a 3-SAT formula φ on n variables, and it is assumed that all variables are quantified in-order, with alternating quantifiers (e.g., $\forall w_1 \exists w_2 \forall w_3 ... \varphi(w_1, ..., w_n)$; see Definition 4.6).

Our starting point is the well-known construction of Trevisan and Vadhan [TV07], which (loosely speaking) transforms the protocol underlying the $\mathcal{IP} = \mathcal{PSPACE}$ proof into a computational problem $L_{TV} : \{0,1\}^* \to \{0,1\}^*$.[20] They required that L_{TV} will meet the weaker requirements (compared to our requirements) of being downward self-reducible and randomly self-reducible, where the latter means reducible from being worst-case computabile to being computable on, say, .99 of the inputs.

Before describing our new construction, let us first review the original construction of L_{TV}. For every $n \in \mathbb{N}$, fix a corresponding interval $I_n = [N_0, N_1]$ of $r(n) = \mathrm{poly}(n)$ input lengths. The input to L_{TV} at any input length in I_n (disregarding necessary padding) is a pair $(\varphi, w) \in \mathbb{F}^{2n}$, where $\mathbb{F}$ is a sufficiently-large field. If $(\varphi, w) \in \{0,1\}^{2n}$ then we think of φ as representing a 3-SAT formula and of w as representing an assignment. At input length N_0 we define $L_{TV}(\varphi, w) = P(\varphi, w)$, where $P(\varphi, x)$ is a low-degree arithmetized version of the Boolean function $(\varphi, w) \mapsto \varphi(w)$.

Now, recall that the $\mathcal{IP} = \mathcal{PSPACE}$ protocol defines three arithmetic operators on polynomials (two quantification operators and a linearization operator). Then, at input length $N_0 + i$, the problem L_{TV} is recursively defined by applying one of the three arithmetic operators on the polynomial from the previous input length $N_0 + i - 1$.[21]

[20] Actually, in [TV07] they define a *Boolean function*, which treats a suffix of its input as an index of an output bit in the non-Boolean version that we describe, and outputs the corresponding bit. To streamline our exposition we ignore this issue.

[21] In more detail, we define three arithmetic operators on functions $\mathbb{F}^{2n} \to \mathbb{F}$, each indexed by a variable

Observe that computing L_{TV} at input length $N_0 + i$ corresponds to the residual computational problem that the verifier faces at the $(r-i)^{th}$ round of the $\mathcal{IP} = \mathcal{PSPACE}$ protocol, when instantiated for formula φ and with $r = r(n)$ rounds. Indeed, at the largest input length $N_1 = N_0 + r(n)$ the polynomial L_{TV} is simply a low-degree arithmetized version of the function that decides whether or not $\varphi \in \texttt{TQBF}$ (regardless of w); thus, TQBF can be reduced to L_{TV} by mapping $\varphi \in \{0,1\}^n$ to $(\varphi, 1^n) \in \mathbb{F}^{2n}$ and adding padding to get the input to be of length $N_1 = \text{poly}(n)$. Note that L_{TV} is indeed both downward self-reducible (since for each operator O and polynomial P, we can compute $O(P)(\varphi, w)$ in polynomial-time with two oracle queries to P), and randomly self-reducible (since the polynomials have low degree.)

Let us now define our $f^{\texttt{ws}} : \{0,1\}^* \to \{0,1\}^*$, which replaces their L_{TV}, and highlight what is different in our setting. Recall that our main goal is to construct the well-structured function $f^{\texttt{ws}}$ such that TQBF is reducible to $f^{\texttt{ws}}$ with *only quasilinear overhead* in the input length (i.e., we need to avoid polynomial overheads), while keeping the running time of all operations (i.e., of the algorithms for downward self-reducibility and for sample-aided worst-case to rare-case reducibility) to be *at most* $2^{n/\text{polylog}(n)}$.

The first issue, which is relatively easy to handle, is the number of bits that we use to represent an (arithmetized) input (φ, w) for $f^{\texttt{ws}}$. Recall that we want $f^{\texttt{ws}}$ to be worst-case to δ-average-case reducible for a tiny $\delta = 2^{-n/\text{polylog}(n)}$; thus, $f^{\texttt{ws}}$ will involve computing polynomials over a field of large size $|\mathbb{F}| \geq \text{poly}(1/\delta)$. Using the approach of [TV07], we would need $2n \cdot \log(|\mathbb{F}|) = \tilde{\Omega}(n^2)$ bits to represent (φ, w), and thus the reduction from TQBF to $f^{\texttt{ws}}$ would incur a polynomial overhead. This is easily solvable by considering a "low-degree extension" instead of their "multilinear extension": To represent an input $(\varphi, w) \in \{0,1\}^{2n}$ to $f^{\texttt{ws}}$ we will use *few elements in a very large field*. Specifically, we will use $\ell = \text{polylog}(n)$ variables (i.e., the polynomial will be $\mathbb{F}^{2\ell} \to \mathbb{F}$) such that each variable "provides" $O(n/\text{polylog}(n))$ bits of information.

A second problem is constructing a low-degree arithmetization $P(\varphi, w)$ of the Boolean function that evaluates φ at w. In [TV07] they solve this by first reducing TQBF to an intermediate problem TQBF$'$ that is amenable to such low-degree arithmetization; however, their reduction incurs a quadratic blow-up in the input length, which we cannot afford in our setting. To overcome this we reduce TQBF to another intermediate problem, denoted $\texttt{TQBF}^{\texttt{loc}}$, which is amenable to low-degree arithmetization, such that the reduction incurs only a quasilinear blow-up in the input length. (Loosely speaking, we define $\texttt{TQBF}^{\texttt{loc}}$ by applying a very efficient Cook-Levin reduction to the Turing machine that gets input (φ, w) and outputs $\varphi(w)$; see Claim 4.7.1 for precise details.) We then carefully arithmetize $\texttt{TQBF}^{\texttt{loc}}$, while "paying" for this efficient arithmetization by the fact that computing the corresponding polynomial now takes time $\exp(n/\ell) = \text{poly}(1/\delta)$, instead of $\text{poly}(n)$ time as in [TV07] (see Claim 4.7.2).

$j \in [n]$, and denote these operators by $\{\mathcal{O}_k^j\}_{k\in[3],j\in[n]}$. In each recursive step $i \in [r(n)]$, the polynomial corresponding to input length $N_0 + i$ is obtained by applying operator $\mathcal{O}_{k(i)}^{j(i)}$, where $j, k : \mathbb{N} \to [3]$ are polynomial-time computable functions, to the polynomial corresponding to input length $N_0 + i - 1$. Thus, at input length $N_0 + i$, we compute $L_{TV}(\varphi, w)$ by applying i operators on the polynomial P and evaluating the resulting polynomial at (φ, w).

Thirdly, the number of polynomials in the construction of L_{TV} (i.e., the size of the interval I_n) is $r(n) = \mathrm{poly}(n)$, corresponding to the number of rounds in the $\mathcal{IP} = \mathcal{PSPACE}$ protocol. This poses a problem for us since the reduction from TQBF maps an input of length n is to an input of length $N_1 \geq \mathrm{poly}(n)$. We solve this problem by "shrinking" the number of polynomials to be polylogarithmic, using an approach similar to an $\mathcal{IP} = \mathcal{PSPACE}$ protocol with only $\mathrm{polylog}(n)$ rounds and a verifier that runs in time $2^{n/\mathrm{polylog}(n)}$: Intuitively, at each input length, we define $f^{\mathtt{ws}}$ by simultaneously applying $O(\log(1/\delta))$ operators (rather than a single operator) to the polynomial that corresponds to the previous input length. Indeed, as one might expect, this increases the running-time of the downward self-reducibility algorithm to $\mathrm{poly}(1/\delta)$, but we can afford this. Implementing this approach requires some care, since multiple operators will be applied to a single variable (which represents many bits of information), and since the linearization operator needs to be replaced by a "degree-lowering operation" (that will reduce the individual degree of a variable to be $\mathrm{poly}(1/\delta)$); see Claim 4.7.3 for details.

Lastly, we also want our function to be downward self-reducible in $\mathrm{polylog}(n)$ steps (i.e., after $\mathrm{polylog}(n)$ "downward" steps, the function at the now-smaller input length is computable in time $\mathrm{poly}(1/\delta)$ without an oracle). This follows by noting that the length of each interval I_n is now polylogarithmic, and that at the "bottom" input length the function $f^{\mathtt{ws}}$ simply computes the arithmetized version of $\mathtt{TQBF}^{\mathtt{loc}}$, which (as mentioned above) is computable in time $\mathrm{poly}(1/\delta)$.

4.1.3 The construction itself

We consider the standard "totally quantified" variant of the Quantified Boolean Formula (QBF) problem, called Totally Quantified Boolean Formula (TQBF). In this version the quantifiers do not appear as part of the input, and we assume that all the variables are quantified, and that the quantifiers alternate according to the index of the variable (i.e., x_i is quantified by $\exists$ if i is odd, and otherwise quantified by $\forall$).

Definition 4.6 (TQBF). *A string $\varphi \in \{0,1\}^*$ of length $n = |\varphi|$ is in the set TQBF $\subseteq \{0,1\}^*$ if φ is a representation of a 3-SAT formula in variables indexed by $[n]$ such that, denoting the variables by $w_1, ..., w_n$, it holds that $\exists w_1 \forall w_2 \exists w_3 \forall w_4 ... \varphi(w_1, ..., w_n)$. In other words, $\varphi \in$ TQBF if the quantified expression that is obtained by quantifying all n variables, in order of their indices and with alternating quantifiers (starting with $\exists$), evaluates to true.*

Recall that QBF, in which the quantifiers are part of the input, is reducible in linear time to TQBF from Definition 4.6 (by renaming variables and adding dummy variables).

The main result in this section is a construction of a well-structured function $f^{\mathtt{ws}}$ such that TQBF can be reduced to $f^{\mathtt{ws}}$ with only quasilinear blow-up. This construction is detailed in the following lemma:

Lemma 4.7 (a well-structured set that is hard for TQBF under quasilinear reductions). *There exists a universal constant $r \in \mathbb{N}$ such that for every constant $c \in \mathbb{N}$ the following holds. For $\ell(n) = \log(n)^{3c}$ and $\delta(n) = 2^{-n/\ell(n)}$, there exists a $(\delta, O(\ell^2))$-well-structured* *function $f^{\texttt{ws}} : \{0,1\}^* \to \{0,1\}^*$ such that $f^{\texttt{ws}}$ is computable in linear space, and* TQBF *deterministically reduces to $f^{\texttt{ws}}$ in time $n \cdot \log^{2c+r}(n)$.*

Proof. In high-level, we first reduce TQBF to a problem $\texttt{TQBF}^{\texttt{loc}}$ that will have a property useful for arithmetization, and then reduce $\texttt{TQBF}^{\texttt{loc}}$ to a function $f^{\texttt{ws}}$ that we will construct as follows. We will first carefully arithmetize a suitable witness-relation that underlies $\texttt{TQBF}^{\texttt{loc}}$; then transform the corresponding arithmetic version of $\texttt{TQBF}^{\texttt{loc}}$ to a collection of low-degree polynomials that also satisfy a property akin to downward self-reducibility (loosely speaking, these polynomials arise from the protocol underlying the proof of $\mathcal{IP} = \mathcal{PSPACE}$ [Lun+92; Sha92]); and finally "combine" these polynomials to a Boolean function $f^{\texttt{ws}}$ that will "inherit" the useful properties of the low-degree polynomials, and will thus be well-structured.

A variant of TQBF that is amenable to arithmetization. We will need a non-standard variant of TQBF, which we denote by $\texttt{TQBF}^{\texttt{loc}}$, such that TQBF is reducible to $\texttt{TQBF}^{\texttt{loc}}$ with quasilinear blow-up, and $\texttt{TQBF}^{\texttt{loc}}$ has an additional useful property. To explain this property, recall that the verification procedure of a "witness" $w = w_1, ..., w_n$ in TQBF is local, in the following sense: For every fixed φ it holds that $\varphi \in \texttt{TQBF}$ iff $\exists w_1 \forall w_2 ... \; 3SAT(\varphi, w)$, where $3SAT(\varphi, w) = \varphi(w)$ is a relation that can be decided by a conjunction of local conditions on the "witness" w. We want the stronger property that the relation that underlies $\texttt{TQBF}^{\texttt{loc}}$ can be tested by a conjunction of conditions that are local *both in the input and in the witness*. That is, denoting the underlying relation by $\texttt{R-TQBF}^{\texttt{loc}}$, we will have that $x \in \texttt{TQBF}^{\texttt{loc}}$ iff $\exists w_1 \forall w_2 ... \; \texttt{R-TQBF}^{\texttt{loc}}(x, w)$, where $\texttt{R-TQBF}^{\texttt{loc}}$ is a conjunction of local conditions on (x, w). In more detail:

Claim 4.7.1 (a variant of TQBF with verification that is local in both input and witness)**.** *There exists a set $\texttt{TQBF}^{\texttt{loc}} \in \mathcal{SPACE}[O(n)]$ and a relation $\texttt{R-TQBF}^{\texttt{loc}} \subseteq (\{0,1\}^* \times \{0,1\}^*)$ such that $\texttt{TQBF}^{\texttt{loc}} = \{x : \exists w_1 \forall w_2 \exists w_3 \forall w_4 ... (x, w) \in \texttt{R-TQBF}^{\texttt{loc}}\}$, and the following holds.*

1. *(Length-preserving witnesses.) For any $(x, w) \in \texttt{R-TQBF}^{\texttt{loc}}$ it holds that $|w| = |x|$.*

2. *(Verification that is local in both input and witness.) For every $n \in \mathbb{N}$ there exist n functions $\{f_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\}_{i \in [n]}$ such that the mapping $(x, w, i) \mapsto f_i(x, w)$ is computable in quasilinear time and linear space, and each f_i depends on only three variables, and $(x, w) \in \texttt{R-TQBF}^{\texttt{loc}}$ if and only if for all $i \in [n]$ it holds that $f_i(x, w) = 1$.*

3. *(Efficient reduction with quasilinear blow-up.) There exists a deterministic linear-space and quasilinear-time algorithm A that gets as input $\varphi \in \{0,1\}^n$ and outputs $x = A(\varphi)$ such that $\varphi \in$ TQBF if and only if $x \in \texttt{TQBF}^{\texttt{loc}}$.*

Proof. Consider a 3-SAT formula $\varphi \in \{0,1\}^n$ as an input to TQBF, and for simplicity assume that n is even (this assumption is insignificant for the proof and only simplifies the notation). By definition, we have that $\varphi \in$ TQBF if and only if

$$\exists w_1 \forall w_2 \exists w_3 .... \exists w_n \; \varphi(w_1, ..., w_n) = 1 \; .$$

Now, let M be a linear-space and quasilinear-time machine that gets as input (φ, w) and outputs $\varphi(w)$. We use an efficient Cook-Levin transformation of the computation of the machine M on inputs of length $2n$ to a 3-SAT formula, and deduce the following:[22] There exists a linear-space and quasilinear-time algorithm that, on input 1^n, constructs a 3-SAT formula $\Phi_n : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^{\mathtt{ql}(n)} \to \{0,1\}$ of size $\mathtt{ql}(n) = \tilde{O}(n)$ such that for any $(\varphi, w) \in \{0,1\}^n \times \{0,1\}^n$ it holds that $\varphi(w) = 1$ if and only if there exists a unique $w' \in \{0,1\}^{\mathtt{ql}(n)}$ satisfying $\Phi_n(x, w, w') = 1$.

Now, using the formula Φ_n, note that $\varphi \in \{0,1\}^n$ is in TQBF if and only if

$$\exists w_1 \forall w_2 \exists w_3 ... \exists w_n \; \exists w'_1 \exists w'_2 ... \exists w'_{\mathtt{ql}(n)} \; \Phi_n(\varphi, w, w') = 1 \; . \tag{4.1}$$

We slightly modify Φ_n in order to make the suffix of existential quantifiers in Eq. (4.1) alternate with universal quantifiers that are applied to dummy variables. (Specifically, for each $i \in [\mathtt{ql}(n)]$, we rename w'_i to w'_{2i}, which effectively introduces a dummy variable before w'_i.) Denoting the modified formula by Φ'_n, we have that $\varphi \in$ TQBF if and only if

$$\exists w_1 \forall w_2 \exists w_3 ... \exists w_n \forall w'_1 \exists w'_2 \forall w'_3 ... \exists w'_{2\mathtt{ql}(n)} \; \Phi'_n(\varphi, w, w') = 1 \; .$$

We define the relation $\mathtt{R\text{-}TQBF}^{\mathtt{loc}}$ to consist of all pairs (x, w) such that $x = (\varphi, 1^{2\mathtt{ql}(|\varphi|)})$ and $w = (w^{(0)}, w^{(1)}) \in \{0,1\}^{|\varphi|} \times \{0,1\}^{2\mathtt{ql}(|\varphi|)}$ and $\Phi'_{|\varphi|}(\varphi, w^{(0)}, w^{(1)}) = 1$. Indeed, in this case the corresponding set $\mathtt{TQBF}^{\mathtt{loc}}$ is defined by

$$\mathtt{TQBF}^{\mathtt{loc}} = \left\{ (\varphi, 1^{2\mathtt{ql}(|\varphi|)}) : \exists w_1^{(0)} \forall w_2^{(0)} ... \exists w_{|\varphi|}^{(0)} \forall w_1^{(1)} \exists w_2^{(1)} ... \exists w_{2\mathtt{ql}(|\varphi|)}^{(1)} \; \Phi'_{|\varphi|}(\varphi, w^{(0)}, w^{(1)}) = 1 \right\} \; .$$

Note that, by definition, for every $(x, w) \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}$ we have that $|w| = |x|$. To see that $\mathtt{R\text{-}TQBF}^{\mathtt{loc}}$ can be tested by a conjunction of efficiently-computable local conditions, note that an n-bit input to $\mathtt{TQBF}^{\mathtt{loc}}$ is of the form $(\varphi, 1^{2\mathtt{ql}(|\varphi|)}) \in \{0,1\}^m \times \{1\}^{2\mathtt{ql}(m)}$, and recall that Φ'_m is a 3-SAT formula of size $\mathtt{ql}(m) < n$ that can be produced in linear space and quasilinear time from input 1^m. Also, $\mathtt{TQBF}^{\mathtt{loc}}$ is computable in linear space, since on input $(\varphi, 1^{2\mathtt{ql}(|\varphi|)})$ the number of variables that are quantified is $|\varphi| + 2\mathtt{ql}(|\varphi|)$, and since $\Phi'_{|\varphi|}$ can be evaluated in space $O(|\varphi|)$. Lastly, TQBF trivially reduces to $\mathtt{TQBF}^{\mathtt{loc}}$ by adding padding $\varphi \mapsto (\varphi, 1^{2\mathtt{ql}(|\varphi|)})$. □

Arithmetic setting. For any $n \in \mathbb{N}$, let $\ell_0 = \ell_0(n) = \lfloor (\log n)^c \rfloor$, let $n' = \lceil n/\ell_0 \rceil$, let $\delta_0(n) = 2^{-n'}$, and let $\mathbb{F}$ be the field with $2^{5n'} = 1/\mathrm{poly}(\delta_0(n))$ elements. Recall that a representation of such a field (i.e., an irreducible polynomial of degree $5n'$ over $\mathbb{F}_2$) can be found deterministically either in linear space (by a brute-force algorithm) or in time $\mathrm{poly}(n') = \mathrm{poly}(n)$ (by Shoup's [Sho90] algorithm).

Fix a bijection π between $\{0,1\}^{5n'}$ and $\mathbb{F}$ (i.e., π maps any string in $\{0,1\}^{5n'}$ to the bit-representation of the corresponding element in $\mathbb{F}$) such that both π and π^{-1} can be

[22] The algorithm transforms M into an oblivious machine [PF79; GS89], and then applies an efficient Cook-Levin transformation of the oblivious machine to a 3-SAT formula (see, e.g., [AB09, Sec 2.3.4]).

computed in polynomial time and linear space. Let $H \subset \mathbb{F}$ be the set of $2^{n'}$ elements that are represented (via π) by bit-strings with a prefix of n' arbitrary bits and a suffix of $4n'$ zeroes (i.e., $H = \left\{\pi(z) : z = x0^{4n'}, x \in \{0,1\}^{n'}\right\} \subset \mathbb{F}$ such that $|H| = 2^{n'}$).[23]

We will consider polynomials $\mathbb{F}^{2\ell_0} \to \mathbb{F}$, and we think of the inputs to each such polynomial as of the form $(x,w) \in \mathbb{F}^{\ell_0} \times \mathbb{F}^{\ell_0}$. Note that, intuitively, x and w each represent about $5n$ bits of information. When x and w are elements in the subset $H^{\ell_0} \subset \mathbb{F}^{\ell_0}$, we think of them as a pair of n-bit strings that might belong to $\texttt{R-TQBF}^{\texttt{loc}}$.

Arithmetization of $\texttt{R-TQBF}^{\texttt{loc}}$. Our first step is to carefully arithmetize the relation $\texttt{R-TQBF}^{\texttt{loc}}$ within the arithmetic setting detailed above. We will mainly rely on the property that there is a "doubly-local" verification procedure for $\texttt{R-TQBF}^{\texttt{loc}}$.

Claim 4.7.2 (low-degree arithmetization)**.** *There exists a polynomial $P^{\texttt{TQBF}^{\texttt{loc}}} : \mathbb{F}^{2\ell_0} \to \mathbb{F}$ such that the following holds:*

1. *(Low-degree.) The degree of $P^{\texttt{TQBF}^{\texttt{loc}}}$ is at most $O(n \cdot 2^{n'})$.*
2. *(Arithmetizes $\texttt{R-TQBF}^{\texttt{loc}}$.) For every $(x,w) \in H^{\ell_0} \times H^{\ell_0}$ it holds that $P^{\texttt{TQBF}^{\texttt{loc}}}(x,w) = 1$ if $(x,w) \in \texttt{R-TQBF}^{\texttt{loc}}$, and $P^{\texttt{TQBF}^{\texttt{loc}}}(x,w) = 0$ otherwise.*
3. *(Efficiently-computable.) There exists a deterministic algorithm that gets as input $(x,w) \in \mathbb{F}^{2\ell_0}$, runs in time $\mathrm{poly}(|\mathbb{F}|)$, and outputs $P^{\texttt{TQBF}^{\texttt{loc}}}(x,w) \in \mathbb{F}$. There also exists a deterministic linear-space algorithm with the same functionality.*

Proof. We first show a polynomial-time and linear-space algorithm that, given input 1^n, constructs a low-degree polynomial $P_0^{\texttt{TQBF}^{\texttt{loc}}} : \mathbb{F}^{2n'\cdot\ell_0} \to \mathbb{F}$ that satisfies the following: For every $(x,w) \in \mathbb{F}_2^{2n'\cdot\ell_0}$ (i.e., when the input is a string of $2n' \cdot \ell_0 \geq 2n$ *bits*, and we interpret it as a pair $(x,w) \in \{0,1\}^{2n}$) it holds that $P_0^{\texttt{TQBF}^{\texttt{loc}}}(x,w) = 1$ if $(x,w) \in \texttt{R-TQBF}^{\texttt{loc}}(x,w)$, and $P_0^{\texttt{TQBF}^{\texttt{loc}}}(x,w) = 0$ otherwise.

