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

LIJIE CHEN, Miller Institute for Basic Research in Science at University of California, Berkeley, USA
RON D. ROTHBLUM, Technion, Israel
ROEI TELL, Institute for Advanced Study and DIMACS, USA
EYLON YOGEV, Bar-Ilan University, Israel

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 (uniform) 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 $co\mathcal{NP} \not\subseteq \mathcal{NP}$) have also been introduced, and have become widely influential.

In this work, we focus on the interaction of exponential-time hypotheses with the fundamental and closely-related questions of *derandomization and circuit lower bounds*. We show that even relatively-mild variants of exponential-time hypotheses have far-reaching implications to derandomization, circuit lower bounds, and the connections between the two. Specifically, we prove that:

(1) The Randomized Exponential-Time Hypothesis (rETH) implies that $\mathcal{BPP}$ can be simulated on "average-case" in *deterministic (nearly-)polynomial-time* (i.e., in time $2^{\tilde{O}(\log(n))} = n^{\log\log(n)^{O(1)}}$). The derandomization relies on a conditional construction of a pseudorandom generator with *near-exponential stretch* (i.e., with seed length $\tilde{O}(\log(n))$); this significantly improves the state-of-the-art in uniform "hardness-to-randomness" results, which previously only yielded pseudorandom generators with sub-exponential stretch from such hypotheses.

(2) The Non-Deterministic Exponential-Time Hypothesis (NETH) implies that derandomization of $\mathcal{BPP}$ is *completely equivalent* to circuit lower bounds against $\mathcal{E}$, and in particular that pseudorandom generators are necessary for derandomization. In fact, we show that the foregoing equivalence follows from a *very weak version of* NETH, and we also show that this very weak version is necessary to prove a slightly stronger conclusion that we deduce from it.

Lastly, 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 $\mathrm{poly}(n)$ can be solved by *probabilistic algorithms* in time $2^{n/\mathrm{polylog}(n)}$, then $\mathcal{BPE}$ does not have circuits of quasilinear size.

CCS Concepts: • **Theory of computation → Pseudorandomness and derandomization**; **Complexity classes**; **Circuit complexity**.

Additional Key Words and Phrases: Exponential-time Hypothesis, Derandomization, Circuit Lower Bounds

1 INTRODUCTION

The Exponential-Time Hypothesis (ETH), introduced by Impagliazzo and Paturi [31] (and refined in [32]), conjectures that 3-SAT with n variables and $m = O(n)$ clauses cannot be deterministically solved in time less than $2^{\epsilon \cdot n}$

Authors' addresses: Lijie Chen, lijiechen@berkeley.edu, Miller Institute for Basic Research in Science at University of California, Berkeley, Berkeley, CA, USA; Ron D. Rothblum, rothblum@cs.technion.ac.il, Technion, Haifa, Israel; Roei Tell, roeitell@gmail.com, Institute for Advanced Study and DIMACS, Princeton, New Jersey, USA; Eylon Yogev, eylon.yogev@biu.ac.il, Bar-Ilan University, Ramat Gan, Israel.

0004-5411/2023/4-ART $15.00
https://doi.org/10.1145/3593581

(for a constant $\epsilon = \epsilon_{m/n} > 0$). The ETH may be viewed as an "exponentially-strong" version of $\mathcal{P} \neq \mathcal{NP}$, since it conjectures that a specific $\mathcal{NP}$-complete problem requires essentially exponential time to solve.

Since the introduction of ETH many related variants, which are also "exponentially-strong" versions of classical complexity-theoretic conjectures, have also been introduced. For example, the Randomized Exponential-Time Hypothesis (rETH), introduced in [15], 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 [7], 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 [61][1]), and other variations conjecture similar lower bounds for seemingly-harder problems (e.g., for #3SAT; see [15]).

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; see, e.g., the surveys [40, 62–64].

Exponential-time hypotheses focus on conjectured lower bounds for *uniform algorithms*. Two other fundamental questions in theoretical computer science are those of *derandomization*, which refers to the power of probabilistic algorithms; and of *circuit lower bounds*, which refers to the power of *non-uniform* circuits. Despite the central place of all three questions, the interactions of exponential-time hypotheses with derandomization and circuit lower bounds have yet to be systematically studied.

1.1 Our results: Bird's eye

In this work we focus on *the interactions between exponential-time hypotheses, derandomization, and circuit lower bounds*. In a nutshell, our main contribution is showing that:

> **Even relatively mild variants of exponential-time hypotheses have far-reaching consequences for derandomization and circuit lower bounds.**

Let us now give a brief overview of our specific results, before describing them in more detail in Sections 1.2, 1.3, and 1.4. Our two main results are the following:

(1) We show that rETH implies a *nearly-polynomial-time* average-case derandomization of $\mathcal{BPP}$. Specifically, assuming rETH, we show that $\mathcal{BPP}$ can be decided, in average-case and on infinitely many input lengths, by deterministic algorithms that run in time $n^{\log\log(n)^{O(1)}}$ (see Theorem 1.1). This significantly improves the state-of-the-art in the long line of *uniform "hardness-to-randomness"* results.