To do so, recall that by Claim 4.7.1 we can construct in polynomial time and linear space a collection of n polynomials $\left\{f_i : \mathbb{F}_2^{2n'\cdot\ell_0} \to \mathbb{F}_2\right\}_{i\in[n]}$ such that for each $i \in [n]$ the polynomial f_i depends only on three variables in the input (x,w), and such that $(x,w) \in \texttt{R-TQBF}^{\texttt{loc}}$ if and only if for all $i \in [n]$ it holds that $f_i(x,w) = 1$. For each $i \in [n]$, let $p_i : \mathbb{F}^{2n'\cdot\ell_0} \to \mathbb{F}$ be the multilinear extension of f_i, which can be evaluated in time $\mathrm{poly}(n)$ and in linear space (since f_i depends only on three variables, and using Lagrange's interpolation formula and the fact that π is efficiently-computable). Then, the polynomial $P_0^{\texttt{TQBF}^{\texttt{loc}}}$ is simply the multiplication of all the p_i's; that is, $P_0^{\texttt{TQBF}^{\texttt{loc}}}(x,w) = \Pi_{i\in[n]} p_i(x,w)$. Note that $P_0^{\texttt{TQBF}^{\texttt{loc}}}$ can indeed be evaluated in time $\mathrm{poly}(n)$ and in linear space, and that the degree of $P_0^{\texttt{TQBF}^{\texttt{loc}}}$ is $O(n)$ (since each p_i is a multilinear polynomial in $O(1)$ variables).

[23]The specific choice of H as the image of $H_0 = \{x0^{4n'} : x \in \{0,1\}^{n'}\}$ under π is immaterial for our argument, as long as we can efficiently decide H_0 and enumerate over H_0.

Now, let $\pi_1^{(H)}, ..., \pi_{n'}^{(H)} : H \to \{0,1\}$ be the "projection" functions such that $\pi_i^{(H)}$ outputs the i^{th} bit in the bit-representation of its input according to π. Abusing notation, we let $\pi_1^{(H)}, ..., \pi_{n'}^{(H)} : \mathbb{F} \to \mathbb{F}$ be the low-degree extensions of the $\pi_i^{(H)}$'s, which are of degree at most $|H| - 1 < 2^{n'}$. Also, for every $\sigma \in \mathbb{F}$, we denote by $\pi^{(H)}(\sigma)$ the string $\pi_1^{(H)}(\sigma), ..., \pi_{n'}^{(H)}(\sigma) \in \mathbb{F}^{n'}$. Note that the mapping of $\sigma \in \mathbb{F}$ to $\pi^{(H)}(\sigma) \in \mathbb{F}^{n'}$ can be computed in time $\mathrm{poly}(|H|) = \mathrm{poly}(|\mathbb{F}|)$ and in linear space (again just using Lagrange's interpolation formula and the fact that π is efficiently-computable).

Finally, we define the polynomial $P^{\mathtt{TQBF}^{\mathtt{loc}}} : \mathbb{F}^{2\ell_0} \to \mathbb{F}$. Intuitively, for $(x, w) \in H^{\ell_0} \times H^{\ell_0}$, the polynomial $P^{\mathtt{TQBF}^{\mathtt{loc}}}$ first uses the $\pi_i^{(H)}$'s to compute the bit-projections of x and w, which are each of length $n' \cdot \ell_0$, and then evaluates the polynomial $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}$ on these $2n' \cdot \ell_0$ bit-projections. More formally, for every $(x, w) \in \mathbb{F}^{2\ell_0}$ we define

$$P^{\mathtt{TQBF}^{\mathtt{loc}}}(x, w) = P_0^{\mathtt{TQBF}^{\mathtt{loc}}}\left(\pi^{(H)}(x_1), ..., \pi^{(H)}(x_{\ell_0}), \pi^{(H)}(w_1), ..., \pi^{(H)}(w_{\ell_0})\right) .$$

The first item in the claim follows since for every $i \in [n']$ the degree of $\pi_i^{(H)}$ is less than $2^{n'}$, and since $\deg(P_0^{\mathtt{TQBF}^{\mathtt{loc}}}) = O(n)$. The second item in the claim follows immediately from the definition of $P^{\mathtt{TQBF}^{\mathtt{loc}}}$. And the third item in the claim follows since $\pi^{(H)}$ can be computed in time $\mathrm{poly}(|\mathbb{F}|)$ and in linear space, and since $P_0^{\mathtt{TQBF}^{\mathtt{loc}}}$ can be constructed and evaluated in polynomial time and in linear space. (The two different algorithms are since we need to find an irreducible polynomial, which can be done either in linear space or in time $\mathrm{poly}(n) < \mathrm{poly}(|\mathbb{F}|)$.) $\square$

Constructing a "downward self-reducible" collection of low-degree polynomials. Our goal now is to define a collection of $O(\ell_0^2)$ polynomials $\{P_{n,i} : \mathbb{F}^{2\ell_0} \to \mathbb{F}\}_{i \in [O(\ell_0^2)]}$ such that the polynomials are of low degree, and $P_{n,1}$ essentially computes $\mathtt{TQBF}^{\mathtt{loc}}$, and computing $P_{n,i}$ can be reduced in time $\mathrm{poly}(1/\delta_0(n))$ to computing $P_{n,i+1}$. The collection and its properties are detailed in the following claim:

Claim 4.7.3. *There exists a collection of $\bar{\ell}_0 = \ell_0(2\ell_0 + 1) + 1$ polynomials, denoted $\{P_{n,i} : \mathbb{F}^{2\ell_0} \to \mathbb{F}\}_{i \in [\bar{\ell}_0]}$, that satisfies the following:*

1. (Low degree:) *For every $i \in [\bar{\ell}_0]$, the degree of $P_{n,i}$ is at most $O(n \cdot \ell_0 \cdot 2^{2n'})$.*
2. ($P_{n,1}$ computes $\mathtt{TQBF}^{\mathtt{loc}}$ on H-inputs:) *For any $(x, w) \in H^{\ell_0} \times H^{\ell_0}$ it holds that $P_{n,1}(x, w) = 1$ if $x \in \mathtt{TQBF}^{\mathtt{loc}}$, and $P_{n,1}(x, w) = 0$ if $x \notin \mathtt{TQBF}^{\mathtt{loc}}$. (Regardless of w.)*
3. ("Forward" self-reducible:) *For every $i \in [\bar{\ell}_0]$ it holds that $P_{n,i}$ can be computed in time* $\mathrm{poly}(2^{n'})$ *when given oracle access to $P_{n,i+1}$.*
4. (Efficiently-computable:) *The polynomial $P_{n,\bar{\ell}_0}$ can be computed in time* $\mathrm{poly}(2^{n'})$. *Moreover, for every $i \in [\bar{\ell}_0]$ it holds that $P_{n,i}$ can be computed in space $O(n \cdot \bar{\ell}_0)$.*

Proof. For simplicity of notation, assume throughout the proof that n' is even. Towards defining the collection of polynomials, we first define two operators on functions $p : \mathbb{F}^{2\ell_0} \to \mathbb{F}$. Loosely speaking, the first operator corresponds to n' alternating quantification steps in the $\mathcal{IP} = \mathcal{PSPACE}$ proof (i.e., n' steps of alternately quantifying the next variable either by $\exists$ or by $\forall$), and the second operator roughly corresponds to a linearization step that is simultaneously applied to n' variables. In both cases, the n' variables that we consider are the bits in the representation of a single element in the second input to p.

<u>Quantifications operator:</u> Let $i \in [\ell_0]$. Loosely speaking, $\texttt{Quant}^{(i)}(p)$ causes p to ignore the i^{th} variable of its second input, and instead consider alternating quantification steps applied to the bits that represent this variable. To do this, we define a sequence of functions such that the first function replaces the i^{th} variable in the second input for p by a dummy variable in H, and each subsequent function corresponds to a quantification step applied to a single bit in the representation of this dummy variable.

Formally, we recurvisely define $n'+1$ functions $\texttt{Quant}^{(i,0)}, ..., \texttt{Quant}^{(i,n')} = \texttt{Quant}^{(i)}(p)$ such that for $j \in \{0, ..., n'\}$ it holds that $\texttt{Quant}^{(i,j)}(p)$ is a function $\mathbb{F}^{2\ell_0} \times \{0,1\}^{n'-j} \to \mathbb{F}$. The function $\texttt{Quant}^{(i,0)}(p)$ gets as input $(x,w) \in \mathbb{F}^{2\ell_0}$ and $\sigma \in \{0,1\}^{n'}$, ignores the i^{th} element of w, and outputs $\texttt{Quant}^{(i,0)}(x,w,\sigma) = p(x, w_1...w_{i-1}\pi(\sigma 0^{4n'}))$. Then, for $j \in [n']$, if j is odd then we define

$$\texttt{Quant}^{(i,j)}(p)(x,w,\sigma_1...\sigma_{n'-j}) = 1 - \left(\prod_{z\in\{0,1\}} \left(1 - \texttt{Quant}^{(i,j-1)}(p)(x,w,\sigma_1,...,\sigma_{n'-j}z)\right) \right) ,$$

and if j is even then we define

$$\texttt{Quant}^{(i,j)}(p)(x,\sigma_1,...,\sigma_{n'-j}) = \prod_{z\in\{0,1\}} \texttt{Quant}^{(i,j-1)}(p)(x,w,\sigma_1...\sigma_{n'-j}z) .$$

Note that the function $\texttt{Quant}^{(i)}(p)$ can be evaluated at any input in linear space with oracle access to p (since each $\texttt{Quant}^{(i,j)}(p)$ can be evaluated in linear space with oracle access to $\texttt{Quant}^{(i,j-1)}(p)$). Also observe the following property of $\texttt{Quant}^{(i)}(p)$, which follows immediately from the definition:

Fact 4.7.3.1. *If for some $x \in H^{\ell_0}$ and any $w \in H^{\ell_0}$ it holds that $p(x,w) \in \{0,1\}$, then for the same x and any $w \in H^{\ell_0}$ it holds that $\texttt{Quant}^{(i)}(p)(x,w) = 1$ if $\exists\sigma_1\forall\sigma_2\exists\sigma_3...\forall\sigma_{n'}$ such that $p(x, w_1...w_{i-1}\pi(\sigma_1...\sigma_{n'}0^{4n'})w_{i+1}...w_{\ell_0}) = 1$, and $\texttt{Quant}^{(i)}(p)(x,w) = 0$ otherwise.*

<u>Degree-reduction operator:</u> For every fixed $z \in H$, let $I_z : H \to \{0,1\}$ be the indicator function of whether the input equals z, and let $\bar{I}_z : \mathbb{F} \to \mathbb{F}$ be the low-degree extension of I_z, which is of degree at most $|H|-1$ (i.e., $\bar{I}_z(x) = \prod_{h\in H\setminus\{z\}} \frac{x-h}{z-h}$). Then, for any $i \in [\ell_0]$, we define

$$\texttt{DegRed}^{(i)}(p)(x,w) = \sum_{z\in H} \bar{I}_z(x_i) \cdot p(x_1...x_{i-1}zx_{i+1}...x_{\ell_0}, w) ,$$

and similarly for $i \in [2\ell_0]$ we denote $i' = i - \ell_0$ and define

$$\texttt{DegRed}^{(i)}(p)(x,w) = \sum_{z \in H} \bar{I}_z(w_{i'}) \cdot p(x, w_1 ... w_{i'-1} z w_{i'+1} ... w_{\ell_0}) \ .$$

Similarly to the operator $\texttt{Quant}^{(i)}$, note that the function $\texttt{DegRed}^{(i)}(p)$ can be evaluated at any input in linear space with oracle access to p. Also, the definition of the operator $\texttt{DegRed}^{(i)}$ implies that:

Fact 4.7.3.2. *For $i \in [2\ell_0]$, let v be the variable whose degree $\texttt{DegRed}^{(i)}$ reduces (i.e., $v = x_i$ if $i \in [\ell_0]$ and $v = w_{i'} = w_{i-\ell_0}$ if $i \in [2\ell_0]$). Then, the individual degree of v in $\texttt{DegRed}^{(i)}(p)$ is $|H| - 1$, and the individual degree of any other input variable to $\texttt{DegRed}^{(i)}(p)$ remains the same as in p. Moreover, for every $(x,w) \in \mathbb{F}^{\ell_0} \times \mathbb{F}^{\ell_0}$, if the input (x,w) assigns the variable v to a value in H, then $\texttt{DegRed}^{(i)}(p)(x,w) = p(x,w)$.*

Composing the operators: We will be particularly interested in what happens when we first apply the quantifications operator to some variable $i \in [\ell_0]$, and then apply the degree-reduction operator to all variables, sequentially. A useful property of this operation is detailed in the following claim:

Claim 4.7.3.3. *Let $p : \mathbb{F}^{2\ell_0} \to \mathbb{F}$ and $x \in H^{\ell_0}$ such that for any $w \in H^{\ell_0}$ it holds that $p(x,w) \in \{0,1\}$. For $i \in [\ell_0]$, let $p' : \mathbb{F}^{2\ell_0} \to \mathbb{F}$ be the function that is obtained by first applying $\texttt{Quant}^{(i)}$ to p, then applying $\texttt{DegRed}^{(j)}$ for each $j = 1, ..., 2\ell_0$. Then, for any $w' \in H^{\ell_0}$ we have that $p'(x,w') = 1$ if $\exists \sigma_1 \forall \sigma_2 \exists \sigma_3 ... \forall \sigma_{n'} : p(x, w'_1 ... w'_{i-1} \pi(\sigma_1 ... \sigma_{n'}) w'_{i+1} ... w'_{\ell_0}) = 1$, and $p'(x,w') = 0$ otherwise.*

Proof. Fix any $w' \in H^{\ell_0}$. By Fact 4.7.3.1, and relying on the hypothesis that for any $w \in H^{\ell_0}$ we have that $p(x,w) \in \{0,1\}$, it follows that $\texttt{Quant}^{(i)}(p)(x,w') = 1$ if $\exists \sigma_1 \forall \sigma_2 \exists \sigma_3 ... \forall \sigma_{n'} : p(x, w'_1 ... w'_{i-1} \pi(\sigma_1 ... \sigma_{n'}) w'_{i+1} ... w'_{\ell_0}) = 1$ and that $\texttt{Quant}^{(i)}(p)(x,w') = 0$ otherwise. Now, let $p^{(0)} = \texttt{Quant}^{(i)}(p)$, and for every $j \in [2\ell_0]$ recursively define $p^{(j)} = \texttt{DegRed}^{(j)}(p^{(j-1)})$. By the "moreover" part of Fact 4.7.3.2, and since $(x,w') \in H^{\ell_0} \times H^{\ell_0}$, for every $j \in [2\ell_0]$ we have that $p^{(j)}(x,w') = p^{(j-1)}(x,w')$, and hence $p'(x,w') = \texttt{Quant}^{(i)}(x,w')$. □

Defining the collection of polynomials: Let us now define the collection of $\bar{\ell}_0 = \ell_0(2\ell_0 + 1) + 1$ polynomials. We first define $P_{n,\ell_0(2\ell_0+1)+1}(x,w) = P^{\texttt{TQBF}^{\texttt{loc}}}(x,w)$. Then, we recursively construct the collection in ℓ_0 blocks such that each block consists of $2\ell_0 + 1$ polynomials. The base case will be block $i = \ell_0$, and we will decrease i down to 1. Loosely speaking, in each block $i \in [\ell_0]$, starting from the last polynomial in the previous block, we first apply a quantification operator to the i^{th} variable of the second input w, and then apply $2\ell_0$ linearization operators, one for each variable in the inputs (x,w). Specifically, for the i^{th} block, we define the first polynomial by $P_{n,i(2\ell_0+1)}(x,w) = \texttt{Quant}^{(i)}(P_{n,i(2\ell_0+1)+1})(x,w)$; and for each $j = 1, ..., 2\ell_0$, we define $P_{n,i(2\ell_0+1)-j}(x,w) = \texttt{DegRed}^{(j)}(P_{n,i(2\ell_0+1)-j+1})(x,w)$.

Note that the claimed Property (3) of the collection holds immediately from our definition. To see that Property (4) also holds, note that the first part (regarding $P_{n,\bar{\ell}_0}$)

holds by Claim 4.7.2; and for the "moreover" part, recall (by the properties of the operators $\mathtt{Quant}^{(i)}$ and $\mathtt{DegRed}^{(i)}$ that were mentioned above) that each polynomial $P_{n,k}$ in the collection can be computed in linear space when given access to the "previous" polynomial $P_{n,k-1}$, and also that we can compute the "first" polynomial $P_{n,\ell_0(2\ell_0+1)+1}$ in linear space (since this polynomial is just $P^{\mathtt{TQBF}^{\mathtt{loc}}}$, and relying on Claim 4.7.2). Using a suitable composition lemma for space-bounded computation (see, e.g., [Gol08, Lem. 5.2]), we can compute any polynomial in the collection in space $O(n \cdot \bar{\ell}_0)$.

We now prove Property (1), which asserts that all the polynomials in the collection are of degree at most $O(n \cdot \ell_0 \cdot 2^{2n'})$. We prove this by induction on the blocks, going from $i = \ell_0$ down to $i = 1$, while maintaining the invariant that the "last" polynomial in the previous block $i+1$ (i.e., the polynomial $P_{n,i(2\ell_0+1)+1}$) is of degree at most $O(n \cdot 2^{n'})$. For the base case $i = \ell_0$ the invariant holds by our definition that $P_{n,\ell_0(2\ell_0+1)+1} = P^{\mathtt{TQBF}^{\mathtt{loc}}}$ and by Claim 4.7.2. Now, for every $i = \ell_0, ..., 1$, note that the first polynomial $P_{n,i(2\ell_0+1)}$ in the block is of degree at most $2^{n'} \cdot \deg(P_{n,i(\ell_0+1)+1}) = O(n \cdot 2^{2n'})$ (i.e., the quantifications operator induces a degree blow-up of $2^{n'}$), and in particular the individual degrees of all variables of $P_{n,i(2\ell_0+1)}$ are upper-bounded by this expression. Then, in the subsequent $2\ell_0$ polynomials in the block, we reduce the individual degrees of the variables (sequentially) until all individual degrees are at most $|H| - 1 < 2^{n'}$ (this relies on Fact 4.7.3.2). Thus, the degree of the last polynomial in the block (i.e., of $P_{n,(i-1)(2\ell_0+1)+1}$) is at most $2\ell_0 \cdot 2^{n'} < n \cdot 2^{n'}$, and the invariant is indeed maintained.

Finally, to see that Property (2) holds, fix any $(x, w) \in H^{\ell_0} \times H^{\ell_0}$. Our goal is to show that $P_{n,1}(x, w) = 1$ if $x \in \mathtt{TQBF}^{\mathtt{loc}}$ and $P_{n,1}(x, w) = 0$ otherwise (regardless of w). To do so, recall that $P_{n,\bar{\ell}_0} = P^{\mathtt{TQBF}^{\mathtt{loc}}}$, and hence for any $w' \in H^{\ell_0}$ it holds that $P_{n,\bar{\ell}_0}(x, w') = 1$ if $(x, w') \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}$ and $P_{n,\bar{\ell}_0}(x, w') = 0$ otherwise. Note that the last polynomial in block $i = \ell_0$ (i.e., the polynomial $P_{n,\ell_0(2\ell_0+1)-2\ell_0}$) is obtained by applying $\mathtt{Quant}^{(\ell_0)}$ to $P_{n,\bar{\ell}_0}$ and then applying $\mathtt{DegRed}^{(j)}$ for each $j = 1, ..., 2\ell_0$. Using Claim 4.7.3.3, for any $w' \in H^{\ell_0}$, when this polynomial is given input (x, w'), it outputs the value 1 if $\exists \sigma_1 \forall \sigma_2 \exists \sigma_3 ... \forall \sigma_{n'} (x, w'_1 ... w'_{\ell_0 - 1} \pi(\sigma_1 ... \sigma_{n'})) \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}$, and outputs 0 otherwise. By repeatedly using Claim 4.7.3.3 for the last polynomial in each block $i = \ell_0 - 1, ..., 1$, we have that $P_{n,1}(x, w) = 1$ if $\exists \sigma_1^{(1)} \forall \sigma_2^{(1)} ... \forall \sigma_{n'}^{(1)} ... \exists \sigma_1^{(\ell_0)} ... \forall \sigma_{n'}^{(\ell_0)} : (x, w') \in \mathtt{R\text{-}TQBF}^{\mathtt{loc}}$, where $w' = (\pi(\sigma_1^{(1)} ... \sigma_{n'}^{(1)}), ..., \pi(\sigma_1^{(\ell_0)} ... \sigma_{n'}^{(\ell_0)}))$; and $P_{n,1}(x, w) = 0$ otherwise. In other words, we have that $P_{n,1}(x, w) = 1$ if $x \in \mathtt{TQBF}^{\mathtt{loc}}$ and $P_{n,1}(x, w) = 0$ otherwise, as we wanted. $\square$

Combining the polynomials into a Boolean function. Intuitively, the polynomials in our collection are already downward self-reducible (where "downward" here means that $P_{n,i}$ is reducible to $P_{n,i+1}$) and sample-aided worst-case to average-case reducible (since the polynomials have low degree, and relying on Proposition A.1). Our goal now is simply to "combine" these polynomials into a single Boolean function $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ that will be δ-well-structured.

For every $n \in \mathbb{N}$, we define a corresponding interval of input lengths $I_n = [N, N + \bar{\ell}_0 - 1]$, where $N = 10n' \cdot \ell_0 + 11n \cdot \bar{\ell}_0 = O(n \cdot \bar{\ell}_0)$. Then, for every $i \in \{0, ..., \bar{\ell}_0 - 1\}$, we define $f^{\mathtt{ws}}$ on input length $N + i$ such that it computes (a Boolean version of) $P_{n,\bar{\ell}_0 - i}$. Specifically, $f^{\mathtt{ws}} : \{0,1\}^{N+i} \to \{0,1\}^{N+i}$ considers only the first $10n' \cdot \ell_0 = 2\ell_0 \cdot \log(|\mathbb{F}|) = O(n)$ bits of its input, maps these bits to $(x, w) \in \mathbb{F}^{2\ell_0}$ using π, computes $P_{n,\bar{\ell}_0 - i}(x, w)$, and outputs the bit-representation of $P_{n,\bar{\ell}_0 - i}(x, w)$ (using π^{-1}), padded to the appropriate length $N + i$. On input lengths that do not belong to any interval I_n for $n \in \mathbb{N}$, we define $f^{\mathtt{ws}}$ in some fixed trivial way (e.g., as the identity function).

A straightforward calculation shows that the intervals $\{I_n\}_{n\in\mathbb{N}}$ are disjoint, and thus $f^{\mathtt{ws}}$ is well-defined.[24] In addition, since the input length to $f^{\mathtt{ws}}$ is $N = O(n \cdot \bar{\ell}_0)$ and each polynomial in the collection is computable in space $O(n \cdot \bar{\ell}_0)$, it follows that $f^{\mathtt{ws}}$ is computable in linear space. To see that $\mathtt{TQBF}$ reduces to $f^{\mathtt{ws}}$, recall that by Claim 4.7.1 we can reduce $\mathtt{TQBF}$ to $\mathtt{TQBF}^{\mathtt{loc}}$ in time $n \cdot (\log n)^r$ (for some universal constant $r \in \mathbb{N}$); and note that we can then further reduce $\mathtt{TQBF}^{\mathtt{loc}}$ to $f^{\mathtt{ws}}$ by mapping any $x \in \{0,1\}^n$ to an $(N + \bar{\ell}_0 - 1)$-bit input of the form (x, w, p), where w is an arbitrary string and p is padding. (This is since $f^{\mathtt{ws}}$ on inputs of length $N + \bar{\ell}_0 - 1$ essentially computes $P_{n,1}$.) This reduction is computable in deterministic time $n \cdot \log(n)^{r+2c+1}$.

We now want to show that $f^{\mathtt{ws}}$ is downward self-reducible in time $\mathrm{poly}(1/\delta)$ and in $O((\log N)^{2c})$ steps, where $\delta(N) = 2^{N/(\log N)^{3c}}$ and N denotes the input length. To see this, first note that given input length $N \in \mathbb{N}$ we can find in polynomial time an input length n such that $N \in I_n$, if such n exists. If such n does not exist, then the function is defined trivially on input length n and can be computed in polynomial time. Otherwise, let $N_0 \leq N$ be the smallest input length in I_n (i.e., $N_0 = 10\lceil n/\ell_0(n)\rceil \cdot \ell_0(n) + 11n \cdot \bar{\ell}_0(n)$), and denote $N = N_0 + i$, for some $i \in \{0, ..., \bar{\ell}_0(n) - 1\}$. Note that $f^{\mathtt{ws}}_N$ corresponds to the polynomial $P_{n,\bar{\ell}_0(n)-i}$, and $f^{\mathtt{ws}}_{N-1}$ corresponds to the polynomial $P_{n,\bar{\ell}_0(n)-(i-1)}$. By Claim 4.7.3, the former can be computed in time $\mathrm{poly}(2^{n'}) = \mathrm{poly}(2^{n/(\log n)^c}) = \mathrm{poly}(2^{N/(\log N)^{3c}})$ with oracle access to the latter. Lastly, recall that $|I_n| = \bar{\ell}_0(n) < O(\log N)^{2c}$ and that $f^{\mathtt{ws}}_{N_0}$ corresponds to $P_{n,\ell_0(n)}$, which can be computed in time $\mathrm{poly}(2^{n'})$; hence, there exists an input length $N_0 \geq N - O((\log N)^{2c})$ such that $f^{\mathtt{ws}}_{N_0}$ can be computed in time $\mathrm{poly}(2^{n'}) < \mathrm{poly}(1/\delta(N_0))$.

To see that $f^{\mathtt{ws}}$ is sample-aided worst-case to δ-average-case reducible, first note that computing $f^{\mathtt{ws}}$ on any input length N on which it is not trivially defined is equivalent (up to a polynomial factor in the runtime) to computing a polynomial $\mathbb{F}^{2\ell_0(n)} \to \mathbb{F}$ of degree $d = O(\mathrm{poly}(n) \cdot 2^{2n'})$ in a field of size $q = |\mathbb{F}| = 2^{5n'}$, where $n < N/(\log N)^{2c}$ and $n' = \lceil n/\ell_0(n)\rceil$. [25] We use Proposition A.1 with parameter

[24] This is the case since the largest input length in I_n is $10\lceil n/\ell_0(n)\rceil \cdot \ell_0(n) + 11n \cdot \bar{\ell}_0(n) + (\bar{\ell}_0(n) - 1) < 10n + 10\ell_0(n) + (11n + 1) \cdot \bar{\ell}_0(n) - 1 < 10n + 11(n + 1) \cdot \bar{\ell}_0(n) - 1$, whereas the smallest input length in I_{n+1} is $10\lceil (n+1)/\ell_0(n+1)\rceil \cdot \ell_0(n+1) + 11(n+1) \cdot \bar{\ell}_0(n+1) \geq 10n + 11(n+1)\bar{\ell}_0(n+1) + 10$.

[25] The only potential issue here is that the Boolean function is actually a "padded" version of the function that corresponds to polynomial: It is not immediate that if there exists an algorithm that computes the Boolean function correctly on $\epsilon > 0$ of the n-bit inputs, then there exists an algorithm that computes the polynomial correctly on the same fraction $\epsilon > 0$ of the $m = \log(|\mathbb{F}^{2\ell_0}|)$-bit inputs. However, the latter

$\rho(\log(|\mathbb{F}^{2\ell_0(n)}|)) = \delta_0(n) < \delta(N)$, and note that its hypothesis $\delta_0(n) \geq 10 \cdot \sqrt{d/|\mathbb{F}|}$ is satisfied since we chose $|\mathbb{F}| = \mathrm{poly}(1/\delta_0(n))$ to be sufficiently large. ■

4.2 PRGs for uniform circuits with almost-exponential stretch

Let $\delta(n) = 2^{-n/\mathrm{polylog}(n)}$. The following proposition asserts that if there exists a function that is both δ-well-structured and "hard" for probabilistic algorithms that run in time $2^{n/\mathrm{polylog}(n)}$, then there exists an i.o.-PRG for uniform circuits with almost-exponential stretch. That is:

Proposition 4.8 (almost-exponential hardness of a well-structured function ⇒ PRG for uniform circuits with almost-exponential stretch)**.** *Assume that for some constant $c \in \mathbb{N}$ and for $\delta(n) = 2^{-n/\log(n)^{c+1}}$ there exists a δ-well-structured function that can be computed in linear space but cannot be computed by probabilistic algorithms that run in time $2^{n/\log(n)^c}$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\mathrm{loglog}(n)^k}$ there exists a $(1/t)$-i.o.-PRG for $(t, \log(t))$-uniform circuits that has seed length $\tilde{O}(\log(n))$ and is computable in time $n^{\mathrm{polyloglog}(n)}$.*

Proposition 4.8 follows as an immediate corollary of the following lemma. Loosely speaking, the lemma asserts that for any δ-well-structured function $f^{\mathtt{ws}}$, there exists a corresponding PRG with almost-exponential stretch such that a uniform algorithm that distinguishes the output of the PRG from uniform yields a uniform probabilistic algorithm that computes $f^{\mathtt{ws}}$. Moreover, the lemma provides a "point-wise" statement: For any $n \in \mathbb{N}$, a distinguisher on a small number (i.e., $\mathrm{polyloglog}(n)$) of input lengths in a small interval around n yields a uniform algorithm for $f^{\mathtt{ws}}$ on input length $\tilde{O}(\log(n)$. We will later use this "point-wise" property of the lemma to extend Proposition 4.8 to "almost everywhere" versions (see Propositions 4.11 and 4.12).

In the following statement we consider three algorithms: The pseudorandom generator G; a potential distinguisher for the PRG, denoted A; and an algorithm F for the "hard" function $f^{\mathtt{ws}}$. Loosely speaking, the lemma asserts that for any $n \in \mathbb{N}$, if G is *not* pseudorandom for A on a every input length in a small set of input lengths surrounding n, then F computes $f^{\mathtt{ws}}$ on input length $\ell(n) = \tilde{O}(\log(n))$. We will first fix a constant c that determines the target running time of F (i.e., running time $t_F(\ell) = 2^{\ell/\log(\ell)^c}$), and the other parameters (e.g., the parameters of the well-structured function, and the seed length of the PRG) will depend on c. Specifically:

Lemma 4.9 (distinguishing a PRG based on $f^{\mathtt{ws}}$ ⇒ computing $f^{\mathtt{ws}}$)**.** *Let $c \in \mathbb{N}$ be an arbitrary constant, let $\delta(n) = 2^{-n/\log(n)^{c+1}}$, and let $s : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function such that $s(n) \leq n/2$ for all $n \in \mathbb{N}$. Let $f^{\mathtt{ws}} : \{0,1\}^* \to \{0,1\}^*$ be a (δ, s)-well-structured function that is computable in linear space, let $t(n) = n^{\mathrm{loglog}(n)^k}$ for some constant $k \in \mathbb{N}$, and let $\ell(n) = \lceil \log(n) \cdot (\mathrm{loglog} n)^b \rceil$ for a sufficiently large constant $b \in \mathbb{N}$. Then, there exist two objects that satisfy the property detailed below:*

1. *(Pseudorandom generator). An algorithm G_0 that gets as input 1^n and a random seed of length $\ell_G(n) = \tilde{O}(\ell(n))$, runs in time $n^{\mathrm{polyloglog}(n)}$, and outputs a string of length n.*

assertion holds in our case since we are interested in *probabilistic* algorithms.

2. *(Mapping of any input length to a small set of surrounding input lengths). A polynomial-time computable mapping of any unary string 1^n to a set $S_n \subset [n, n^2]$ of size $|S_n| = s(\tilde{O}(\log(n)))$, where $a \in \mathbb{N}$ is a sufficiently large constant that depends on k.*

The property that the foregoing objects satisfy is the following. For every probabilistic time-t algorithm A that uses $\log(t)$ bits of non-uniform advice there exists a corresponding probabilistic algorithm F that runs in time $t_F(\ell) = 2^{\ell/\log(\ell)^c}$ such that for any $n \in \mathbb{N}$ we have that: If for every $m \in S_n$ it holds that $G_0(1^m, \mathbf{u}_{\ell_{G_0}(m)})$ is not *$(1/t(m))$-pseudorandom for A, then F computes $f^{\mathtt{ws}}$ on strings of length $\ell(n)$.*

Moreover, for any function $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ such that $\mathtt{str}(n) \le n$, the above property holds if we replace G_0 by the algorithm G that computes G_0 and truncates the output to length $\mathtt{str}(n)$ (i.e., $G(1^n, z) = G_0(1^n, z)_1, ..., G_0(1^n, z)_{\mathtt{str}(n)}$).

Observe that Proposition 4.8 indeed follows as a contra-positive of Lemma 4.9 (with $\mathtt{str}$ being the identity function, which means that $G = G_0$): If every probabilistic algorithm F that gets an ℓ-bit input and runs in time $2^{\ell/\log(\ell)^c}$ fails to compute $f^{\mathtt{ws}}$ infinitely-often, then for every corresponding time-t algorithm A there exists an infinite set of inputs on which G is pseudorandom for A.

Proof of Lemma 4.9. For any p, s, δ, k, t, and $f^{\mathtt{ws}}$ that satisfy our hypothesis, let $f^{\mathtt{GL(ws)}} : \{0,1\}^* \to \{0,1\}$ be defined as follows: For any $(x, r) \in \{0,1\}^n \times \{0,1\}^n$ we let $f^{\mathtt{GL(ws)}}(x, r) = \sum_{i \in [n]} f^{\mathtt{ws}}(x)_i \cdot r_i$, where the arithmetic is over $\mathbb{F}_2$.[26] (We use the notation $f^{\mathtt{GL(ws)}}$ since we will use the algorithm of Goldreich and Levin [GL89] to transform a circuit that agrees with $f^{\mathtt{GL(ws)}}$ on $1/2 + \epsilon$ of the inputs into a circuit that computes $f^{\mathtt{ws}}$ on $\mathrm{poly}(\epsilon)$ of the inputs.)

The algorithm G_0 is the Nisan-Wigderson generator, instantiated with $f^{\mathtt{GL(ws)}}$ as the hard function and with combinatorial designs such that the output length is n, the sets in the design are of size $\ell(n) = \lceil \log(n) \cdot (\log\log n)^b \rceil$ (where b is a sufficiently large constant that depends on k), the seed length is $\ell_G(n) = \tilde{O}(\ell(n)) = \tilde{O}(\log(n))$, and the size of the intersection between any two sets in the design is $\gamma \cdot \log(n)$ where $\gamma > 0$ is a sufficiently small constant (see, e.g., [Vad12, Prob 3.2] for a suitable construction). Since $f^{\mathtt{ws}}$ is computable in linear space, the function $f^{\mathtt{GL(ws)}}(x, r)$ is computable in time $n^{\mathrm{polyloglog}(n)}$, and hence G_0 is computable in time $n^{\mathrm{polyloglog}(n)}$.

Fix a mapping of any 1^n to a corresponding set S_n that will be defined in a moment (and depends only on the parameters up to this point). Now, let $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ be any polynomial-time computable function satisfying $\mathtt{str}(n) \le n$, and let G be such that $G(1^n, s) = G_0(1^n, s)_{1,...,\mathtt{str}(n)}$. For $t(n) = n^{\log\log(n)^k}$, let A be a probabilistic algorithm that gets input 1^n and $\log(t(n))$ bits of non-uniform advice and runs in time $t(n)$. For any sufficiently large $n \in \mathbb{N}$, we assume that for every $m \in S_n$, when A is given input $1^{\mathtt{str}(m)}$ and corresponding "good" advice, with probability at least $1/t(m)$ it outputs a circuit $D_{\mathtt{str}(m)} : \{0,1\}^{\mathtt{str}(m)} \to \{0,1\}$ that $(1/t(m))$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform. Under this assumption, we will construct a probabilistic

[26]On odd input lengths the function $f^{\mathtt{GL(ws)}}$ is defined by ignoring the last input bit; that is, $f^{\mathtt{GL(ws)}}(x, r\sigma) = f^{\mathtt{GL(ws)}}(x, r)$, where $|x| = |r|$ and $|\sigma| = 1$.

algorithm that gets input $1^{\ell(n)}$, runs in time $\text{poly}(1/\delta(\ell(n)) = 2^{O(\ell(n)/\log(\ell(n))^{c+1})}$, and with high probability outputs a circuit $\{0,1\}^{\ell(n)} \to \{0,1\}$ that correctly computes $f^{\texttt{ws}}$ on $\ell(n)$-bit inputs. This implies that a probabilistic algorithm can decide $f^{\texttt{ws}}$ on $\{0,1\}^{\ell(n)}$ in time at most $2^{\ell(n)/\log(\ell(n))^c}$.

Towards presenting the construction, denote $\ell'(n) = \ell(n)/\log(\ell(n))^{c+1}$, and fix a sufficiently small universal constant $\epsilon > 0$ (which depends only on universal constants from arguments in [NW94; IW98]). We assume that $\ell(n)$ is sufficiently large such that $t(n) = n^{\text{loglog}(n)^k} \le 2^{\epsilon \cdot \ell'(n)}$. Recall that, since $f^{\texttt{ws}}$ is downward self-reducible in s steps, there exists an input length $\ell_0(n) \ge \ell(n) - s(\ell(n))$ such that $f^{\texttt{ws}}_{\ell_0(n)}$ is computable in time $\text{poly}(1/\delta(\ell_0(n)))$. For $L_n = \{\ell_0(n), ..., \ell(n)\}$, we define $S_n = \{\ell^{-1}(2i) : i \in L_n\}$. Note that indeed $|S_n| \le s(\ell(n)) = s(\tilde{O}(\log(n)))$; and relying on the fact that $s(\ell(n)) \le \ell(n)/2$, we have that $S_n \subset [n_0, n_1]$ where $n_0 = \ell^{-1}(2\ell_0) \ge \ell^{-1}(\ell(n)) = n$ and $n_1 = \ell^{-1}(2\ell(n)) < n^2$. Lastly, note that S_n does not depend on the function $\texttt{str}$ or on the algorithm A.

Our first step is to show that (loosely speaking) under our assumption about A, for any $m \in S_n$ we can efficiently construct (using only a small amount of non-uniform advice) a circuit that computes $f^{\texttt{GL(ws)}}$ on noticeably more than half of the inputs of length $\ell(m)$. The proof of this claim is a variation on the standard efficient transformation of distinguishers for the Nisan-Wigderson PRG to approximating circuits for the "hard" function, from [IW98] (following [NW94]).

Claim 4.9.1. *There exists a probabilistic algorithm such that for any $m \in S_n$, when the algorithm is given input $1^{\ell(m)}$, and oracle access to $f^{\texttt{GL(ws)}}$ on $\ell(m)$-bit inputs, and $2\epsilon \cdot \ell'(m)$ bits of non-uniform advice, the algorithm runs in time $2^{\ell'(m)}$ and with probability more than $2^{-\ell'(m)}$ outputs a circuit $\{0,1\}^{\ell(m)} \to \{0,1\}$ that computes $f^{\texttt{GL(ws)}}$ correctly on more than $1/2 + 2^{-\ell'(m)}$ of the inputs.*

Proof. Let $\ell = \ell(m)$, let $\ell' = \ell'(m)$, and let $m' = \texttt{str}(m) \le m$. Let us first assume that $m' = m$ (i.e., $G_0 = G$ and $\texttt{str}$ is the identity function). In this case, a standard argument (based on [NW94] and first noted in [IW98]) shows that there exists a probabilistic polynomial time algorithm A_{NW} that satisfies the following: When given as input a circuit $D_m : \{0,1\}^m \to \{0,1\}$ that $(1/m^{\text{loglog}(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and also given oracle access to $f^{\texttt{GL(ws)}}$ on ℓ-bit inputs, with probability at least $1/O(m)$ the algorithm A_{NW} outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{x \in \{0,1\}^\ell}[C_\ell(x) = f^{\texttt{GL(ws)}}(x)] \ge 1/2 + 1/O(m^{\text{loglog}(m)^k})$.

Towards extending this claim to the setting of $\texttt{str}(m) < m$, let us quickly recap the original construction of A_{NW}: The algorithm randomly chooses an index $i \in [m]$ (for a hybrid argument) and values for all the bits in the seed of the NW generator outside the i^{th} set (in the underlying design); then uses its oracle to query $\text{poly}(m)$ values for $f^{\texttt{GL(ws)}}$ (these are potential values for the output indices whose sets in the seed intersect with the i^{th} set), and "hard-wires" them into a circuit C_ℓ that gets input $x \in \{0,1\}^\ell$, simulates the corresponding m-bit output of the PRG, and uses the distinguisher to decide if $x \in f^{\texttt{GL(ws)}}$. Now, note that if the output of the PRG is truncated to length

$m' = \mathtt{str}(m) < m$, the construction above works essentially the same if we choose an initial index $i \in [m']$ instead of $i \in [m]$, and if C_ℓ completes x to an m'-bit output of the PRG instead of an m-bit output. Indeed, referring to the underlying analysis, these changes only improve the guarantee on the algorithm's probability of success (we do not use the fact that the guarantee is better). Thus, for any $m' = \mathtt{str}(m) \leq m$, there is an algorithm A_{NW} that gets as input a circuit $D_{m'} : \{0,1\}^{m'} \to \{0,1\}$ that $(1/m^{\text{loglog}(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and oracle access to $f_\ell^{\mathtt{GL(ws)}}$, and with probability at least $1/O(m)$ outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{x \in \{0,1\}^\ell}[C_\ell(x) = f^{\mathtt{GL(ws)}}(x)] \geq 1/2 + 1/O(m^{\text{loglog}(m)^k})$.

Now, for $\ell \in \mathbb{N}$, let $m = m(\ell)$ be such that ℓ is the seed length of G on m-bit inputs, and let $m' = \mathtt{str}(m)$. Our probabilistic algorithm is given as input 1^ℓ and non-uniform advice (a, m') such that $|a| = \log(t(m)) = \log(m) \cdot \text{loglog}(m)^k = \epsilon \cdot \ell'$; note that, since $m' \leq m$, the total length of the advice is at most $\epsilon \cdot \ell' + \log(m) < 2\epsilon \cdot \ell'$. Our probabilistic algorithm simulates the algorithm A on input $1^{m'}$ with the advice a, and feeds the output of A as input for A_{NW}. This algorithm runs in time $m^{O(\text{loglog}(m)^k)} = 2^{\ell'}$. Note that with probability more than $(1/m^{\text{loglog}(m)^k})$, the algorithm A outputs $D_{m'} : \{0,1\}^{m'} \to \{0,1\}$ that $(1/m^{\text{loglog}(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and conditioned on this event, with probability at least $1/O(m)$ the combined algorithm outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ that correctly computes $f^{\mathtt{GL(ws)}}$ on $1/2 + 1/O(m^{\text{loglog}(m)^k}) > 1/2 + 2^{-\ell'}$ of the ℓ-bit inputs. □

We will call the algorithm in the statement of Claim 4.9.1 a weak learner for $f^{\mathtt{GL(ws)}}$ on input length $\ell(m)$. Then, Claim 4.9.1 implies that there exists a weak learner for $f^{\mathtt{GL(ws)}}$ on any input length in $2L_n = \{2i : i \in L_n\}$. See Figure 1 for a pictorial description of the sets L_n, $2L_n$, and S_n, and for a reminder about our assumptions at this point.

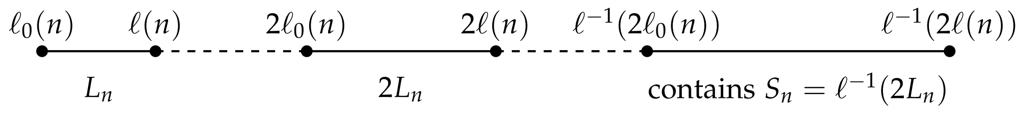

Figure 1: We want to compute $f^{\mathtt{ws}}$ on inputs of length $\ell(n)$. We define a corresponding interval $L_n = \{\ell_0(n), ..., \ell(n)\}$ of input lengths, where $\ell_0(n) \geq \ell(n) - s(\ell(n))$, in which we will use the downward self-reducibility of $f^{\mathtt{ws}}$. We assume that there is a uniform distinguisher A for the PRG on all input lengths in $S_n = \ell^{-1}(2L_n)$, and deduced that there exists a weak learner for $f^{\mathtt{GL(ws)}}$ on all input lengths in $2L_n$.

Given as input $1^{\ell(n)}$, we construct in time $\text{poly}(1/\delta(\ell(n))) = 2^{O(\ell(n)/\log(\ell(n))^{c+1})} = 2^{O(\ell'(n))}$ a circuit for $f_{\ell(n)}^{\mathtt{ws}}$, by inductively constructing circuits for $f_i^{\mathtt{ws}}$, for increasing values of $i \in L_n = \{\ell_0(n), ..., \ell(n)\}$, where for each i we will construct the corresponding circuit in time $2^{O(i/\log(i)^{c+!})}$. Indeed, the construction for the base case $i = \ell_0(n)$ is trivial, since $f_{\ell_0(n)}^{\mathtt{ws}}$ is computable in time $\text{poly}(1/\delta(\ell_0(n))) \leq 2^{O(\ell_0(n)/\log(\ell_0(n))^{c+1})}$, where the inequality is due to our hypothesis that δ is sufficiently large (the precise requirement from δ will be specified below). Therefore we just need to prove the inductive step. This will be done as follows:

Claim 4.9.2. *There exists an algorithm that gets as input $i \in L_n \setminus \{\ell_0(n)\}$ and a circuit $C_{i-1} : \{0,1\}^{i-1} \to \{0,1\}$ that computes $f^{\mathtt{ws}}_{i-1}$, runs in time $2^{O(i/\log(i)^{c+1})} \cdot \mathrm{poly}(|C_{i-1}|)$, and with probability at least $1 - \exp(i/\log(i)^{c+1})$ outputs a circuit $C_i : \{0,1\}^i \to \{0,1\}$ of size $2^{O(i/\log(i)^{c+1})}$ that computes $f^{\mathtt{ws}}_i$. (Note that the size of the output circuit C_i does not depend on the size of the input circuit C_{i-1}.)*

Proof. Let $i' = 2i/\log(2i)^{c+1}$, and let $S = |C_{i-1}|$. First note that the algorithm can compute $f^{\mathtt{ws}}_i$ in time $\mathrm{poly}(1/\delta(i), S)$ (using the downward self-reducibility of $f^{\mathtt{ws}}$ and the circuit C_{i-1}) and also compute $f^{\mathtt{GL(ws)}}_{2i}$ in time $\mathrm{poly}(1/\delta(i), S)$ (using the fact that $f^{\mathtt{GL(ws)}}(x,r) = \sum_{j\in[i]} f^{\mathtt{ws}}_i(x)_j \cdot r_j$). We will construct C_i in four steps:

1. Simulating the learner for $f^{\mathtt{GL(ws)}}_{2i}$. We use the weak learner for $f^{\mathtt{GL(ws)}}_{2i}$ to construct a list of $2^{O(i')}$ circuits $\{0,1\}^{2i} \to \{0,1\}$ of size $2^{i'}$ such that at least one circuit in the list correctly decides $f^{\mathtt{GL(ws)}}_{2i}$ on $1/2 + 2^{-i'}$ of the $(2i)$-bit inputs.

To do so, we enumerate over all $2^{2\epsilon\cdot i'}$ possible advice strings for the weak learner for $f^{\mathtt{GL(ws)}}_{2i}$. For each fixed advice string $a \in \{0,1\}^{2\epsilon\cdot i'}$, we simulate the weak learner with advice a for $2^{O(i')}$ times (using independent randomness in each simulation), while answering its queries to $f^{\mathtt{GL(ws)}}_{2i}$ using C_{i-1}. Note that when a is the "good" advice, each simulation of the learner is successful with probability at least $2^{-i'}$. Thus, with probability at least $1 - \exp(-i')$ our list contains at least one circuit that correctly computes $f^{\mathtt{GL(ws)}}_{2i}$ on at least $1/2 + 2^{-i'}$ of its inputs.

2. Weeding the list to find a circuit for $f^{\mathtt{GL(ws)}}_{2i}$. We now test each of the $2^{O(i')}$ circuits in order to find a single circuit $C'_i : \{0,1\}^{2i} \to \{0,1\}$ that computes $f^{\mathtt{GL(ws)}}_{2i}$ on $1/2 + 2^{-2i'}$ of the inputs.

To test each circuit we randomly sample $2^{O(i')}$ inputs, compute $f^{\mathtt{GL(ws)}}_{2i}$ at each of these inputs using C_{i-1}, and compare the value of $f^{\mathtt{GL(ws)}}_{2i}$ to the output of the candidate circuit. For each circuit, with probability at least $1 - 2^{-O(i')}$ over the sampled inputs, we correctly estimate its agreement with $f^{\mathtt{GL(ws)}}_{2i}$ up to error $2^{-2i'-1}$. Union-bounding over the $2^{O(i')}$ circuits, with probability at least $1 - 2^{-O(i')}$, the circuit that we find in this step has agreement at least $1/2 + 2^{-2i'}$ with $f^{\mathtt{ws}}$.

3. Conversion to a circuit that computes $f^{\mathtt{ws}}_i$ on average. We now convert the circuit C'_i for $f^{\mathtt{GL(ws)}}_{2i}$ to a circuit $\{0,1\}^i \to \{0,1\}^i$ of size $2^{O(i')}$ that computes $f^{\mathtt{ws}}_i$ correctly on $\delta(i) = 2^{-O(i')}$ of its i-bit inputs.[27]

[27] Recall that in our hypothesis we required a δ-well-structured function where $\delta(n) = 2^{-n/\mathrm{polylog}(n)}$ for a sufficiently large polylogarithmic function. At this point we can specify our precise requirement,

To do so, we first use the algorithm of Goldreich and Levin [GL89] to convert the deterministic circuit C'_i into a probabilistic circuit C''_i of size $2^{O(i')}$ such that $\Pr[C''_i(x) = f^{\mathtt{ws}}_i(x)] \geq 2^{-O(i')}$, where the probability is taken both over a random choice of $x \in \{0,1\}^i$ and over the internal randomness of C''_i. Specifically, the circuit $C''_i : \{0,1\}^i \to \{0,1\}$ gets input $x \in \{0,1\}^i$, and simulates the algorithm from [Gol08, Thm 7.8] with parameter $\delta_0 = 2^{-2i'}$, while resolving the oracle queries of the algorithm using the circuit C'_i; then, the circuit C''_i outputs a random element from the list that is produced by the algorithm from [Gol08]. Since $\mathbb{E}_x[\Pr_r[C'_i(x,r) = f^{\mathtt{GL(ws)}}_{2i}(x,r)]] \geq 1/2 + \delta_0$, it follows that for at least $\delta_0/2$ of the inputs $x \in \{0,1\}^i$ it holds that $\Pr_r[C'_i(x,r) = f^{\mathtt{GL(ws)}}_{2i}(x,r)] \geq 1/2 + \delta_0/2$. For each such input, with probability at least $1/2$ the algorithm of [GL89] outputs a list of size $\mathrm{poly}(1/\delta_0)$ that contains $f^{\mathtt{ws}}(x)$, and thus the circuit C''_i outputs $f^{\mathtt{ws}}(x)$ with probability $\mathrm{poly}(\delta_0)$.

To conclude we now choose randomness for C'_i and "hard-wire" it into the circuit. With probability at least $1 - \exp(i')$, we obtain a circuit C'''_i of size $2^{O(i')}$ that computes $f^{\mathtt{ws}}_i$ correctly on $\delta = \mathrm{poly}(\delta_0)$ of the inputs.

4. Worst-case to δ-average-case reduction for $f^{\mathtt{ws}}_i$. Our final step is to convert C''_i (which computes $f^{\mathtt{ws}}_i$ correctly on $\delta(i)$ of the i-bit inputs) into a circuit C_i of size $2^{O(i')}$ that correctly computes $f^{\mathtt{ws}}_i$ on all inputs.

To do so we will use the fact that $f^{\mathtt{ws}}$ is sample-aided worst-case to δ-average-case reducible, and the fact that we can generate random labeled samples $(r, f^{\mathtt{ws}}_i(r))$ by using the circuit C_{i-1} to compute $f^{\mathtt{ws}}_i(r)$. With probability at least $1 - \delta(i)$, the uniform reduction outputs a probabilistic circuit C'''_i of size $2^{O(i')}$ such that for every $x \in \{0,1\}^i$ it holds that $\Pr_r[C'''_i(x,r) = f^{\mathtt{ws}}(x)] \geq 2/3$. [28] Using naive error-reduction we obtain a circuit of size $2^{O(i')}$ that correctly computes $f^{\mathtt{ws}}$ at any input with probability $1 - 2^{-O(i)}$. Then we uniformly choose randomness of this circuit and "hard-wire" the randomness into it, such that with probability at least $1 - 2^{-i}$ we obtain a deterministic circuit $C_i : \{0,1\}^i \to \{0,1\}$ that computes $f^{\mathtt{ws}}_i$. $\square$

Repeating the algorithm from Claim 4.9.2 for $i = \ell_0(n) + 1, ..., \ell(n)$, we obtain an algorithm that runs in time $2^{O(\ell')}$, and outputs a circuit for $f^{\mathtt{ws}}_{\ell(n)}$ with probability at least $1 - \sum_{i=\ell'}^{\ell} \exp(i/\log(i)^{c+1}) \geq 2/3$, assuming that ℓ is sufficiently large. $\blacksquare$

In the last part of the proof of Lemma 4.9, after we converted a distinguisher for $f^{\mathtt{GL(ws)}}$ into a weak learner for $f^{\mathtt{GL(ws)}}$ (i.e., after Claim 4.9.1), we used the existence of the weak learner for $f^{\mathtt{GL(ws)}}$ on $2L_n$ to obtain a circuit that computes $f^{\mathtt{ws}}$ on L_n. This part of the proof immediately implies the following, weaker corollary. (The corollary

which is that $\delta(n) = 2^{-O(n/\log(n)^{c+1})}$, where the universal constant hidden inside the O-notation depends only on universal constants from [GL89] as explained in the argument that we now present.

[28] In Definition 4.3 the output circuit has oracle gates to a function that agrees with the target function on a δ fraction of the inputs. Indeed, we replace these oracle gates with copies of the circuit C''_i.

is weaker since it does not have any "point-wise" property, i.e. does not convert a learner on specific input lengths to a circuit for $f^{\texttt{ws}}$ on a corresponding input length.)

Corollary 4.10 (learning $f^{\texttt{GL(ws)}} \implies$ computing $f^{\texttt{ws}}$)**.** *Let $c \in \mathbb{N}$ be an arbitrary constant, let $f^{\texttt{ws}} : \{0,1\}^* \to \{0,1\}^*$ be a δ-well-structured function for $\delta(n) = 2^{-n/\log(n)^{c+1}}$, and let $f^{\texttt{GL(ws)}}$ be defined as in the proof of Lemma 4.9. Assume that for every $\ell \in \mathbb{N}$ there exists a* weak learner *for $f^{\texttt{GL(ws)}}$; that is, an algorithm that gets input 1^ℓ and oracle access to $f_\ell^{\texttt{GL(ws)}}$, runs in time $\delta^{-1}(\ell)$, and with probability more than $\delta(\ell)$ outputs a circuit over ℓ bits that computes $f^{\texttt{GL(ws)}}$ correctly on more than $1/2 + \delta(\ell)$ of the inputs. Then, there exists an algorithm that for every ℓ, when given input 1^ℓ, runs in time $2^{\ell/\log(\ell)^c}$ and outputs an ℓ-bit circuit that computes $f^{\texttt{ws}}$.*

We now use the "point-wise" property of Lemma 4.9 to deduce two "almost-always" versions of Proposition 4.8. Recall that in our construction of a well-structured function $f^{\texttt{ws}}$, on some input lengths $f^{\texttt{ws}}$ is defined trivially, and thus it cannot be that $f^{\texttt{ws}}$ is "hard" almost-almost.[29] However, since $\texttt{TQBF}$ can be reduced to $f^{\texttt{ws}}$ with a quasilinear blow-up $b : \mathbb{N} \to \mathbb{N}$, we can still deduce the following: If $\texttt{TQBF}$ is "hard" almost-always, then for every $n \in \mathbb{N}$ there exists $n' \le b(n)$ such that $f^{\texttt{ws}}$ is "hard" on input length n' (i.e., this holds for the smallest $n' \ge n$ of the form $b(n_0)$ for $n_0 \in \mathbb{N}$).

In our first "almost-always" result, the hypothesis is that a well-structured function is "hard" on a dense set of input lengths as above, and the conclusion is that there exists an "almost-everywhere" HSG for uniform circuits.

Proposition 4.11 ("almost everywhere" hardness of $f^{\texttt{ws}} \Rightarrow$ "almost everywhere" derandomization of $\mathcal{RP}$ "on average")**.** *Assume that for some constant $c \in \mathbb{N}$ and for $\delta(n) = 2^{-n/\log(n)^{c+1}}$ there exists a $(\delta, \text{polylog}(n))$-well-structured function and $b(n) = \tilde{O}(n)$ such that for every probabilistic algorithm that runs in time $2^{n/\log(n)^c}$, and every sufficiently large $n \in \mathbb{N}$, the algorithm fails to compute $f^{\texttt{ws}}$ on input length $\overline{n} = \min\{b(n_0) \ge n : n_0 \in \mathbb{N}\}$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\text{loglog}(n)^k}$ there exists a $(1/t)$-HSG for $(t, \log(t))$-uniform circuits that is computable in time $n^{\text{polyloglog}(n)}$ and has seed length $\tilde{O}(\log(n))$.*

Proof. We instantiate Lemma 4.9 with the constant c, the function $f^{\texttt{ws}}$, the parameter $2k$ instead of k (i.e., the parameter t in Lemma 4.9 is $t(n) = n^{\text{loglog}(n)^{2k}}$) and with $\texttt{str}(n) = n$ (i.e., $\texttt{str}$ is the identity function). Let $\ell(n) = \lceil \tilde{O}(\log(n)) \rceil$ be the quasilogarithmic function given by Lemma 4.9, let $G = G_0$ be the corresponding PRG, and let $\ell_G(n) = \tilde{O}(\log(n))$ be the seed length of G. From our hypothesis regarding the hardness of $f^{\texttt{ws}}$, we can deduce the following:

Corollary 4.11.1. *For every $n \in \mathbb{N}$ there is a polynomial-time-enumerable set $\overline{S_n} = S_{n^{\text{polyloglog}(n)}} \subset [n, n^{\text{polyloglog}(n)}]$ of size* polyloglog(n) *such that for every probabilistic algorithm A' that runs in time t^2 and uses $2\log(t)$ bits of advice, if $n \in \mathbb{N}$ is sufficiently large then there exists $m \in \overline{S_n}$ such that $G(1^m, \mathbf{u}_{\ell_G(m)})$ is $(1/t^2(m))$-pseudorandom for A'.*

[29]Moreover, in every small interval of input lengths, there is an input length on which $f^{\texttt{ws}}$ can be solved in time poly$(1/\delta)$ (without using an oracle).

Proof. For every $n \in \mathbb{N}$, let $\bar{\ell}(n) = \min\{b(\ell_0) \geq \ell(n) : \ell_0 \in \mathbb{N}\}$, and let $\bar{n} = \ell^{-1}(\bar{\ell}(n)) \in [n, n^{\mathrm{polyloglog}(n)}]$. We define $\overline{S_n} = S_{\bar{n}}$, where $S_{\bar{n}}$ is the set from Item (2) of Lemma 4.9 that corresponds to $\bar{n}$. Note that $\overline{S_n} \subset [n, n^{\mathrm{polyloglog}(n)}]$ and that $|\overline{S_n}| \leq \mathrm{polyloglog}(n)$.

Now, let A' be a probabilistic algorithm as in our hypothesis, let F' be the corresponding probabilistic algorithm from Lemma 4.9 that runs in time $t_{F'}(i) = 2^{i/\log(i)^c}$, and let $n \in \mathbb{N}$ be sufficiently large. By Lemma 4.9, if there is no $m \in \overline{S_n}$ such that $G(1^m, \mathbf{u}_{\ell_G(m)})$ is $(1/t(m))$-pseudorandom for A', then F' correctly computes $f^{\mathtt{ws}}$ on input length $\bar{\ell}(\bar{n}) = \bar{\ell}(n)$, which contradicts our hypothesis. $\square$

The HSG, denoted H, gets input 1^n, uniformly chooses $m \in \overline{S_n}$, computes $G(1^m, s)$ for a random $s \in \{0,1\}^{\ell_G(m)}$, and outputs the n-bit prefix of $G(1^m, s)$. Note that the seed length that H requires is $\tilde{O}(\log(n^{\mathrm{polyloglog}(n)})) + \log(|\overline{S_n}|) = \tilde{O}(\log(n))$, and that H is computable in time at most $n^{\mathrm{polyloglog}(n)}$.

To prove that H is a $(1/t)$-HSG for $(t, \log(t))$-uniform circuits, let A be a probabilistic algorithm that runs in time t and uses $\log(t)$ bits of advice. Assume towards a contradiction that there exists an infinite set $B_A \subseteq \mathbb{N}$ such that for every $n \in B_A$, with probability more than $1/t(n)$ the algorithm A outputs a circuit $D_n : \{0,1\}^n \to \{0,1\}$ satisfying $\Pr_s[D_n(H(1^n, s)) = 0] = 1$ and $\Pr_{x \in \{0,1\}^n}[D_n(x) = 1] > 1/t(n)$. We will construct an algorithm A' that runs in time less than t^2, uses $\log(t) + \log(n) < 2\log(t)$ bits of advice, and for infinitely-many sets of the form $\overline{S_n}$, for every $m \in \overline{S_n}$ it holds that $G(1^m, \mathbf{u}_{\ell_G(m)})$ is not $(1/t(m))$-pseudorandom for A'. This contradicts Corollary 4.11.1.

The algorithm A' gets input 1^m, and as advice it gets an integer of size at most m. Specifically, if m is in a set $\overline{S_n}$ for some $n \in B_A$, then the advice will be set to n; and otherwise the advice is zero (which signals to A' that it can fail on input length m). For any $m \in \mathbb{N}$ such that the first case holds, we know that $A(1^n)$ outputs, with probability more than $1/t(n)$, a circuit $D_n : \{0,1\}^n \to \{0,1\}$ satisfying both $\Pr_{s \in \{0,1\}^{\tilde{O}(\log(n))}}[D_n(H(1^n, s)) = 0] = 1$ and $\Pr_{x \in \{0,1\}^n}[D_n(x) = 1] > 1/t(n)$. The algorithm A' simulates A on input length n, and outputs a circuit $D_m : \{0,1\}^m \to \{0,1\}$ such that D_m computes D_n on the n-bit prefix of its input. By our hypothesis regarding D_n, when fixing the first part of the seed of H to be the integer m, we have that $\Pr_{s'}[D_n(H(1^n, m \circ s')) = 0] = \Pr_{s'}[D_m(G(1^m, s')) = 0] = 1$, whereas $\Pr_{x \in \{0,1\}^m}[D_m(x) = 1] > 1/t(n)$. It follows that D_m distinguishes the m-bit output of G from uniform with advantage $1/t(n) \geq 1/t(m)$. ■

We also prove another "almost-everywhere" version of Proposition 4.8. Loosely speaking, under the same hypothesis as in Proposition 4.11, we show that $\mathcal{BPP}$ can be derandomized "on average" using only a small (triple-logarithmic) amount of advice. In contrast to the conclusion of Proposition 4.11, in the following proposition we do *not* construct a PRG or HSG, but rather simulate every $\mathcal{BPP}$ algorithm by a corresponding deterministic algorithm that uses a small amount of non-uniform advice.

Proposition 4.12 ("almost everywhere" hardness of $f^{\mathtt{ws}} \Rightarrow$ "almost everywhere" derandomization of $\mathcal{BPP}$ "on average" with short advice)**.** *Assume that for some constant*

$c \in \mathbb{N}$ and for $\delta(n) = 2^{-n/\log(n)^{c+1}}$ there exists a $(\delta, \text{polylog}(n))$-well-structured function and $b(n) = \tilde{O}(n)$ such that for every probabilistic algorithm that runs in time $2^{n/\log(n)^c}$, and every sufficiently large $n \in \mathbb{N}$, the algorithm fails to compute $f^{\mathtt{ws}}$ on input length $\overline{n} = \min\{b(n_0) \geq n : n_0 \in \mathbb{N}\}$.

For $k \in \mathbb{N}$ and $t(n) = n^{\text{loglog}(n)^k}$, let $L \in \mathcal{BPTIME}[t]$ and let F be a probabilistic t-time algorithm. Then, there exists a deterministic machine D that runs in time $n^{\text{polyloglog}(n)}$ and gets $O(\text{logloglog}(n))$ bits of non-uniform advice such that for all sufficiently large $n \in \mathbb{N}$, the probability (over coin tosses of F) that $F(1^n)$ is an input $x \in \{0,1\}^n$ for which $D(x) \neq L(x)$ is at most $1/t(n)$.

Proof. Let us first prove the claim assuming that $L \in \mathcal{BPTIME}[t]$ can be decided using only a number of random coins that equals the input length; later on we show how to remove this assumption (by a padding argument). For t as in our hypothesis for L as above, let M be a probabilistic t-time algorithm that decides L and that for every input $x \in \{0,1\}^*$ uses $|x|$ random coins, and let F be a probabilistic t-time algorithm. Consider the algorithm A that, on input 1^n, simulates F on input 1^n to obtain $x \in \{0,1\}^n$, and outputs a circuit $C_x : \{0,1\}^n \to \{0,1\}$ that computes the decision of M at input x as a function of the random coins of M.

We instantiate Lemma 4.9 with the constant c, the function $f^{\mathtt{ws}}$, and the parameter k. Let $\ell = \tilde{O}(\log(n))$ be the quasilogarithmic function given by the lemma, let G_0 be the PRG, and let $\ell_G = \tilde{O}(\log(n))$ be the seed length of G_0. We first need a claim similar to Corollary 4.11.1, but this time also quantifying over the function $\mathtt{str}$:

Corollary 4.12.1. *For every $n \in \mathbb{N}$ there is a polynomial-time-enumerable set $\overline{S_n} = S_{n^{\text{polyloglog}(n)}} \subset [n, n^{\text{polyloglog}(n)}]$ of size $\text{polyloglog}(n)$ that satisfies the following. For every $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ satisfying $\mathtt{str}(n) \leq n$, let $G_{\mathtt{str}}$ be the algorithm that on input 1^n uses a random seed of length $\tilde{O}(\log(n))$, computes G_0, which outputs an n-bit string, and truncates the output to length $\mathtt{str}(n)$. Then, for every probabilistic algorithm A' that runs in time t and uses $\log(t)$ bits of advice, if $n \in \mathbb{N}$ is sufficiently large then there exists $m \in \overline{S_n}$ such that $G_{\mathtt{str}}(1^m, \mathbf{u}_{\ell_G(m)})$ is $(1/t(m))$-pseudorandom for A'.*

Proof. For any $n \in \mathbb{N}$ we define $\overline{\ell}(n)$ and $\overline{S_n}$ as in the proof of Corollary 4.11.1. For any $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ satisfying $\mathtt{str}(n) \leq n$, let $G_{\mathtt{str}}$ be the corresponding function. Now, let A' be any probabilistic algorithm as in our hypothesis, let F' be the corresponding probabilistic algorithm from Lemma 4.9 that runs in time $t_{F'}(i) = 2^{i/\log(i)^c}$, and let $n \in \mathbb{N}$ be sufficiently large. By Lemma 4.9, if there is no $m \in \overline{S_n}$ such that $G_{\mathtt{str}}(1^m, \mathbf{u}_{\ell_G(m)})$ is $(1/t(m))$-pseudorandom for A', then F' correctly computes $f^{\mathtt{ws}}$ on input length $\overline{\ell}(n)$. This contradicts our hypothesis regarding $f^{\mathtt{ws}}$. □

The machine D gets input $x \in \{0,1\}^n$ and advice of length $O(\text{logloglog}(n))$, which is interpreted as an index of an element m in the set $\overline{S_n}$. Then, for each $s \in \{0,1\}^{\ell_G(m)}$ the algorithm computes the n-bit prefix of $G_0(1^m, s)$, denoted $w_s = G_0(1^m, s)_{1,\ldots,n}$, and outputs the majority value of $\{M(x, w_s) : s \in \{0,1\}^{\ell_G(m)}\}$. Note that the machine D indeed runs in time $m^{\text{polyloglog}(m)} = n^{\text{polyloglog}(n)}$.

Our goal now is to prove that for every sufficiently large $n \in \mathbb{N}$ there exists advice $m \in \overline{S_n}$ such that with probability at least $1 - 1/t(n)$ over the coin tosses of F (which determine $x \in \{0,1\}^n$ and $C_x \colon \{0,1\}^n \to \{0,1\}$) it holds that

$$\left| \Pr_{r \in \{0,1\}^n}[C_x(r) = 1] - \Pr_s[C_x(G_0(1^m, s)_{1,\ldots,n}) = 1] \right| < 1/t(n)\,, \tag{4.2}$$

which is equivalent (for a fixed $x \in \{0,1\}^n$) to the following statement:

$$\left| \Pr_{r \in \{0,1\}^n}[M(x,r) = 1] - \Pr_s[M(x, w_s) = 1] \right| < 1/t(n)\,. \tag{4.3}$$

Indeed, proving this would suffice to prove our claim, since for every $x \in \{0,1\}^n$ such that Eq. (4.3) holds we have that $D(x) = L(x)$.

To prove the claim above, assume towards a contradiction that there exists an infinite set of input lengths $B_A \subseteq \mathbb{N}$ such that for every $n \in B_A$ and every advice $m \in \overline{S_n}$, with probability more than $1/t(n)$ over $x \leftarrow F(1^n)$ it holds that $C_x : \{0,1\}^n \to \{0,1\}$ violates Eq. (4.2). Let $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ be defined by $\mathtt{str}(m) = n$ if $m \in \overline{S_n}$ for some $n \in B_A$, and $\mathtt{str}(m) = m$ otherwise.[30] Then, our assumption implies that for infinitely-many input lengths $n \in B_A$, for every $m \in \overline{S_n}$ it holds that $G_{\mathtt{str}}(1^m, \mathbf{u}_{\ell_G(m)})$ is not $(1/t(n))$-pseudorandom for A. This contradicts Corollary 4.12.1.

Finally, let us remove the assumption that L can be decided using a linear number of coins, by a padding argument. For any $L \in \mathcal{BPTIME}[t]$, consider a padded version $L^{\mathtt{pad}} = \{(x, 1^{t(|x|)}) : x \in L\}$, and note that $L^{\mathtt{pad}}$ can be decided in linear time using $|z|$ coins on any input z. By the argument above, for every probabilistic t-time algorithm $F^{\mathtt{pad}}$ there exists an algorithm $D^{\mathtt{pad}}$ that runs in time $t_{D^{\mathtt{pad}}}(m) = m^{\mathrm{polyloglog}(m)}$ such that for all sufficiently large $m \in \mathbb{N}$ it holds that $\Pr_{z \leftarrow F^{\mathtt{pad}}(1^m)}[D^{\mathtt{pad}}(z) \neq L^{\mathtt{pad}}(z)] \leq 1/t(m)$.

We define the algorithm D in the natural way, i.e. $D(x) = D^{\mathtt{pad}}(x, 1^{t(|x|)})$, and note that this algorithm runs in time $n^{\mathrm{polyloglog}(n)}$. Assume towards a contradiction that there exists a t-time algorithm F and an infinite set of input lengths $B_F \subseteq \mathbb{N}$ such that for every $n \in B_F$, with probability more than $1/t(n)$ it holds that $D(x) \neq L(x)$. Consider the algorithm $F^{\mathtt{pad}}$ that on input of the form $1^{n+t(n)}$ runs $F(1^n)$ to obtain $x \in \{0,1\}^n$, and outputs $(x, 1^n)$ (on inputs of another form $F^{\mathtt{pad}}$ fails and halts), and let $B_{F^{\mathtt{pad}}} = \{n + t(n) : n \in B_F\}$. For any $m \in B_{F^{\mathtt{pad}}}$ we have that

$$\Pr_{z \leftarrow F^{\mathtt{pad}}(1^m)}[D^{\mathtt{pad}}(z) \neq L^{\mathtt{pad}}(z)] = \Pr_{x \leftarrow F(1^n)}[D(x) \neq L(x)] > 1/t(n) > 1/t(m)\,,$$

which yields a contradiction. ■

An aside: Derandomization using quasilogarithmic space. The PRG constructed in Lemma 4.9 actually works in *quasilogarithmic space* (since $f^{\mathtt{ws}}$ is computable in linear space), except for one crucial part: The construction of combinatorial designs.

[30] Note that $\mathtt{str}$ is well-defined, since we can assume without loss of generality that $\overline{S_n} \cap \overline{S_{n'}} = \emptyset$ for distinct $n, n' \in B_A$ (i.e., we can assume without loss of generality that n and n' are sufficiently far apart).

Combinatorial designs with parameters as in our proof actually *can* be constructed in logarithmic space, but only for values of ℓ that are of a specific form (since the constructions are algebraic).[31] However, in our downward self-reducibility argument we need such designs for *every* integer ℓ (such that we can assume the existence of distinguishers on the set $S_n = \ell^{-1}(2L_n)$, and hence of learners for $f^{\mathtt{GL}(\mathtt{ws})}$ on $2L_n$).

4.3 Proofs of Theorems 1.1 and 1.2

Let us now formally state Theorem 1.1 and prove it. The theorem follows immediately as a corollary of Lemma 4.7 and Proposition 4.8.

Theorem 4.13 (rETH $\Rightarrow$ i.o.-PRG for uniform circuits). *Assume that there exists $i \geq 1$ such that $\mathtt{TQBF} \notin \mathcal{BPTIME}[2^{n/\log(n)^i}]$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\mathrm{loglog}(n)^k}$ there exists a $(1/t)$-i.o.-PRG for $(t, \log(t))$-uniform circuits that has seed length $\tilde{O}(\log(n))$ and is computable in time $n^{\mathrm{polyloglog}(n)}$.*

Proof. Let $\delta(n) = 2^{n/\log(n)^{3c}}$ for a sufficiently large constant $c \in \mathbb{N}$. By Lemma 4.7, there exists $(\delta, O(\ell^2))$-well-structured function $f^{\mathtt{ws}}$ that is computable in linear space, and such that $\mathtt{TQBF}$ reduces to $f^{\mathtt{ws}}$ in time $\mathtt{ql}(n) = n \cdot \log(n)^{2c+r}$, where $r \in \mathbb{N}$ is a universal constant. Using our hypothesis, we deduce that $f^{\mathtt{ws}}$ cannot be computed in probabilistic time $2^{n/\log(n)^{3c-1}}$; this is the case since otherwise, $\mathtt{TQBF}$ could have been computed in probabilistic time

$$2^{\mathtt{ql}(n)/\log(\mathtt{ql}(n))^{3c-1}} = 2^{n\cdot\log(n)^{2c+r}/\log(\mathtt{ql}(n))^{3c-1}} < 2^{n/\log(n)^{c-r-1}}, \tag{4.4}$$

which is a contradiction if $c \geq i + r + 1$. Our conclusion now follows from Proposition 4.8. ■

We also formally state Theorem 1.2 and prove it, as a corollary of Lemma 4.7 and of Propositions 4.11 and 4.12.

Theorem 4.14 (a.a.-rETH $\Rightarrow$ almost-always HSG for uniform circuits and alm0st-always "average-case" derandomization of $\mathcal{BPP}$). *Assume that there exists $i \geq 1$ such that $\mathtt{TQBF} \notin$ i.o.-$\mathcal{BPTIME}[2^{n/\log(n)^i}]$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\mathrm{loglog}(n)^k}$:*

1. *There exists a $(1/t)$-HSG for $(t, \log(t))$-uniform circuits that is computable in time $n^{\mathrm{polyloglog}(n)}$ and has seed length $\tilde{O}(\log(n))$.*

2. *For every $L \in \mathcal{BPTIME}[t]$ and probabilistic t-time algorithm F there exists a deterministic machine D that runs in time $n^{\mathrm{polyloglog}(n)}$ and gets $O(\mathrm{logloglog}(n))$ bits of non-uniform advice such that for all sufficiently large $n \in \mathbb{N}$ the probability (over coin tosses of F) that $F(1^n)$ is an input $x \in \{0,1\}^n$ for which $D(x) \neq L(x)$ is at most $1/t(n)$.*

[31] This can be done using an idea from [HR03, Lemma 5.5] (attributed to Salil Vadhan), essentially "composing" Reed-Solomon codes over $GF(n)$ of degree $n/\mathrm{polylog}(n)$ with standard designs (a-la Nisan and Wigderson [NW94]; see [HR03, Lemma 2.2]) with set-size $\ell = \mathrm{polylog}(n)$.

Proof. Note that both Proposition 4.11 and Proposition 4.12 rely on the same hypothesis, and that their respective conclusions correspond to Items (1) and (2) in our claim. Thus, it suffices to prove that their hypothesis holds.

To see this, as in the proof of Theorem 4.13, let $\delta(n) = 2^{n/\log(n)^{3c}}$ for a sufficiently large constant $c \in \mathbb{N}$, and let $f^{\mathtt{ws}}$ be the $(\delta, \mathrm{polylog}(n))$-well-structured function that is obtained from Lemma 4.7 with parameter δ. Let $r \in \mathbb{N}$ be the universal constant from Lemma 4.7, and let $\mathtt{ql}(n) = n \cdot \log(n)^{2c+r}$. Note that for every algorithm that runs in time $2^{n/\log(n)^{3c-1}}$ and every sufficiently large $n_0 \in \mathbb{N}$, the algorithm fails to compute $f^{\mathtt{ws}}$ on input length $n = \mathtt{ql}(n_0)$; this is because otherwise we could have computed $\mathtt{TQBF}$ on infinitely-often n_0's in time $2^{n/\log(n)^{c-r-1}} \le 2^{n_0/\log(n_0)^k}$, where the calculation is as in Eq. (4.4). This implies the hypothesis of Propositions 4.11 and 4.12. ■

5 NOT-rETH and circuit lower bounds from randomized algorithms

In this section we prove Theorem 1.3. We first show the desired $\mathcal{BPE}$ lower bounds follow from a non-trivial weak learner of general circuits of quasi-linear size, and then show such a weak learner follows from the $2^{n/\mathrm{polylog}(n)}$-time randomized $\mathtt{CircuitSAT}$ algorithm for roughly quadratic-size circuits.

We are going to apply Corollary 4.10 to show that non-trivial weak learners imply faster randomized algorithms for $\mathtt{TQBF}$. For that purpose, we first generalize the definition of weak learners so that the algorithm is now required to learn any possible small oracle circuits.

For a function $O : \{0,1\}^n \to \{0,1\}$, we also use $\mathcal{SIZE}(O)$ to denote the size of the smallest circuit computing O.

Definition 5.1 (weak learner for general circuits). *For $S : \mathbb{N} \to \mathbb{N}$ and $\delta : \mathbb{N} \to \mathbb{R}$, we say that a randomized oracle machine A is a δ-*weak learner* for S-size circuits, if the following holds.*

- *On input 1^n, A is given oracle access to an oracle $O : \{0,1\}^n \to \{0,1\}$, and runs in time $\delta^{-1}(n)$.*
- *If $\mathcal{SIZE}(O) \le S(n)$, then with probability at least δ, A outputs a circuit C on n input bits with size $\le S(n)$ such that C computes O correctly on at least a $1/2 + \delta$ fraction of inputs.*

Next, we need the following standard diagonalization argument.

Proposition 5.2 (diagonalization against circuits in Σ_4). *Let $\delta = 2^{-n/\mathrm{polylog}(n)}$, k_{ckt} be a constant, and $f^{\mathtt{ws}}$ be the δ-well-structured function guaranteed by Lemma 4.7, there is a language L^{diag} which is $n \cdot \mathrm{polylog}(n)$-time reducible to $f^{\mathtt{ws}}$, and $L^{\mathsf{diag}} \notin \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$.*

Proof. Let $s = n \cdot (\log n)^{k_{\mathsf{ckt}}}$ and $s' = s \cdot \log n$. By standard arguments, there exists an s'-size circuit on n bits which cannot be computed by s-size circuits.

Consider the following Σ_4 algorithm:

- Given an input $x \in \{0,1\}^n$, we guess a circuit C of size s' on n input bits, and reject immediately if $C(x) = 0$. Then we check the following two conditions and accept if and only if both of them are satisfied.
- (A): For all circuits D on n input bits with size $\le s$, there exists an input $y \in \{0,1\}^n$ such that $C(y) \ne D(y)$. That is, C cannot be computed by any circuit with size $\le s$.
- (B): For all circuits D on n input bits with size s' such that the description of D is lexicographically smaller than that of C, there exists a circuit E with size $\le s$ such that for all $y \in \{0,1\}^n$, $E(y) = D(y)$. That is, C is the lexicographically first s'-size circuit which cannot be computed by s-size circuits.

Clearly, the above algorithm can be formulated as an $n \cdot \mathrm{polylog}(n)$-size $\Sigma_4 SAT$ instance, and therefore also an $n \cdot \mathrm{polylog}(n)$-size `TQBF` instance (which can be further reduced to $f^{\mathtt{ws}}$ in $n \cdot \mathrm{polylog}(n)$ time). Moreover, it is easy to see that it computes the truth-table of the lexicographically first s'-size circuit on n input bits which cannot be computed by any circuit with size $\le s$.

Therefore, we can set L^{diag} to be the language computed by the above algorithm. ∎

Remark 5.3. *We remark that the standard $\Sigma_3 P$ construction of a truth-table hard for s-size circuits actually takes $\widetilde{O}(s^2)$ time: in which one first existentially guesses an s'-length (where $s' = s \cdot \mathrm{polylog}(s)$) truth-table L, then enumerates all possible s-size circuits C and all s'-length truth-tables L' such that $L' < L$ (lexicographically), and checks there exists an input x such that $C(x) \ne L(x)$, and an s-size circuit C' computing L'. In the last step, checking C' computing L' requires evaluating C' on s' many inputs, which takes $\widetilde{O}(s^2)$ time.*

Now we are ready to show that non-trivial learning algorithms imply non-trivial circuit lower bounds for $\mathcal{BPE}$.

Theorem 5.4 (non-trivial learning algorithms imply $\mathcal{BPE}$ lower bounds)**.** *For any constant $k_{\mathsf{ckt}} > 0$, there is another constant $k_{\mathsf{learn}} = k_{\mathsf{learn}}(k_{\mathsf{ckt}})$, such that letting $\delta_{\mathsf{learn}} = 2^{-n/(\log n)^{k_{\mathsf{learn}}}}$, if there is a δ_{learn}-weak learner for $n \cdot (\log n)^{k_{\mathsf{ckt}}}$-size circuits, then $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$.*

Proof. Let $\delta = 2^{-n/(\log n)^{k_\delta}}$ where k_δ is a large enough constant depending on k_{ckt}. Let $f^{\mathtt{ws}}$ be the δ-well-structured function guaranteed by Lemma 4.7.

Recall that $f^{\mathtt{ws}} \in \mathcal{SPACE}[O(n)]$. Hence, the Boolean function $f^{\mathtt{GL}(\mathtt{ws})}$, which is defined as in the proof of Lemma 4.9, is computable in $\mathcal{SPACE}[O(n)]$ as well.

We can safely assume $f^{\mathtt{GL}(\mathtt{ws})} \in \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$ as otherwise the theorem follows immediately. Then, by our assumption, it follows that there is a δ_{learn}-weak learner for $f_n^{\mathtt{GL(ws)}}$. Applying Corollary 4.10 and setting $k_{\mathsf{learn}} = k_\delta$, it follows that $f^{\mathtt{ws}}$ can be computed by randomized $T^{ws}(n) \overset{\text{def}}{=\!=} 2^{n/(\log n)^{k_{\mathsf{learn}}-1}}$.

Let L^{diag} be the language guaranteed by Proposition 5.2 such that $L^{\mathsf{diag}} \notin \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$, and $d = d(k_{\mathsf{ckt}})$ be a constant such that L^{diag} is $n \cdot (\log n)^d$-time reducible to $f^{\mathtt{ws}}$. We can then compute L_n^{diag} in randomized $T^{ws}(n \cdot (\log n)^d) = 2^{o(n)}$ time, by setting k_{learn} to be large enough. Therefore, it follows that $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$. ■

5.1 Randomized `CircuitSAT` algorithms imply $\mathcal{BPE}$ circuit lower bounds

We now prove Theorem 1.3, which asserts that "non-trivial" randomized algorithms that solve `CircuitSAT` in time $2^{n/\mathrm{polylog}(n)}$ imply circuit lower bounds against $\mathcal{BPE}$. As explained in Section 2.2, we do so by showing that "non-trivial" randomized algorithms for `CircuitSAT` imply the weak learner for quasi-linear size circuits, which enables us to apply Theorem 5.4.

Reminder of Theorem 1.3. *For any constant $k_{\mathsf{ckt}} \in \mathbb{N}$ there exists a constant $k_{\mathsf{sat}} \in \mathbb{N}$ such that the following holds. If* `CircuitSAT` *for circuits over n variables and of size $n^2 \cdot (\log n)^{k_{\mathsf{sat}}}$ can be solved in probabilistic time $2^{n/(\log n)^{k_{\mathsf{sat}}}}$, then $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$.*

Proof. Let $s = s(n) = n \cdot (\log n)^{k_{\mathsf{ckt}}}$. Let k_{learn} and δ_{learn} be as in Theorem 5.4 such that a δ_{learn}-weak learner for s-size circuits implies that $\mathcal{BPE} \not\subset \mathcal{SIZE}[s]$. In the following we construct such a weak learner A with the assumed `CircuitSAT` algorithm. In fact, we are going to construct a stronger learner such that:

- If $\mathcal{SIZE}(O) \le s(n)$, then with probability at least $2/3$, A outputs a circuit C on n input bits with size $\le s(n)$ such that C computes O correctly on at least a 0.99 fraction of inputs.

Let $k_{\mathsf{sat}} = k_{\mathsf{sat}}(k_{\mathsf{ckt}})$ be a constant to be specified later. The learner A first draws $t = n \cdot (\log n)^{k_{\mathsf{ckt}}+2}$ uniform random samples $x_1, x_2, \dots, x_t$ from $\{0,1\}^n$, and asks O to get $y_i = O(x_i)$ for all $i \in [t]$. Note that A operates *incorrectly* if and only if $\mathcal{SIZE}(O) \le s(n)$ and it outputs a circuit D of size $\le s(n)$ such that $\Pr_{x \in \{0,1\}^n}[O(x) = D(x)] < 0.99$.

We say that a circuit D is bad if it has size $\le s(n)$ and $\Pr_{x \in \{0,1\}^n}[O(x) = D(x)] < 0.99$. For a fixed bad circuit D, by a Chernoff bound, with probability at least $1 - 2^{-\Omega(t)}$, we have $D(x_i) \neq y_i$ for some i. Since there are at most $n^{O(s)}$ bad circuits, with probability at least $1 - n^{O(s)} \cdot 2^{-\Omega(t)} \ge 1 - 2^{-\Omega(t)+O(s)\cdot \log n} = 1 - 2^{-\Omega(t)}$ (the last equality follows as $t = n \cdot (\log n)^{k_{\mathsf{ckt}}+2}$), it follows that for every bad circuit D there exists an index i such that $D(x_i) \neq y_i$. In the following we condition on such a good event.

By repeating the `CircuitSAT` algorithm $O(n)$ times and taking the majority of the outputs, we can assume without loss of generality that the `CircuitSAT` algorithm has an error probability of at most 2^{-n}. Now, we use the randomized CircuitSAT algorithm to construct a circuit C of size $\leq s(n)$ such that $C(x_i) = y_i$ for all i, bit-by-bit (this can be accomplished with the well-known search-to-decision reduction for SAT) with probability at least 0.99. Note that in each iteration, the length of the input to the CircuitSAT algorithm is the length of the description of a circuit of size $s(n)$, and hence at most $s'(n) = O(n \cdot (\log n)^{k_{\mathsf{ckt}}+1})$. Setting k_{sat} large enough, it follows that A runs in randomized $(\delta_{\mathsf{learn}}(n))^{-1}$ time.

Assuming $\mathcal{SIZE}(O) \leq s(n)$, such circuits exist, and we can find one with probability at least 0.99. Conditioning on the good event, this circuit cannot be bad, and therefore it must agree with O on at least a 0.99 fraction of inputs. Putting everything together, when $\mathcal{SIZE}(O) \leq s(n)$, the algorithm A outputs a circuit C such that $\Pr_{x \in \{0,1\}^n}[O(x) = D(x)] \geq 0.99$ with probability at least $0.99 - 2^{-\Omega(t)} \geq 2/3$, which completes the proof. ■

5.2 Randomized Σ_2-SAT$[n]$ algorithms imply $\mathcal{BPE}$ circuit lower bounds

One shortcoming of Theorem 1.3 is that the hypothesized algorithm needs to decide the satisfiability of an n-bit circuit of size $\tilde{O}(n^2)$, rather than the satisfiability of circuits (or of 3-SAT formulas) of linear size.[32] To address this shortcoming, we now prove a different version of Theorem 1.3, which asserts that "non-trivial" randomized algorithms that solve Σ_2-SAT for formulas of *linear size* in time $2^{n/\mathrm{polylog}(n)}$ imply circuit lower bounds against $\mathcal{BPE}$.

Theorem 5.5 (randomized Σ_2-SAT algorithms imply circuit lower bounds against $\mathcal{BPE}$**).** *For any constant $k_{\mathsf{ckt}} > 0$, there is another constant $k_{\mathsf{sat}} = k_{\mathsf{sat}}(k_{\mathsf{ckt}})$ such that if Σ_2-SAT with n variables and n clauses can be decided in randomized $2^{n/(\log n)^{k_{\mathsf{sat}}}}$ time, then $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$.*

Proof. Let $\mathtt{TQBF}^{\mathtt{loc}}$ be the function from Claim 4.7.1, and recall that $\mathtt{TQBF}^{\mathtt{loc}} \in \mathcal{SPACE}[O(n)]$. Therefore, we can safely assume $\mathtt{TQBF}^{\mathtt{loc}} \in \mathcal{SIZE}[s(n)]$, for $s(n) = n \cdot (\log n)^{k_{\mathsf{ckt}}}$.

Now we describe a randomized algorithm computing a circuit for $\mathtt{TQBF}^{\mathtt{loc}}$ on inputs of length n. First, it computes the trivial circuit of size-$s(1)$ for ${\mathtt{TQBF}^{\mathtt{loc}}}_1$. Now, suppose we have an $s(m)$-size circuit C_m computing ${\mathtt{TQBF}^{\mathtt{loc}}}_m$ where $m < n$, we wish to find an $s(m+1)$-size circuit for ${\mathtt{TQBF}^{\mathtt{loc}}}_{m+1}$.

By the downward self-reducibility of $\mathtt{TQBF}^{\mathtt{loc}}$, we can obtain directly an $O(s(m))$-size circuit D for ${\mathtt{TQBF}^{\mathtt{loc}}}_{m+1}$. Our goal is to utilizing the circuit D and our fast Σ_2-SAT algorithm to compute an $s(m+1)$-size circuit for ${\mathtt{TQBF}^{\mathtt{loc}}}_{m+1}$. Consider the following Σ_2-SAT question: given a prefix p, is there an $s(m+1)$ circuit C whose description starts with p, such that for all $x \in \{0,1\}^{m+1}$ we have $C(x) = D(x)$. This can be

[32]Since we are interested in algorithms that run in time $2^{n/\mathrm{polylog}(n)}$ for a sufficiently large polylogarithmic function, there is no significant difference for us between circuits and 3-SAT formulas of linear (or quasilinear) size. This is since any circuit can be transformed to a formula with only a polylogarithmic overhead, using an efficient Cook-Levin reduction; and since we can "absorb" polylogarithmic overheads by assuming that the polylogarithmic function in the running time $2^{n/\mathrm{polylog}(n)}$ is sufficiently large.

formulated by a Σ_2-SAT instance of $n \cdot \mathrm{polylog}(n)$ size. By fixing the description bit by bit, we can obtain an $s(m+1)$-size circuit for $\mathtt{TQBF^{loc}}_{m+1}$. The success probability can be boosted to $1 - 2^{-2n}$ by repeating each call to the Σ_2-SAT algorithm a polynomial number of times and taking the majority.

Let L^{diag} be the language guaranteed by Proposition 5.2, and d be a constant such that L^{diag} is $n \cdot (\log n)^d$-time reducible to $\mathtt{TQBF^{loc}}$. By setting k_{sat} large enough, we can compute $\mathtt{TQBF^{loc}}_{n\cdot(\log n)^d}$ (and therefore also L_n^{diag}) in $2^{o(n)}$ time, Therefore, it follows that $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$. ■

Finally, we now use a "win-win" argument to deduce, unconditionally, that either we have an average-case derandomization of $\mathcal{BPP}$, or $\mathcal{BPE}$ is "hard" for circuits of quasilinear size (or both statements hold). An appealing interpretation of this result is as a Karp-Lipton-style theorem: If $\mathcal{BPE}$ has circuits of quasilinear size, then $\mathcal{BPP}$ can be derandomized in average-case.

Corollary 5.6 (a "win-win" result for average-case derandomization of $\mathcal{BPP}$ and circuit lower bounds against $\mathcal{BPE}$)**.** *At least one of the following statements is true:*

1. *For every constant $k \in \mathbb{N}$ it holds that $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^k]$.*
2. *For every constant $k \in \mathbb{N}$ and for $t(n) = n^{\mathrm{loglog}(n)^k}$ there exists a $(1/t)$-i.o.-PRG for $(t, \log(t))$-uniform circuits that has seed length $\tilde{O}(\log(n))$ and is computable in time $n^{\mathrm{polyloglog}(n)}$.*

Proof. If for every $k' \in \mathbb{N}$ it holds that Σ_2-SAT for n-bit formulas with $O(n)$ clauses can be decided by probabilistic algorithms that run in time $2^{n/(\log n)^{k'}}$, then by Theorem 5.5 we have that Item (1) holds. Otherwise, for some $k' \in \mathbb{N}$ it holds that Σ_2-SAT for n-bit formulas with $O(n)$ clauses cannot be decided by probabilistic algorithms that run in time $2^{n/(\log n)^{k'}}$. In particular, since solving satisfiability of a given n-bit Σ_2 formula with $O(n)$ clauses can be reduced in linear time to solving TQBF, we have that $\mathtt{TQBF} \notin \mathcal{BPTIME}[2^{n/(\log n)^{k'+1}}]$. In this case, Item (2) follows from Theorem 4.13. ■

We note that to prove Corollary 5.6 we do not have to use Theorem 5.5. An alternative proof relies on the fact that the Σ_4 formula from the proof of Proposition 5.2 can be constructed in polynomial time. In particular, if TQBF can be decided in probabilistic time $2^{n/\mathrm{polylog}(n)}$ for an arbitrarily large polylogarithmic function, then for every k_{ckt} we can construct the corresponding Σ_4 formula from Proposition 5.2 in polynomial time, and decide its satisfiability in probabilistic time $2^{o(n)}$, which implies that $L^{\mathsf{diag}} \in \mathcal{BPE}$; Item (1) of Corollary 5.6 then follows. Otherwise, we have that TQBF cannot be solved in probabilistic time $2^{n/\mathrm{polylog}(n)}$ for some polylogarithmic function; then we can invoke Theorem 4.13 to deduce Item (2) of Corollary 5.6.

6 NETH and the equivalence of derandomization and circuit lower bounds

Recall that our results in this section show two-way implications between the statement that derandomization and circuit lower bounds are equivalent, and a very weak variant of NETH. Specifically, the latter variant is that $\mathcal{E}$ does not have $\mathcal{NTIME}[T]$-uniform circuits of small size. Let us now properly define this notion:

Definition 6.1 ($\mathcal{NTIME}[T]$-uniform circuits)**.** *For $S, T : \mathbb{N} \to \mathbb{N}$, we say that a set $L \subseteq \{0,1\}^*$ can be decided by $\mathcal{NTIME}[T]$-*uniform circuits of size *S if there exists a non-deterministic machine M that gets input 1^n, runs in time $T(n)$, and satisfies the following:*

1. *For every $n \in \mathbb{N}$ there exist non-deterministic choices such that $M(1^n)$ outputs a circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most $S(n)$ that decides L_n.*
2. *Whenever M outputs a circuit $C : \{0,1\}^n \to \{0,1\}$, the circuit C decides L_n.*

*When we simply say that L can be decided by $\mathcal{NTIME}[T]$-*uniform circuits *(without specifying a size bound S), we consider the trivial size bound $S(n) \leq T(n)$.*

This section is organized as follows. We first prove, in Section 6.1, the main technical results that underlie Theorems 1.4 and 1.5. Then, in Section 6.2 we prove Theorems 1.4 and 1.5. Finally, in Section 6.3 we prove Theorem 1.6.

6.1 A refined Karp-Lipton-style result

In this section we prove the main technical results that underlie Theorems 1.4 and 1.5. Specifically, we prove a refined Karp-Lipton-style result, which asserts that if $\mathcal{E}$ has small circuits, and $pr\mathcal{BPP}$ has quick non-deterministic algorithms, then $\mathcal{E}$ has small circuits that can be constructed by quick non-deterministic machines.

The first ingredient that we will use is a set $L \in \mathcal{E}$ such that L has an instance checker with linear-length queries, and is randomly self-reducible, and both the instance checker and the random self-reducibility algorithm use only a linear number of random bits. Let us properly define these notions and prove that such L exist:

Definition 6.2 (instance checkers)**.** *A probabilistic polynomial-time oracle machine M is an* instance checker *for a set $L \subseteq \{0,1\}^*$ if for every $x \in \{0,1\}^*$ the following holds:*

1. *(Completeness.) $M^L(x) = L(x)$, with probability one.*
2. *(Soundness.) For every $L' \subseteq \{0,1\}^*$ we have that $\Pr[M^{L'}(x) \notin \{L(x), \perp\}] \leq 1/6$.*

For $\ell : \mathbb{N} \to \mathbb{N}$, if for every $x \in \{0,1\}^*$, all the oracle queries of M on input x are of length at most $\ell(|x|)$, then we say that M has queries of length ℓ*. We will also measure the maximal* number of queries *that M makes on inputs of any given length.*

Definition 6.3 (random self-reducible function). *We say that $f : \{0,1\}^* \to \{0,1\}^*$ is* randomly self-reducible *if there exists a probabilistic oracle machine A that gets input $x \in \{0,1\}^n$ and access to an oracle $g : \{0,1\}^* \to \{0,1\}^*$, runs in time* $\mathrm{poly}(n)$, *and if every query $q \in \{0,1\}^n$ that it issues is answered by $f(q)$, then $A^g(x) = f(x)$.*

Proposition 6.4 (an $\mathcal{E}$-complete problem that is random self-reducible and has a good instance checker). *There exists $L \in \mathcal{E}$ such that L is defined non-trivially only on input lengths of the form $n = 2m \cdot \lceil \log(n_0) \rceil + \lceil \mathrm{loglog}(n_0) \rceil + 1$ where $m = O(n_0/\log(n_0))$ is an integer, and L has the following properties:*

1. *Any $L' \in \mathcal{DTIME}[2^n]$ can be reduced to L in linear time, where the reduction produces inputs of length n as above (i.e., on which L is defined non-trivially).*

2. *There is an instance checker for L that on inputs of length n uses $O(n)$ random bits and makes $O(1)$ queries of length $\ell(n) = O(n)$.*

3. *The set L is randomly self-reducible by an oracle machine A that on inputs of length n uses $O(n)$ random bits, and its queries come in batches, where each batch is defined by a uniform choice of prefix $q_1 \in \{0,1\}^{2m \cdot \lceil \log(n_0) \rceil}$ that is then extended by every possible suffix $q_2 \in \{0,1\}^{\lceil \mathrm{loglog}(n_0) \rceil + 1}$; that is, the batch defined by q_1 consists of the $O(\log(n))$ queries $\{(q_1, q_2) \in \{0,1\}^n : q_2 \in \{0,1\}^{\lceil \mathrm{loglog}(n_0) \rceil + 1}\}$.*

Proof. Let $L_0 = \{(\langle M \rangle, x) : M \text{ accepts } x \text{ in } 2^{|x|} \text{ steps}\}$, and note that $L_0 \in \mathcal{E}$ and that any set in $\mathcal{DTIME}[2^n]$ can be reduced to L_0 in linear time. Let $f_{L_0} : \{0,1\}^* \to \{0,1\}^*$ be the low-degree extension of L_0 where inputs of size n_0 for L_0 are mapped to m-variate inputs, where $m = O(n_0/\mathrm{loglog}(n_0))$, for a polynomial of individual degree $\lceil \sqrt{n_0} \rceil$ over a field of size $|\mathbb{F}| = 2^{2^{\lceil \log(n_0) \rceil}}$. Finally, let L be the set of inputs of the form $(z, i) \in \{0,1\}^{m \cdot \lceil \log(|\mathbb{F}|) \rceil} \times \{0,1\}^{\lceil \mathrm{loglog}(|\mathbb{F}|) \rceil}$ such that the i^{th} bit of $f_{L_0}(z) \in \mathbb{F}$ equals one. Note that L is indeed defined non-trivially only on inputs of the form $n = 2m \cdot \lceil \log(n_0) \rceil + \lceil \mathrm{loglog}(n_0) \rceil + 1 = O(n_0)$, and that L_0 is linear-time reducible to L, which means that any $L' \in \mathcal{DTIME}[2^n]$ is linear-time reducible to L.

To see that L is randomly self-reducible as claimed, consider the standard random self-reducibility algorithm for f_{L_0}, denoted A_0 (see, e.g., [Gol08, Sec. 7.2.1.1]). Recall that A_0 uniformly chooses two points $\vec{u}, \vec{v} \in \mathbb{F}^m$, which requires $O(n)$ random bits, and then queries its oracle on a fixed set of points on the line corresponding to $\vec{u}, \vec{v}$. Now, given input $(z, i) \in \{0,1\}^{m \cdot \lceil \log(|\mathbb{F}|) \rceil + \lceil \mathrm{loglog}(|\mathbb{F}|) \rceil}$, we simulate A_0 on input $z \in \mathbb{F}^m$, and answer each of its oracle queries $q_1 \in \mathbb{F}^m$ by querying L on all values of $q = (q_1, q_2)$ for $q_2 \in \{0,1\}^{\lceil \mathrm{loglog}(|\mathbb{F}|) \rceil}$. Indeed, if each query q is answered by $L(q)$, then we can reconstruct the value $f_{L_0}(z)$ and answer the query q_1 of A_0 correctly.

To see that L has an instance checker that uses $O(1)$ queries of length $O(n)$, fix a PCP system for $\mathcal{E}$ with a polynomial-time verifier that uses $O(n)$ bits of randomness and $O(1)$ queries, and with proofs that can be constructed in time $2^{O(n)}$.[33] Now, since

[33]Such PCPs follow from the PCPP of Miele [Mie09], using a standard PCPP-to-PCP transformation (as in [DR06; BS+06]). For explicit statements that proofs in the PCPP of [Mie09] can be constructed in time $2^{O(n)}$, see e.g. [BS+17, Thm 2.9 and Rmk 2.10] or [RZR19, Thm 8.4 and Rmk 8.5].

the set $\{(z,i,b) : L(z,i) = b\}$ is in $\mathcal{E}$, it has a PCP verifier as above, and there exists a mapping $(z,i,b) \mapsto \pi(z,i,b)$ of inputs to proofs for this verifier that can be computed in time $2^{O(n)}$. To correctly answer the verifier's queries it suffices to decide the set $Q = \{((z,i,b),j) : \pi(z,i,b)_j = 1\}$. Since Q is in $\mathcal{E}$, it can be reduced to L. The instance checker for L simulates the PCP verifier on inputs $(z,i,0)$ and $(z,i,1)$, and whenever the verifier issues a query to the PCP proof, the instance checker reduces this query to an input for Q, and further reduces this to an input for L; since the PCP proof is of length $2^{O(n)}$, and the reductions to Q and to L incur only a linear overhead, the (constantly-many) queries of the instance checker are of length $O(n)$. ■

We will also need a relatively *standard* Karp-Lipton-style result, which asserts that if $\mathcal{E}$ has small circuits and $pr\mathcal{BPP}$ can be solved by a quick non-deterministic, then $\mathcal{E} \subseteq \mathcal{NTIME}[T'] \cap co\mathcal{NTIME}[T']$ for a small function T'.

Proposition 6.5 (a standard Karp-Lipton-style result, following [Bab+93]). *There exists a constant $k > 1$ and $L \in \mathcal{E}$ such that the following holds. Let $S,T : \mathbb{N} \to \mathbb{N}$ be time-computable functions such that $S(n) \geq n$ and $T(n) \geq n$ for every $n \in \mathbb{N}$. Assume that $L \in \mathcal{SIZE}[S]$ and that $\mathsf{CAPP}[S(k \cdot n)^k] \in pr\mathcal{NTIME}[T]$. Then, $\mathcal{DTIME}[2^n] \subseteq \mathcal{NTIME}[T'] \cap co\mathcal{NTIME}[T']$, where $T'(n) = T(S(k \cdot n)^k)^k$.*

Proof. Let $L_0 \in \mathcal{DTIME}[2^n]$, and let us construct a non-deterministic machine that gets input $x_0 \in \{0,1\}^{n_0}$, and for some non-deterministic choices outputs $L(x_0)$, but for all other non-deterministic choices outputs $\perp$. The non-deterministic machine first computes $x \in \{0,1\}^n$, where $n = O(n_0)$, such that $x_0 \in L_0$ if and only if $x \in L$, where L is the problem from Proposition 6.4.

Let $n' = \ell(n) = O(n) = O(n_0)$. By our hypothesis, there exists a circuit over n' input bits of size $S(n')$ that decides $L_{n'}$. Our machine non-deterministically guesses a circuit $C_L : \{0,1\}^{n'} \to \{0,1\}$ of size $S(n')$, and constructs a circuit $C : \{0,1\}^{O(n')} \to \{0,1,\perp\}$ that compute the decision of the machine M from Proposition 6.4 on input x as a function of the random coins of M, while resolving the oracle queries of M using C_L as a sub-circuit. Then, the machine converts C into two circuits $C^{(\mathtt{y})}$ and $C^{(\mathtt{n})}$ such that $C^{(\mathtt{y})}$ outputs 1 if and only if C outputs 1 (i.e., both $\perp$ and 0 are mapped to 0), and $C^{(\mathtt{n})}$ outputs 1 if and only if C outputs 0. Note that each of these circuits are over $O(n')$ input bits and of size $\mathrm{poly}(S(n'))$.

Now, recall that in general, $\mathsf{CAPP} \in \mathcal{NTIME}[t]$ if and only if $\mathsf{CAPP} \in \mathcal{NTIME}[t'] \cap co\mathcal{NTIME}[t']$ for $t' \approx t$; that is:

Fact 6.5.1. *For any time-computable $s,t : \mathbb{N} \to \mathbb{N}$, if $\mathsf{CAPP}[s] \in pr\mathcal{NTIME}[t]$ then $\mathsf{CAPP}[s] \in pr\mathcal{NTIME}[t] \cap co\text{-}pr\mathcal{NTIME}[t + O(n)]$.*

Proof. This follows since $\neg\mathsf{CAPP}$ can be reduced to CAPP in linear time and without any overhead in the input size (by negating the output bit of the given circuit). □

Thus, by Fact 6.5.1 and by our hypothesis, there exists a non-deterministic machine M' that gets as input a circuit C' and satisfies the following: If C' accepts all of its inputs, then there exist non-deterministic choices for M' such that $M'(C') = 1$, and for all other non-deterministic choices $M'(C') = \perp$; and if C' accepts at most $1/6$ of its inputs, then there exist non-deterministic choices such that $M'(C') = 0$, and for all other non-deterministic choices $M'(C') = \perp$. Our machine simulates M' on $C^{(\mathtt{y})}$ and if $M'(C^{(\mathtt{y})}) = 1$ then it outputs "yes", and then simulates M' on $C^{(\mathtt{n})}$ and if $M'(C^{(\mathtt{n})}) = 1$ outputs "no", and otherwise outputs $\perp$.

Note that both $C^{(\mathtt{y})}$ and $C^{(\mathtt{n})}$ are of size $\mathrm{poly}(n) \cdot S(n') \leq S(n')^{O(1)}$, and therefore the running time of our machine is at most $T'(n)$. Also note that if $x \in L$ then for some guess of C_L the circuit $C^{(\mathtt{y})}$ will have acceptance probability one, and will therefore be accepted given some non-deterministic choice of M'; and for all guesses of C_L the circuit $C^{(\mathtt{n})}$ accepts at most $1/6$ of its inputs, and hence for all non-deterministic choices of M' we have that $M'(C^{(\mathtt{n})}) \in \{0, \perp\}$, and thus x will never be rejected. By a symmetric argument, any $x \notin L$ will be rejected for some non-deterministic choices but will never be accepted. Hence $L_0 \in \mathcal{NTIME}[T'] \cap co\mathcal{NTIME}[T']$. ∎

We are now ready to prove the main result in this section. Recall that the proof was described in high-level in Section 2.3.

Proposition 6.6 (a refined Karp-Lipton-style result)**.** *There exists a constant $k > 1$ and $L \in \mathcal{E}$ such that the following holds. Let $S, T : \mathbb{N} \to \mathbb{N}$ be time-computable functions such that $S(n) \geq n$ and $T(n) \geq n$ for every $n \in \mathbb{N}$, and assume that $L \in \mathcal{SIZE}[S]$. Then:*

1. *If* $\mathsf{CAPP}[S(k \cdot n)^k] \in pr\mathcal{DTIME}[T]$ *then any $L_0 \in \mathcal{DTIME}[2^n]$ can be decided by $\mathcal{NTIME}[T']$-uniform circuits, where $T'(n) = T(S(k \cdot n)^k)^k$.*
2. *If* $\mathsf{CAPP}[S(k \cdot n)^k] \in pr\mathcal{NTIME}[T]$ *then any $L_0 \in \mathcal{DTIME}[2^n]$ can be decided by $\mathcal{NTIME}[T'']$-uniform circuits of size $n^k \cdot S(k \cdot n)$, where $T''(n) = T\left(S\left(S(k \cdot n)^k\right)^k\right)^k$.*

Proof. Let us first prove Item (2), and then explain how to modify the proof to obtain Item (1). Fixing any $L_0 \in \mathcal{DTIME}[2^n]$, we prove that there exist $\mathcal{NTIME}[T'']$-uniform circuits of size $\mathrm{poly}(n) \cdot S(O(n))$ for L_0. The non-deterministic machine gets input 1^{n_0}, and constructs a circuit $C : \{0,1\}^{n_0} \to \{0,1\}$ as follows. As a first step the machine will construct a *probabilistic* circuit C, and then convert C into a deterministic circuit.

Step 1: Reduction to L. The circuit C first computes the linear-time reduction from L_0 to L from Proposition 6.4; that is, C maps its input $x_0 \in \{0,1\}^{n_0}$ into $x \in \{0,1\}^n$, where $n = O(n_0)$, such that $x_0 \in L_0$ if and only if $x \in L$.

Step 2: Non-deterministic guess and verification of a circuit for L. Recall that inputs to L (that are targets of reductions from other sets in $\mathcal{DTIME}[2^n]$) are of the form $z = (z_1, z_2) \in \{0,1\}^n$, where $z_2 \in \{0,1\}^{c_0 \cdot \log\log(n)}$ and c_0 is a constant. Let M be the instance checker for L, let $n' = \ell(n) = O(n) = O(n_0)$, and let $c \in \mathbb{N}$ such that the number of queries of the machine A underlying the random self-reducibility of L makes at inputs of length n is at most n^c.

By our hypothesis, there exists a circuit $C_L : \{0,1\}^{n'} \to \{0,1\}$ of size $S(n')$ that decides $L_{n'}$. Our machine non-deterministically guesses such a circuit. Now, consider the following promise problem Π:

- The input is guaranteed to be a circuit $C_L : \{0,1\}^{n'} \to \{0,1\}$ of size $S(n')$.
- **YES instances:** The circuit C_L decides L.
- **NO instances:** It holds that

$$\Pr_{z_1}\left[\exists z_2 : \Pr[M^{C_L}(z_1,z_2) = \perp] > 1/6\right] > n^{-2c} , \tag{6.1}$$

  where the internal probability in the LHS of Eq. (6.1) is over the random choices of the machine M.

Note that $\Pi \in \mathit{pr}\text{-}\mathit{co}\mathcal{RP}$, since a probabilistic polynomial-time algorithm that gets C_L as input can estimate the value of the expression in the LHS of Eq. (6.1) up to accuracy $1/\text{poly}(n)$ and with high confidence (and since for YES instances this value is zero); moreover, using the sampler from Theorem 3.6, there is a probabilistic $\mathit{co}\mathcal{RP}$ algorithm for this problem that on input C_L of size $\tilde{O}(S'(n))$ uses only $O(n)$ random bits.[34] Hence, the problem Π is reducible in $\text{poly}(|C_L|)$ time to $\mathsf{CAPP}[S(O(n))^{O(1)}]$.

By our hypothesis that $\mathsf{CAPP}[S(O(n))^{O(1)}] \in \mathit{pr}\mathcal{NTIME}[T]$, there exists a non-deterministic machine M_Π for Π that gets input C_L, runs in time $T(\text{poly}(S(n')))$, accepts every YES instance on some non-deterministic path, and rejects every NO instance on all non-deterministic paths. Our non-deterministic machine simulates M_Π on its guess of C_L, and aborts unless M_Π reaches an accepting state. So from now on we assume that C_L is not a NO instance, or in other words that for at least $1 - 1/n^{2c}$ of the z_1's there does not exist z_2 such that $\Pr[M^{C_L}(z_1,z_2) = \perp] > 1/6$.

Step 3: Using the guessed circuit and the instance checker to get a "suitably-corrupt" version of L. Let $m = O(n)$ be the number of random bits that M uses on inputs of length n. Consider the following probabilistic algorithm P that gets input $z_1 \in$

[34]Specifically, the algorithm for estimating the LHS of Eq. (6.1) uses the sampler from Theorem 3.6 (with a sufficiently large constant for the accuracy parameter) to sample $D = \text{poly}(n)$ strings $z_1 \in \{0,1\}^{n - c_0 \cdot \text{loglog}(n)}$, and then uses this sampler again to sample $\text{poly}(n)$ strings $r_1, ..., r_D \in \{0,1\}^{O(n)}$ to be used as randomness for the machine M. For each z_1 in the first sample, and for each $z_2 \in \{0,1\}^{c_0 \cdot \text{loglog}(n)}$, and for each r_i, the algorithm simulates $M^{C_L}(z_1,z_2)$ with r_i as its fixed randomness, and thus obtain an estimate of $\Pr[M^{C_L}(z_1,z_2) = \perp]$. The algorithm considers a string z_1 "bad" if its estimate of the latter value is more than .01, and rejects its input C_L if and only if the fraction of "bad" z_1's is larger than $1/2n^{-2c}$. By the properties of the sampler, the probability that a fixed z_1 such that the $\Pr[M^{C_L}(z_1,z_2) = \perp] \geq 1/6$ will be considered "bad" by the algorithm is $\exp(-n)$; hence, by a union-bound, with probability $1 - \exp(-n)$ over the choice of randomness for the second sample, all strings z_1 in the first sample such that for some z_2 it holds that $\Pr[M^{C_L}(z_1,z_2) = \perp] \geq 1/6$ will be classified as "bad" by the algorithm. Conditioned on this event, the algorithm rejects, with high probability, every NO instance.

$\{0,1\}^{n-c_0\cdot\text{loglog}(n)}$, and outputs $2^{c_0\cdot\text{loglog}(n)} = \text{polylog}(n)$ symbols such that each symbol is either a bit or $\perp$:

1. The algorithm uses the sampler from Theorem 3.6, instantiated for output length m and with accuracy $1/n$, in order to obtain a sample of $D = \text{poly}(n)$ strings $r_1, ..., r_D \in \{0,1\}^m$.

2. For each $z_2 \in \{0,1\}^{c_0\cdot\text{loglog}(n)}$, the algorithm P outputs the majority vote among the values $\{v_i\}_{i\in[D]}$, where v_i is the output of $M^{C_L}(z_1, z_2)$ when simulated with the fixed randomness r_i.

Note that P uses $O(n)$ random bits. We claim that there exists a set G of $1 - 1/n^{2c}$ inputs z_1 such that for each such z_1, with probability at least $1 - \exp(-n)$ over the randomness of P it holds that $P(z_1) = \{L(z_1, z_2)\}_{z_2\in\{0,1\}^{c_0\cdot\text{loglog}(n)}}$. To see this, let G be the set of z_1's such that for all z_2 it holds that $\Pr[M^{C_L}(z_1, z_2) =\perp] \leq 1/6$, and recall that the density of G is at least $1 - n^{-2c}$. Note that for any $z_1 \in G$ and any z_2 we have that $\Pr[M^{C_L}(z_1, z_2) = L(z_1, z_2)] \geq 2/3$; this is the case because $\Pr[M^{C_L}(z_1, z_2) \neq L(z_1, z_2)] \leq \Pr[M^{C_L}(z_1, z_2) =\perp] + \Pr[M^{C_L}(z_1, z_2) = \neg L(x)] \leq 1/3$. Thus, for any fixed $z_1 \in G$, and any z_2, the probability (over the random choice of P) that the majority vote of the v_i's will not equal $L(z_1, z_2)$ is at most $\exp(-n)$. By a union-bound over the polylogarithmically-many z_2's, for any fixed $z_1 \in G$, the probability that for some z_2 it holds that the corresponding output bit of P is not $L(z_1, z_2)$ is $\exp(-n)$.

Step 4: Using random self-reducibility to decide L. The circuit C chooses $O(n)$ random bits to be used as randomness for P, and simulates the oracle machine A from Proposition 6.4 at input $x \in \{0,1\}^n$, while answering the oracle queries of A using the procedure P with the fixed randomness chosen in advance. We claim that with high probability this procedure outputs $L(x)$.

To see this, recall that A makes $\text{poly}(n)$ queries, where the queries are organized in batches such that each batch is defined by a uniform choice of $z_1 \in \{0,1\}^{n-c_0\cdot\text{loglog}(n)}$ and all choices of $z_2 \in \{0,1\}^{c_0\cdot\text{loglog}(n)}$. Since each z_1 is uniformly-distributed, and using a union-bound, the probability that all queries of A lie in the set G is at least $1 - n^{-c}$. Conditioned on this event, for each fixed choice of z_1, the probability over choice of randomness for P that $P(z_1)$ does not output $\{L(z_1, z_2)\}_{z_2}$ is at most $\exp(-n)$. Hence, by another union-bound, with high probability all the queries of A are answered correctly, in which case A outputs $L(x)$.

So far we described a machine that on input 1^{n_0} makes $\tilde{O}(S(n'))$ non-deterministic choices, verifies them in time $T(\text{poly}(S(n')))$, and then constructs a probabilistic circuit over n_0 input bits that first performs a linear-time reduction (of x_0 to x), and then simulates a polynomial-time machine A while answering its queries using the (guessed) circuit of size $S(n')$. Thus, the machine runs in time $T(S(O(n_0))^{O(1)})^{O(1)}$, and constructs a probabilistic circuit of size $S'(n_0) = \text{poly}(n_0) \cdot S(O(n_0))$ that uses $O(n_0)$ random bits.

Step 5: Converting to a deterministic circuit. Denote by $m = O(n_0)$ the number of random bits that C uses. We first use the sampler from Theorem 3.6, instantiated with output length n_0 and accuracy $1/n$, for randomness-efficient error-reduction of C. Specifically, we construct a circuit C' that gets input $x_0 \in \{0,1\}^{n_0}$ and randomness $r' \in \{0,1\}^{m'=O(m)}$, computes C at x_0 with the $\text{poly}(n)$ random strings $\{Samp(r',i)\}_{i\in[D]}$, and outputs a majority vote of the decisions of C. Thus, C' is of size $S''(n_0) = \text{poly}(n_0) \cdot S'(n_0) = \text{poly}(n_0) \cdot S(O(n_0))$. For any $x \in \{0,1\}^{n_0}$, the probability that $C'(x_0) \neq L_0(x_0)$ is at most $2^{-2\cdot n_0}$, and hence almost all of the random strings lead C' to correctly decide L_0 on all of the inputs.

Our goal now is to find (using non-determinism) a fixed random string for C' with which it will be correct on all inputs. We construct this fixed string iteratively, bit-by-bit. In each iteration we have a prefix $\sigma \in \{0,1\}^{i-1}$, where $i \in [m']$, and wish to extend by a bit while preserving the property that there exists a continuation of the string with which C' is correct on all inputs. The crucial point is that each iteration requires solving a decision problem in $\mathcal{E}$ on inputs of length $O(S''(n_0) \cdot \log(S''(n_0)))$. This is the case because the set $\{(\langle C' \rangle, \sigma) : \exists \sigma' \in \{0,1\}^{m'-|\sigma|} \; \forall x_0 \in \{0,1\}^{n_0} \; C'(x_0, \sigma\sigma') = L_0(x_0)\}$ is in $\mathcal{E}$, relying on the fact that there are at most $2^{O(n_0)}$ choices of continuations σ' and that $L_0 \in \mathcal{E}$. By Proposition 6.5, there exists a non-deterministic machine that in each iteration runs in time $T(S(O(S''(n_0) \cdot \log(S''(n_0))))^{O(1)})^{O(1)}$ and either allows us to extend the current prefix by a correct bit, or outputs $\perp$ (in which case we abort).

The running time of our non-deterministic machine is dominated by the last step, and is thus bounded by

$$\begin{aligned} T\left(S\left(O(S''(n_0) \cdot \log(S''(n_0)))\right)^{O(1)}\right)^{O(1)} &\leq T\left(S\left((n_0)^k \cdot S(k \cdot n_0) \cdot \log(S(k \cdot n_0))\right)^k\right)^k \\ &\leq T\left(S\left(S(2k \cdot n_0)^{2k}\right)^{2k}\right)^{2k} , \end{aligned}$$

where $k \in \mathbb{N}$ is a sufficiently large constant. The size of the circuit that the machine outputs is still $S''(n_0) = (n_0)^k \cdot S(k \cdot n_0)$.

The modifications needed to prove Item (1). Note that we used the hypothesis that $\mathsf{CAPP}[S(k \cdot n)^k] \in pr\mathcal{NTIME}[T]$ both in Step 2 and in Step 5. In the current proof we can use the stronger hypothesis that $\mathsf{CAPP}[S(k \cdot n)^k] \in pr\mathcal{DTIME}[T]$. We leave the first four steps of our construction without change (including the Step 2, for which we don't need the stronger hypothesis), and modify only the fifth step.

Recall that in the end of the fourth step, our non-deterministic machine ran in time $T(S(O(n_0))^{O(1)})^{O(1)}$ and constructed a probabilistic circuit C of size $S'(n_0)$. Now, instead of constructing C' in the fifth step, our non-deterministic machine constructs a circuit $D : \{0,1\}^{n_0} \to \{0,1\}$ that, on input $x_0 \in \{0,1\}^{n_0}$, constructs a description of the circuit C (using the fixed guess of C_L that we already have), "hard-wires" x_0 into C to obtain a circuit C_{x_0} that computes the decision of C at x as a function of the random bits of C, and then runs the CAPP algorithm on the description of C_{x_0}. The time it takes to construct D is polynomial in $T(S'(n_0))$, and so our final running time is still bounded by $T(S(O(n_0))^{O(1)})^{O(1)}$. ■

Lastly, we also prove an "infinitely-often" version of Item (1) in Proposition 6.6. Loosely speaking, in this version our assumption is that $\mathcal{E}$ has small circuits infinitely-often, and the corresponding conclusion is that $\mathcal{E}$ can be decided infinitely-often by circuits that can be uniformly constructed by quick non-deterministic machines.

Proposition 6.7 (an "infinitely-often" version of Item (1) in Proposition 6.6)**.** *There exists a constant $k > 1$ and $L \in \mathcal{E}$ such that the following holds. Let $S, T : \mathbb{N} \to \mathbb{N}$ be time-computable functions such that $S(n) \geq n$ and $T(n) \geq n$ for every $n \in \mathbb{N}$. Assume that $L \in'$ io$\mathcal{SIZE}[S]$ and that* $\mathsf{CAPP}[S(k \cdot n)^k] \in pr\mathcal{DTIME}[T]$*. Then, any $L_0 \in \mathcal{DTIME}[2^n]$ can be decided infinitely-often by $\mathcal{NTIME}[T']$-uniform circuits of size $n^k \cdot S(k \cdot n)$, where $T'(n) = T(S(k \cdot n)^k)^k$.*

Proof. We adapt the proof of Item (1) in Proposition 6.6. The only place in which we used the hypothesis that $\mathcal{E}$ has small circuits is in Step 2, where this was applied to the set L from Proposition 6.4. In our current setting we are only guaranteed that there exists an infinite set of input lengths $S \subseteq \mathbb{N}$ such that for every $n'' \in S$ it holds that $L_{n''}$ can be decided by a circuit of size $S(n'')$.

The non-deterministic machine that we use is essentially identical to that in Proposition 6.6, and we wish to claim that it succeeds in non-deterministically constructing circuits for L_0 on infinitely-many input lengths. Recall that on any input length n_0, the machine non-deterministically guesses a circuit C_L for L on input length $n'' = c \cdot n_0$, where $c > 1$ is some universal constant, and deterministically verifies that C_L is "sufficiently good" to support the construction of the larger circuit for L_0 (where "sufficiently good" means that C_L is not a NO instance for the problem Π defined there). The only problem in this approach is that the set $\{c \cdot n_0\}_{n_0 \in \mathbb{N}}$ might not intersect the set S infinitely-many times, in which case we are not guaranteed that a "sufficiently good" circuit C_L exists for infinitely-many n_0's.

This problem can be easily solved using a padding argument. Specifically, instead of the set L, we consider a set L' that for every input length $m \in \mathbb{N}$ satisfies the following: If $m = c \cdot n_0$ for some $n_0 \in \mathbb{N}$ (and c as above) then $L'_m = L_m$, and if $m \in [c \cdot n_0 + 1, c \cdot (n_0 + 1) - 1]$ for some n_0, then L' depends only on the first $c \cdot n_0$ bits of its input, and computes $L_{c \cdot n_0}$. Note that if for some $m \in [c \cdot n_0 + 1, c \cdot (n_0 + 1) - 1]$ there exists a circuit of size $S(m)$ for L'_m, then there exists a circuit of size less than $S(c \cdot (n_0 + 1)) = S(c \cdot n_0 + O(1))$ for $L_{c \cdot n_0}$. Our new hypothesis is therefore that $L' \in \mathtt{i.o.}\mathcal{SIZE}[s]$ (instead of $L \in \mathtt{i.o.}\mathcal{SIZE}[s]$), and in this case, for an infinite set of input lengths in n_0 there exists a circuit for L on input length $c \cdot n_0$ of size $S(n'' + O(1)) = S(O(n_0))$. The overhead in the circuit size only affects the constant hidden inside the O-notation in the definition of S'. ■

6.2 Proof of Theorems 1.4 and 1.5

We first prove Theorem 1.4, which refers to the "low-end" parameter setting and shows an equivalence between circuit lower bounds and non-deterministic derandomization. Then we will prove Theorem 1.5, which handles the "high-end" setting and also extends to an "almost-always" version. In both cases the proofs follow from Propositions 6.6 and 6.7 and from Corollary 3.4, instantiated with specific useful parameters.

We note in advance that the statements generalize to other parameter settings in a straightforward way (for an example of such a generalized statement, see Theorem 6.11).

Theorem 6.8 (Theorem 1.4, restated)**.** *There exists $L \in \mathcal{E}$ such that the following holds. Assume that there exists $\delta > 0$ such that $\mathcal{DTIME}[2^n]$ cannot be decided by $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of polynomial size (for any polynomial). Then, denoting $pr\mathcal{NSUBEXP} = \cap_{\epsilon>0} pr\mathcal{NTIME}[2^{n^\epsilon}]$, we have that*

$$pr\mathcal{BPP} \subset \texttt{i.o.}pr\mathcal{NSUBEXP} \iff L \notin \mathcal{P}/\text{poly} .$$

Proof. The "$\Longleftarrow$" direction follows from [Bab+93]. For the "$\Longrightarrow$" direction, assume that $pr\mathcal{BPP} \subset \texttt{i.o.}pr\mathcal{NSUBEXP}$, which implies that CAPP $\in \texttt{i.o.}pr\mathcal{NTIME}[2^{n^\epsilon}]$ for every $\epsilon > 0$. Fixing an arbitrarily large $c \in \mathbb{N}$, we now prove that $L \notin \mathcal{SIZE}[n^c]$, where $L \in \mathcal{E}$ is the problem from Proposition 6.6.

Indeed, assuming towards a contradiction that $L \in \mathcal{SIZE}[n^c]$, we use Item (2) of Proposition 6.6 with parameters $S(n) = n^c$ and $T(n) = 2^{n^\epsilon}$, where $\epsilon > 0$ is sufficiently small. We conclude that $\mathcal{DTIME}[2^n]$ has $\mathcal{NTIME}[T'']$-uniform circuits of size $n^k \cdot S(k \cdot n) < n^{ck}$, where

$$T''(n) \leq T(S(S(k \cdot n)^k)^k)^k = T(n^{O_{c,k}(1)}) = 2^{n^{O_{c,k}(\epsilon)}} ,$$

which contradicts our hypothesis if $O_{c,k}(\epsilon) < \delta$. ∎

Theorem 6.9 (Theorem 1.5, restated)**.** *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\delta \cdot n}]$-uniform circuits. Then:*

1. *It holds that $pr\mathcal{BPP} \subset \texttt{i.o.}pr\mathcal{P} \iff \exists \epsilon > 0 : \mathcal{DTIME}[2^n] \not\subset \mathcal{SIZE}[2^{\epsilon \cdot n}]$.*
2. *For every fixed $\eta > 0$ it holds that $pr\mathcal{BPP} \subset \texttt{i.o.} \cup_{c \in \mathbb{N}} pr\mathcal{DTIME}[2^{c \cdot (\log n)^{1/\eta}}] \iff \exists \epsilon > 0 : \mathcal{DTIME}[2^n] \not\subset \mathcal{SIZE}[2^{\epsilon \cdot n^\eta}]$.*

Moreover, if we assume that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\delta \cdot n}]$-uniform circuits even infinitely-often, then the two foregoing statements hold with both derandomization and circuit lower bounds that hold "almost-always".

Proof. We first prove Item (1). The "$\Longleftarrow$" direction was proved in [IW99] (and follows from the more general Corollary 3.4). For the "$\Longrightarrow$" direction, assume that CAPP $\in \texttt{i.o.}pr\mathcal{DTIME}[T]$ where $T(n) = n^c$ for some constant $c \in \mathbb{N}$, and assume towards a contradiction that $\mathcal{DTIME}[2^n] \subset \mathcal{SIZE}[S]$ where $S(n) = 2^{\epsilon \cdot n}$ for a sufficiently small constant $\epsilon = \epsilon(c) > 0$. Then, Item (1) of Proposition 6.6 implies that for a universal constant $k \in \mathbb{N}$ it holds that $\mathcal{DTIME}[2^n]$ can be decided by $\mathcal{NTIME}[T']$-uniform circuits, where $T'(n) = T(S(k \cdot n)^k)^k = 2^{\epsilon \cdot (ck^2) \cdot n} < 2^{\delta \cdot n}$, which is a contradiction.

The proof of Item (2) is similar. The "$\Longleftarrow$" follows from Corollary 3.4, instantiated with $S(n) = 2^{\epsilon \cdot n^\eta}$, to deduce that CAPP $\in pr\mathcal{DTIME}[T]$ for $T(n) = 2^{c \cdot S^{-1}(n^c)} = 2^{c/\epsilon^{1/\eta} \cdot (\log n)^{1/\eta}}$. For the "$\Longrightarrow$" direction, we use Item (1) of Proposition 6.6 with $T(n) = 2^{c \cdot (\log n)^{1/\eta}}$ and with $S(n) = 2^{\epsilon \cdot n^\eta}$, where $\epsilon < (\delta/c)^\eta \cdot k^{\eta^2/(1+\eta)}$ is sufficiently small, and rely on the fact that $T'(n) = T(S(k \cdot n)^k)^k < 2^{\delta \cdot n}$.

The "moreover" part follows from the exact same arguments above, just replacing Proposition 6.6 with Proposition 6.7. ∎

6.3 Proof of Theorem 1.6

In this section we prove Theorem 1.6, which asserts that an equivalence between derandomization and circuit lower bounds implies that $\mathcal{E}$ does not have small circuits that can be uniformly constructed by quick non-deterministic machines. In fact, as mentioned in Section 2.3, we will prove the stronger conclusion that $\mathcal{E} \not\subseteq \mathcal{NTIME}[T] \cap co\mathcal{NTIME}[T]$ for a small function T.

The first step towards proving the theorem is the following easy claim, which asserts that if $\mathcal{E} \subseteq \mathcal{NTIME}[T] \cap co\mathcal{NTIME}[T]$ for a small function T (and in particular if $\mathcal{E}$ has $\mathcal{NTIME}[T]$-uniform circuits) then we can non-deterministically find a "hard" function, and then use this function to instantiate the pseudorandom generator of Umans [Uma03] and derandomize $pr\mathcal{BPP}$.

Claim 6.10 (collapse of $\mathcal{E}$ to small non-deterministic time implies non-deterministic derandomization). *Let $c \in \mathbb{N}$ be the constant from Theorem 3.3, and let $T : \mathbb{N} \to \mathbb{N}$ be time-computable. Assume that $\mathcal{E} \subseteq \mathcal{NTIME}[T] \cap co\mathcal{NTIME}[T]$. Then,* CAPP $\in$ *$pr\mathcal{NTIME}[T']$, where $T' = T(n^c) \cdot \mathrm{poly}(n)$.*

Proof. Let A be an algorithm that gets input 1^n, and for $\ell = c \cdot \log(n)$ constructs the truth-table of the lexicographically-first Boolean function over ℓ bits that cannot be computed by circuits of size n. Note that this can be done deterministically in time polynomial in $2^{2^\ell} \cdot 2^{\tilde{O}(n)} = 2^{O(n^c)}$. Therefore, applying our assumption to a padded input of length n^c,[35] there exists a non-deterministic machine that runs in time $\mathrm{poly}(n) \cdot T(n^c)$, for some non-deterministic guesses constructs this truth-table, and for all other non-deterministic guesses aborts. Feeding this truth-table into the PRG from Theorem 3.3, we can solve CAPP in time $\mathrm{poly}(n) \cdot T(n^c)$. ∎

[35]Specifically, we consider the set L of triplets $(1^n, 1^{n^c}, x)$ such that $x \in \{0,1\}^\ell$ and the lexicographically-first Boolean function over ℓ bits that cannot be computed by circuits of size n accepts x. Since $\mathcal{L} \in \mathcal{E}$, we have that $\mathcal{L} \subseteq \mathcal{NTIME}[T] \cap co\mathcal{NTIME}[T]$. Therefore, we can use the non-deterministic machine for L to construct the truth-table bit-by-bit, for all values of $x \in \{0,1\}^\ell$.

We now prove Theorem 1.6. As mentioned in the introduction, we actually prove a stronger version, which shows tighter two-way implications between the statement "derandomization and lower bounds are equivalent" and the statement "$\mathcal{E}$ does not have $\mathcal{NTIME}[T]$-uniform circuits".

Towards proving the result, we define a class of growth functions "in between" quasipolynomial functions and subexponential functions. Specifically, for every two constants $k, c \in \mathbb{N}$, we denote by $\mathsf{e}^{(k,c)} : \mathbb{N} \to \mathbb{N}$ the function that applies k logarithms to its input, raises the obtained expression to the power c, and then takes k exponentiations of this expression. For example, $\mathsf{e}^{(1,c)}(n) = 2^{(\log n)^c}$ and $\mathsf{e}^{(2,c)}(n) \in 2^{2^{\mathrm{polyloglog}(n)}}$. Note that $\mathsf{e}^{(k+1,c)}$ grows asymptotically faster than $\mathsf{e}^{(k,c')}$ for any constants c, c', and that $\mathsf{e}^{(k,c)}$ is smaller than any subexponential function. Then, we have that:

Theorem 6.11 (Theorem 1.6, a tighter version)**.** *For any constant $k \in \mathbb{N}$ we have that:*

$$\exists \delta > 0 : \mathcal{DTIME}[2^n] \text{ does not have } \mathcal{NTIME}[T]\text{-uniform circuits, for } T = 2^{\mathsf{e}^{(k,\delta)}} \tag{6.2}$$

$$\Downarrow$$

$$pr\mathcal{BPP} \subset \cap_{\epsilon>0}\mathtt{i.o.}pr\mathcal{NTIME}[2^{\mathsf{e}^{(k,\epsilon)}}] \iff \mathcal{DTIME}[2^n] \not\subset \cup_{c_0 \in \mathbb{N}}\mathcal{SIZE}[\mathsf{e}^{(k,c_0)}] \tag{6.3}$$

$$\Downarrow$$

$$\forall c_0 \in \mathbb{N}, \mathcal{DTIME}[2^n] \text{ does not have } \mathcal{NTIME}[T]\text{-uniform circuits, for } T(n) = \mathsf{e}^{(k,c_0)} \tag{6.4}$$

that is, statement (6.2) implies statement (6.3), which in turn implies statement (6.4).

We stress that the gap between the values of T in statements (6.2) and (6.4) is substantial, but nevertheless much smaller than an exponential gap. This is since in statement (6.2) the hypothesis is for T that is exponential in $\mathsf{e}^{(k,\delta)}$ where $\delta > 0$ is an *arbitrarily small constant*, whereas in statement (6.4) the conclusion is for $T = \mathsf{e}^{(k,c_0)}$ where c_0 is an *arbitrarily large constant*. For example, for $k = 1$ this is the difference between quasipolynomial functions and functions of the form $2^{2^{(\log n)^\epsilon}} \ll 2^{n^\epsilon}$.

Proof of Theorem 6.11. To see that statement (6.2) implies statement (6.3), first observe that for any two constants $c, c' \in \mathbb{N}$ it holds that $(\mathsf{e}^{(k,c)})^{-1}(n) = \mathsf{e}^{(k,1/c)}(n)$ and that $\mathsf{e}^{(k,c)}(\mathsf{e}^{(k,c')}(n)) = \mathsf{e}^{(k,cc')}(n)$. The "$\Longleftarrow$" then follows from Corollary 3.4, instantiated with $S(n) = \mathsf{e}^{(k,c_0)}$, to deduce that CAPP $\in pr\mathcal{DTIME}[T]$ for $T(n) = 2^{c \cdot S^{-1}(n^c)} = 2^{c \cdot \mathsf{e}^{(k,1/c_0)}(n^c)} < 2^{\mathsf{e}^{(k,2/c_0)}}$. For the "$\Longrightarrow$" direction we use Item (2) of Proposition 6.6 with $S(n) = \mathsf{e}^{(k,c_0)}$ and with $T(n) = 2^{\mathsf{e}^{(k,\epsilon)}}(n)$ for a sufficiently small $\epsilon > 0$, and rely on the fact that $T'(n) = T(S(k \cdot n)^k)^k < T(\mathsf{e}^{(k,2c_0)}(n)) = 2^{\mathsf{e}^{(k,2\epsilon c_0)}(n)}$.

To see that statement (6.3) implies statement (6.4), assume towards a contradiction that for some $c_0 \in \mathbb{N}$ it holds that $\mathcal{DTIME}[2^n]$ has $\mathcal{NTIME}[T]$-uniform circuits, where $T(n) = \mathsf{e}^{(k,c_0)}(n)$. Using Claim 6.10, we have that CAPP $\in pr\mathcal{NTIME}[T(n^c) \cdot$

$n^c]$, and hence $pr\mathcal{BPP} \subseteq \cup_{c\in\mathbb{N}} pr\mathcal{NTIME}\left[\mathsf{e}^{(k,c)}\right] \subseteq \cap_{\epsilon>0} pr\mathcal{NTIME}\left[2^{\mathsf{e}^{(k,\epsilon)}}\right]$. By our hypothesis it now follows that $\mathcal{DTIME}[2^n] \not\subseteq \cup_{c_0\in\mathbb{N}} \mathcal{SIZE}\left[\mathsf{e}^{(k,c_0)}\right]$, which is a contradiction. ■

Acknowledgements

The authors are grateful to Igor Oliveira for pointing them to the results in [OS17, Sec. 5], which serve as a basis for the proof of Theorem 1.3. The authors thank Oded Goldreich, who provided feedback throughout the research process and detailed comments on the manuscript, both of which helped improve the work. We also thank Ryan Williams for a helpful discussion, for asking us whether a result as in Theorem 1.3 can be proved, and for feedback on the manuscript. The work was initiated in the 2018 Complexity Workshop in Oberwolfach; the authors are grateful to the Mathematisches Forschungsinstitut Oberwolfach and to the organizers of the workshop for the productive and lovely work environment.

Lijie Chen is supported by NSF CCF-1741615 and a Google Faculty Research Award. Ron Rothblum is supported in part by a Milgrom family grant, by the Israeli Science Foundation (Grant No. 1262/18), and the Technion Hiroshi Fujiwara cyber security research center and Israel cyber directorate. Roei Tell is supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702). Eylon Yogev is visiting the Simons Institute for the Theory of Computing.

References

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

[Abb+16] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. "Simulating branching programs with edit distance and friends". In: *Proc. 48th Annual ACM Symposium on Theory of Computing (STOC)*. 2016, pp. 375–388.

[ACR98] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. "A new general derandomization method". In: *Journal of the ACM* 45.1 (1998), pp. 179–213.

[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.

[Bab+93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. "BPP has subexponential time simulations unless EXPTIME has publishable proofs". In: *Computational Complexity* 3.4 (1993), pp. 307–318.

[BF99] Harry Buhrman and Lance Fortnow. "One-Sided Versus Two-Sided Error in Probabilistic Computation". In: *Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS)*. 1999, pp. 100–109.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. "Nonrelativizing separations". In: *Proc. 13th Annual IEEE Conference on Computational Complexity (CCC)*. 1998, pp. 8–12.

[BG81] Charles H. Bennett and John Gill. "Relative to a random oracle A, $\mathbf{P}^A \neq \mathbf{NP}^A \neq \text{co} - \mathbf{NP}^A$ with probability 1". In: *SIAM Journal of Computing* 10.1 (1981), pp. 96–113.

[BS+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Robust PCPs of proximity, shorter PCPs and applications to coding". In: *SIAM Journal of Computing* 36.4 (2006), pp. 889–974.

[BS+17] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. "Interactive oracle proofs with constant rate and query complexity". In: *Proc. 44th International Colloquium on Automata, Languages and Programming (ICALP)*. 2017, Art. No. 40, 15.

[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.

[Car+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. "Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility". In: *Proc. 7th Conference on Innovations in Theoretical Computer Science (ITCS)*. 2016, pp. 261–270.

[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.

[Che+19] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. "Relations and Equivalences Between Circuit Lower Bounds and Karp-Lipton Theorems". In: *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 2019, 30:1–30:21.

[CIS18] Marco L. Carmosino, Russell Impagliazzo, and Manuel Sabin. "Fine-grained derandomization: from problem-centric to resource-centric complexity". In: *Proc. 45th International Colloquium on Automata, Languages and Programming (ICALP)*. 2018, Art. No. 27, 16.

[CNS99] Jin-Yi Cai, Ajay Nerurkar, and D. Sivakumar. "Hardness and hierarchy theorems for probabilistic quasi-polynomial time". In: *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC))*. 1999, pp. 726–735.

[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.

[Del+14] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. "Exponential time complexity of the permanent and the Tutte polynomial". In: *ACM Transactions on Algorithms* 10.4 (2014), Art. 21, 32.

[DR06] Irit Dinur and Omer Reingold. "Assignment testers: towards a combinatorial proof of the PCP theorem". In: *SIAM Journal of Computing* 36.4 (2006), pp. 975–1024.

[FK09] Lance Fortnow and Adam R. Klivans. "Efficient learning algorithms yield circuit lower bounds". In: *Journal of Computer and System Sciences* 75.1 (2009), pp. 27–36.

[GL89] Oded Goldreich and Leonid A. Levin. "A Hard-core Predicate for All One-way Functions". In: *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*. 1989, pp. 25–32.

[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.

[GR17] Oded Goldreich and Guy N. Rothblum. "Worst-case to Average-case reductions for subclasses of P". In: *Electronic Colloquium on Computational Complexity: ECCC* 26 (2017), p. 130.

[GS89] Yuri Gurevich and Saharon Shelah. "Nearly linear time". In: *Logic at Botik, Symposium on Logical Foundations of Computer Science*. Lecture Notes in Computer Science. 1989, pp. 108–118.

[GSTS03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. "Uniform hardness versus randomness tradeoffs for Arthur-Merlin games". In: *Computational Complexity* 12.3-4 (2003), pp. 85–130.

[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.

[GVW11] Oded Goldreich, Salil Vadhan, and Avi Wigderson. "Simplified derandomization of BPP using a hitting set generator". In: *Studies in complexity and cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, Heidelberg, 2011, pp. 59–67.

[HH13] Ryan C. Harkins and John M. Hitchcock. "Exact learning algorithms, betting games, and circuit lower bounds". In: *ACM Transactions on Computation Theory* 5.4 (2013), Art. 18, 11.

[HR03] Tzvika Hartman and Ran Raz. "On the distribution of the number of roots of polynomials and explicit weak designs". In: *Random Structures & Algorithms* 23.3 (2003), pp. 235–263.

[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.

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the complexity of k-SAT". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 367–375.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which problems have strongly exponential complexity?" In: *Journal of Computer and System Sciences* 63.4 (2001), pp. 512–530.

[IW98] R. Impagliazzo and A. Wigderson. "Randomness vs. Time: De-Randomization Under a Uniform Assumption". In: *Proc. 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1998, pp. 734–.

[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.

[JS12] Maurice Jansen and Rahul Santhanam. "Stronger lower bounds and randomness-hardness trade-offs using associated algebraic complexity classes". In: *Proc. 29th Symposium on Theoretical Aspects of Computer Science (STACS)*. Vol. 14. 2012, pp. 519–530.

[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.

[KKO13] Adam Klivans, Pravesh Kothari, and Igor Oliveira. "Constructing Hard Functions Using Learning Algorithms". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 2013, pp. 86–97.

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. "Lower bounds based on the exponential time hypothesis". In: *Bulletin of the European Association for Theoretical Computer Science (EATCS)* 105 (2011), pp. 41–71.

[Lun+92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic methods for interactive proof systems". In: *Journal of the Association for Computing Machinery* 39.4 (1992), pp. 859–868.

[Mie09] Thilo Mie. "Short PCPPs verifiable in polylogarithmic time with $O(1)$ queries". In: *Annals of Mathematics and Artificial Intelligence* 56.3-4 (2009), pp. 313–338.

[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.

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

[OS17] Igor C. Oliveira and Rahul Santhanam. "Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness". In: *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 79. 2017, Art. No. 18, 49.

[PF79] Nicholas Pippenger and Michael J. Fischer. "Relations among complexity measures". In: *Journal of the ACM* 26.2 (1979), pp. 361–381.

[RZR19] Noga Ron-Zewi and Ron Rothblum. "Local Proofs Approaching the Witness Length". In: *Electronic Colloquium on Computational Complexity: ECCC* 26 (2019), p. 127.

[San09] Rahul Santhanam. "Circuit lower bounds for Merlin-Arthur classes". In: *SIAM Journal of Computing* 39.3 (2009), pp. 1038–1061.

[Sha92] Adi Shamir. "IP = PSPACE". In: *Journal of the ACM* 39.4 (1992), pp. 869–877.

[Sho90] Victor Shoup. "New algorithms for finding irreducible polynomials over finite fields". In: *Mathematics of Computation* 54.189 (1990), pp. 435–447.

[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.

[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.

[Tel19] Roei Tell. "Proving that $pr\mathcal{BPP} = pr\mathcal{P}$ is as hard as proving that "almost $\mathcal{NP}$" is not contained in $\mathcal{P}/\text{poly}$". In: *Information Processing Letters* 152 (2019), p. 105841.

[TV07] Luca Trevisan and Salil P. Vadhan. "Pseudorandomness and Average-Case Complexity Via Uniform Reductions". In: *Computational Complexity* 16.4 (2007), pp. 331–364.

[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.

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

[Wil15] Virginia V. Williams. "Hardness of easy problems: basing hardness on popular conjectures such as the Strong Exponential Time Hypothesis". In: *Proc. 10th International Symposium on Parameterized and Exact Computation*. Vol. 43. 2015, pp. 17–29.

[Wil16] Richard Ryan Williams. "Strong ETH breaks with Merlin and Arthur: short non-interactive proofs of batch evaluation". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 50. 2016, Art. No. 2, 17.

[Wil18] Virginia Vassilevska Williams. *On some fine-grained questions in algorithms and complexity*. Accessed at https://people.csail.mit.edu/virgi/eccentri.pdf, October 17, 2019. 2018.

[Woe03] Gerhard J. Woeginger. "Exact algorithms for NP-hard problems: a survey". In: *Combinatorial optimization—Eureka, you shrink!* Vol. 2570. Lecture Notes in Computer Science. Springer, Berlin, 2003, pp. 185–207.

A Low-degree polynomials are sample-aided worst-case to average-case reducible

Recall that in Section 4.1 we defined the notion of sample-aided worst-case to δ-average-case-reducible function (see Definitions 4.2 and 4.3), following [GR17]. In this appendix we explain why labeled samples can be helpful for uniform worst-case to "rare-case" reductions, and show that low-degree polynomials are indeed sample-aided worst-case to average-case-reducible.

Consider a function f whose truth-table is a codeword of a locally list-decodable code, and also assume that f is randomly self-reducible (i.e., computing f in the worst-case is reducible to computing f on, say, .99 of the inputs). Then, for every circuit $\tilde{C}$ that agrees with f on a tiny fraction of inputs (i.e., $\tilde{C}$ computes a "corrupt" version of f), we can efficiently produce a small list of circuits with oracle gates to $\tilde{C}$ such that one of these circuits correctly computes f on all inputs. The main trouble is that we don't know which candidate circuit in this list to use. This is where the labeled samples come in: We can iterate over the candidates in the list, use the labeled samples to *test* each candidate circuit for agreement with f, and with high probability find a circuit that agrees with f on (say) .99 of the inputs. Then, using the random self-reducibility of f, we obtain a circuit that correctly computes f on each input, with high probability.

The crucial property that we need from the code in order to make the foregoing algorithmic approach work is that the local list-decoding algorithm will *efficiently* produce a relatively *short* list. Specifically, recall that by our definition, a sample-aided worst-case to δ-average-case reduction needs to run in time $\text{poly}(1/\delta)$. Hence, we need a list-decoding algorithm that runs in time $\text{poly}(1/\delta)$ (and indeed produces a list of such size). A suitable local list-decoding algorithm indeed exists in the case that the code is the Reed-Muller code, which leads us to the following result:

Proposition A.1 (low-degree polynomials are uniformly worst-case to average-case reducible with a self-oracle). *Let $q : \mathbb{N} \to \mathbb{N}$ be a field-size function, let $\ell : \mathbb{N} \to \mathbb{N}$ such that $n \geq \ell \cdot \log(q)$, and let $d, \rho : \mathbb{N} \to \mathbb{N}$ such that $10\sqrt{d(n)/q(n)} \leq \rho(n) \leq (q(n))^{-\Omega(1)} = o(1)$. Let $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ be a sequence of functions such that f_n computes a polynomial $\mathbb{F}_n^{\ell(n)} \to \mathbb{F}_n$ of degree $d(n)$ where $|\mathbb{F}_n| = q(n)$. Then f is sample-aided worst-case to ρ-average-case reducible.*

Proof. We construct a probabilistic machine M that gets input 1^n, and oracle access to a function $\widetilde{f_n}$ that agrees with f_n on $\rho(n)$ of the inputs, and also $\text{poly}(1/\rho(n))$ labeled samples for f_n, and with probability $1 - \rho(n)$ outputs a circuit $C : \mathbb{F}^\ell \to \mathbb{F}$ such that for every $x \in \mathbb{F}^\ell$ it holds that $\Pr_r[C^{\widetilde{f_n}}(x, r) = f_n(x)] \geq 2/3$.

The first step of the machine M is to invoke the local list-decoding algorithm of [STV01, Thm 29], instantiated with degree parameter $d = d(n)$ and agreement parameter $\rho = \rho(n)$. The algorithm runs in time $\mathrm{poly}(\ell(n), d, \log(q(n)), 1/\rho) = \mathrm{poly}(n, 1/\rho)$ and outputs a list of $O(1/\rho)$ probabilistic oracle circuits $C_1, ..., C_{O(1/\rho)} : \{0,1\}^n \to \{0,1\}^n$ such that with probability at least $2/3$ there exists $i \in [O(1/\rho)]$ satisfying $\Pr[C_i^{\widetilde{f_n}}(x) = f_n(x)] \geq 2/3$ for all $x \in \{0,1\}^n$. We call any circuit that satisfies the latter condition good. By invoking the algorithm of [STV01] for $\mathrm{poly}(1/\rho)$ times, we obtain a list of $t = \mathrm{poly}(1/\rho)$ circuits $C_1, ..., C_t$ such that with probability at least $1 - \mathrm{poly}(\rho)$ there exists $i \in [t]$ such that C_i is good.

The second step of the machine is to transform the probabilistic circuits into deterministic circuits such that, with high probability, the deterministic circuit corresponding to the "good" circuit C_i will correctly compute f_n on .99 of the inputs (when given oracle access to $\widetilde{f_n}$). Specifically, by implementing naive error-reduction in all circuits, we can assume that for every $x \in \mathbb{F}^\ell$ it holds that $\Pr_r[C_i^{\widetilde{f_n}}(x, r) = f_n(x)] \geq .995$. Now the machine M creates $O(\log(1/\rho))$ copies of each circuit in the list, and for each copy M "hard-wires" a randomly-chosen fixed value for the circuit's randomness. The result is a list of $t' = \mathrm{poly}(1/\rho)$ *deterministic* circuits $D_1, ..., D_{t'}$ such that with probability $1 - \mathrm{poly}(\rho)$ there exists a circuit D_i satisfying $\Pr_x[D_i^{\widetilde{f_n}}(x) = f_n(x)] \geq .99$.

The third step of the machine M is to "weed" the list in order to find a single circuit D_i that (when given access to $\widetilde{f_n}$) correctly computes f on .95 of the inputs. To do so M iterates over the list, and for each circuit D_j estimates the agreement of $D_j^{\widetilde{f_n}}$ with f_n with error .01 and confidence $1 - \mathrm{poly}(\rho)$, using the random samples.

The final step of the machine M is to use the standard random self-reducibility of the Reed-Muller code to transform the circuit D_i into a probabilistic circuit that correctly computes f at each input with probability at least $2/3$. Specifically, the probabilistic circuit implements the standard random self-reducibility algorithm for the (q, ℓ, d) Reed-Muller code (see, e.g., [AB09, Thm 19.19]), while resolving its oracle queries using the circuit D_i. The standard algorithm runs in time $\mathrm{poly}(q, \ell, d)$, and works whenever D_i agrees with f_n on at least $1 - \frac{1-d/q}{6} < .95 + d/q$ of the inputs, which holds in our case since $d/q < \delta = o(1)$. ■

B When even Merlin and Arthur encounter difficulties

As mentioned in Section 1.4, the assumption MAETH can be easily shown to imply strong circuit lower bounds and derandomization of $pr\mathcal{BPP}$ (and thus also of $pr\mathcal{MA}$). Specifically, the following more general (i.e., parametrized) Theorem B.1 relies on a standard Karp-Lipton-style argument, which originates in [Bab+93] and also underlies the proof of Proposition 6.5.

We note in advance that after the proof of Theorem B.1 we prove another result, which shows a very different tradeoff between $\mathcal{MA}$ lower bounds (specifically, lower bounds for fixed-polynomial-time verifiers) and derandomization.

Theorem B.1 (lower bounds for $\mathcal{MA}$ algorithms imply non-uniform circuit lower bounds)**.** *There exists $L \in \mathcal{E}$ and a constant $k > 1$ such that for any time-computable function $S : \mathbb{N} \to \mathbb{N}$ such that $S(n) \geq n$ the following holds. Assume that $DTIME[2^n] \not\subseteq MATIME[S']$, where $S'(n) = S(k \cdot n)^k$. Then, $L \notin SIZE[S]$.*

Note that, using Corollary 3.4, under the hypothesis of Theorem B.1 we have that $\mathsf{CAPP} \in \mathtt{i.o.}prDTIME[T]$, where $T(n) = 2^{O(S^{-1}(n^{O(1)}))}$. In particular, under MAETH (which refers to $S(n) = 2^{\Omega(n/\log(n))}$) we have that $pr\mathcal{BPP} \subseteq \mathtt{i.o.}prDTIME[n^{O(\log\log(n))}]$.

Proof of Theorem B.1. Let L be the problem L from Proposition 6.4. Assuming towards a contradiction that $L \in SIZE[S]$, we show that $DTIME[2^n] \subseteq MATIME[S']$.

Let $L_0 \in DTIME[2^n]$. We construct a probabilistic verifier that gets input $x_0 \in \{0,1\}^{n_0}$, and if $x_0 \in L_0$ then for some non-deterministic choices the verifier accepts with probability one, and if $x_0 \notin L_0$ then for all non-deterministic choices the verifier rejects, with high probability. The verifier first reduces L_0 to L, by computing $x \in \{0,1\}^n$ of length $n = O(n_0)$ such that $x_0 \in L_0$ if and only if $x \in L$.

Let $n' = \ell(n) = O(n) = O(n_0)$. By our hypothesis, there exists a circuit over n' input bits of size $S(n')$ that decides $L_{n'}$. The verifier guesses a circuit $C_L : \{0,1\}^{n'} \to \{0,1\}$ of size $S(n')$, and simulates the machine M from Proposition 6.4 on input x, while resolving its oracle queries of using C_L. The verifier accepts if and only if M accepts. Note that if $x_0 \in L_0$ and the verifier's guess was correct (i.e., C_L decides $L_{n'}$), then the verifier accepts with probability one. On the other hand, if $x_0 \notin L_0$, then for every guess of C_L (i.e., every oracle for M) the verifier rejects, with high probability. The running time of the verifier is $\text{poly}(n) \cdot \text{poly}(S(n')) = S(O(n))^{O(1)}$. ■

The following result uses a hypothesis that is intuitively much weaker than MAETH, namely that there exists a function in $\mathcal{P}$ that cannot be solved by any $\mathcal{MA}$ verifier that runs in fixed polynomial time (i.e., in time n^k for some constant $k \in \mathbb{N}$). As a consequence, we obtain a derandomization of $pr\mathcal{BPP}$ either in sub-exponential time, or in polynomial time but with $n^{.01}$ bits of non-uniform advice.[36]

Theorem B.2 (fixed-polynomial-size lower bounds for $\mathcal{MA} \implies$ derandomization and circuit lower bounds)**.** *Assume that for every $k \in \mathbb{N}$ it holds that $\mathcal{P} \not\subseteq \mathtt{i.o.}MATIME[n^k]$. Then, for every $\epsilon > 0$ it holds that $pr\mathcal{BPP} \subseteq (pr\mathcal{P}/n^{\epsilon} \cap prDTIME[2^{n^{\epsilon}}])$.*

Proof. In high-level, we want to use our hypothesis to deduce that there exists a polynomial-time algorithm that outputs the truth-table of a "hard" function, and then use that "hard" function for derandomization. Loosely speaking, the following claim,

[36]Recall that, by Adleman's theorem [Adl78; BG81], we can derandomize $pr\mathcal{BPP}$ with $\text{poly}(n)$ bits of non-uniform advice (and even with $O(n)$ bits, using Theorem 3.6). However, it is not known how to derandomize $pr\mathcal{BPP}$ with $o(n)$ bits of non-uniform advice.

whose proof is a refinement of on an argument from [Che+19], asserts that if the output string of every polynomial-time algorithm has circuit complexity at most n^k, then all of $\mathcal{P}$ can be decided by $\mathcal{MA}$ verifiers running in time $n^{O(k)}$.

Claim B.2.1. *Assume that there exists $k \in \mathbb{N}$ such that for every deterministic polynomial-time machine M there exists an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ the following holds: For every $x \in \{0,1\}^n$, when the output string $M(x)$ is viewed as a truth-table of a function, this function has circuit complexity at most n^k. Then, $\mathcal{P} \subseteq \texttt{i.o.}MATIME[n^{O(k)}]$.*

Proof. Let $L \in \mathcal{P}$, and let M be a polynomial-time machine that decides L. Our goal is to decide L in $MATIME[n^k]$ on infinitely-many input lengths.

For every $x \in \{0,1\}^n$, let $T_x : \{0,1\}^{\text{poly}(n)} \to \{0,1\}$ be a polynomial-sized circuit that gets as input a string Π, and accepts if and only if Π is the computational history of $M(x)$ and $M(x) = 1$. Note that the mapping of $x \mapsto T_x$ can be computed in polynomial time (since M runs in polynomial time). Also, fix a PCP system for CircuitSAT with the following properties: The verifier runs in polynomial time and uses $O(\log(n))$ randomness and $O(1)$ queries; the verifier has perfect completeness and soundness error $1/3$; and there is a polynomial-time algorithm W that maps any circuit C and a satisfying assignment for C (i.e., $y \in C^{-1}(1)$) to a PCP proof that the verifier accepts. For every $x \in \{0,1\}^n$ and every input $\Pi \in \{0,1\}^{\text{poly}(n)}$ for T_x, let $W(T_x, \Pi)$ be the corresponding PCP proof that W produces.

Observe that there is a polynomial-time algorithm A that gets as input $x \in \{0,1\}^n$, produces the computational history of $M(x)$, which we denote by $H_{M(x)}$, produces the circuit T_x, and finally prints the PCP witness $W(T_x, H_{M(x)})$. Thus, by our hypothesis, there exists an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ and every $x \in \{0,1\}^n$ there exists a circuit $C_x : \{0,1\}^{O(\log(n))} \to \{0,1\}$ of size n^k whose truth-table is $W(T_x, H_{M(x)})$.

The $\mathcal{MA}$ verifier V gets input x, and expects to get as proof a circuit $C : \{0,1\}^{O(\log(n))} \to \{0,1\}$ bits. The verifier V now simulates the PCP verifier, while resolving its queries to the PCP using the circuit C. Note that for every $n \in S$ and every $x \in \{0,1\}^n$ the following holds: If $M(x) = 1$ then there exists a proof (i.e., a circuit C_x) such that the verifier accepts with probability one; on the other hand, if $M(x) = 0$, then T_x rejects all of its inputs, which implies that for every proof, with probability at least $2/3$ the $\mathcal{MA}$ verifier rejects. □

Using our hypothesis that for every $k \in \mathbb{N}$ it holds that $\mathcal{P} \not\subseteq \texttt{i.o.}MATIME[n^k]$, and taking the counter-positive of Claim B.2.1, we deduce that:

Corollary B.2.2. *For every $k \in \mathbb{N}$ there exists a polynomial-time machine M such that for every sufficiently large $n \in \mathbb{N}$ there exists an input $x \in \{0,1\}^n$ such that $M(x)$ is the truth-table of a function with circuit complexity more than n^k.*

Now, fix $\epsilon > 0$, let $L \in pr\mathcal{BPP}$, and let R be a probabilistic polynomial-time machine that decides L. Given input $x \in \{0,1\}^n$, we decide whether $x \in L$ in polynomial-time and with n^ϵ advice, as follows. Consider the circuit R_x that computes the decision of R at x as a function of the random coins of R, and let $c > 1$ such that the size of R_x is at most n^c. We instantiate Corollary B.2.2 with $k = c'/\epsilon$, where $c' > c$ is a sufficiently large constant. We expect as advice an input y of length n^ϵ to the machine M such that $M(y)$ has circuit complexity $n^{c'}$. We then use $M(y)$ to instantiate Theorem 3.3 with seed length $O(\log(n))$ and error $1/10$ and for circuits of size n^c (such that the PRG "fools" the circuit R_x), and enumerate its seeds to approximate the acceptance probability of R_x (and hence decide whether or not $x \in L$).

We now also show that $L \in prDTIME[2^{n^{2\epsilon}}]$. To do so, consider the foregoing algorithm, and assume that it gets no advice. Instead, it enumerates over all 2^{n^ϵ} possible advice strings to obtain 2^{n^ϵ} truth-tables, each of size poly(n). We know that at least one of these truth-tables has circuit complexity $n^{c'}$. Now the algorithm constructs the truth-table of a function f over $n^\epsilon + O(\log(n))$ bits, which uses the first n^ϵ bits to "choose" one of the 2^{n^ϵ} truth-tables, and uses the $O(\log(n))$ bits as an index to an entry in that truth-table (i.e., for $i \in \{0,1\}^{n^\epsilon}$ and $z \in O(\log(n))$ it holds that $f(i,z) = g_i(z)$, where g_i is the function that is obtained from the i^{th} advice string). Note that, since at least one of the 2^{n^ϵ} functions had circuit complexity $n^{c'}$, it follows that f also has circuit complexity $n^{c'}$. Thus, this algorithm can use f to instantiate Theorem 3.3 with seed length $n^\epsilon + O(\log(n))$ and for circuits of size n^c to "fool" the circuit R_x. ■