(2) A classical open question is whether worst-case derandomization of $\mathcal{BPP}$ *requires* pseudorandom generators. We show that a weak version of NETH yields a positive answer to this question; specifically, it suffices to assume that $\mathcal{E} = \mathcal{DTIME}[2^{O(n)}]$ is hard for *small circuits* that are *uniformly generated* by non-deterministic machines (see Section 1.3). This indicates that the answer to the classical question might be positive, and suggests a path towards proving so.

Lastly, we show that *disproving* a conjecture similar to rETH requires proving breakthrough circuit lower bounds (see Theorem 1.7, and see the discussion in Section 1.4 for a comparison with the state-of-the-art).

Relation to Strong *Exponential Time Hypotheses.* The exponential-time hypotheses that we consider also have "strong" variants that conjecture 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; see, e.g., [63]).[2] In this paper we focus only on

[1] In [61], the introduction of these variants is credited to a private communication from Carmosino, Gao, Impagliazzo, Mihajlin, Paturi, and Schneider [7].

[2] Some "strong" variants of standard exponential-time hypotheses are in fact known to be false (see [61]).

the "non-strong" variants that conjecture lower bounds of $2^{\epsilon \cdot n}$ for some $\epsilon > 0$. Indeed, the point is that even the variants that we consider already have far-reaching consequences for derandomization and circuit lower bounds.

We mention that a recent work of Carmosino, Impagliazzo, and Sabin [8] studied the implications of hypotheses in *fine-grained complexity* on derandomization. These fine-grained hypotheses are implied by the "strong" version of rETH (i.e., by rSETH), but are not known to follow from the "non-strong" versions that we consider in this paper. We will refer again to their results in Section 1.2.

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 $\epsilon > 0$. 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 $2^{\epsilon \cdot (n/\log(n))}$.

1.2.1 Background: Uniform hardness vs randomness. Intuitively, using "hardness-to-randomness" results, we expect that a strong lower bound such as rETH would imply a strong derandomization result. When starting from lower bounds for *non-uniform* circuits, and aiming to deduce *worst-case* derandomization, smooth tradeoffs that yield such results are well-known (see, e.g., [34, 44, 50, 54, 57]) The key problem, however, is that the long line-of-works that starts from hardness for *uniform algorithms* (and aims to deduce average-case derandomization) did not yield such smooth trade-offs so far (see [6, 8, 20, 26, 27, 33, 35, 41, 51, 56]).

Ideally, given an exponential lower bound for uniform probabilistic algorithms (such as $\mathcal{E} \nsubseteq \mathtt{i.o.}\mathcal{BPTIME}[2^{\epsilon \cdot n}]$)[3] we would like to deduce that there exists a PRG with exponential stretch for uniform circuits, and consequently that $\mathcal{BPP} = \mathcal{P}$ in "average-case".[4] However, prior to the current work, the state-of-the-art (by Trevisan and Vadhan [56]) could at best yield PRGs with *sub-exponential stretch* (i.e., with seed length polylog(n)), even if the hypothesis refers to an exponential lower bound. Moreover, the best currently-known PRG only works on infinitely many input lengths.

Previous works bypassed these two obstacles in various indirect ways. Carmosino, Impagliazzo, and Sabin [8] deduced polynomial-time derandomization of $\mathcal{BPP}$ on all input lengths relying on strong hypotheses from *fine-grained complexity* (these hypotheses are implied by the "strong" version of rETH, i.e. by rSETH). Gutfreund and Vadhan [27] deduced (subexponential-time) derandomization of $\mathcal{RP}$ on all input lengths, rather than of $\mathcal{BPP}$ (see details below). Lastly, a line-of-works dealing with uniform "hardness-to-randomness" for $\mathcal{AM}$ (rather than for $\mathcal{BPP}$) was able to bypass both obstacles in this context (see, e.g., [26, 41, 51]).[5]

1.2.2 Our contribution to uniform hardness vs randomness. In this work we tackle both obstacles directly. Loosely speaking, our first main result is that rETH implies the existence of a PRG for uniform circuits with *near-exponential stretch*, which can be used for average-case derandomization of $\mathcal{BPP}$ in *nearly-polynomial-time*. Specifically, the PRG that we construct has seed length $\tilde{O}(\log(n)$, and the corresponding derandomization runs in time $2^{\tilde{O}(\log(n))} = n^{\mathrm{loglog}(n)^{O(1)}}$.

Our hardness assumption will in fact be weaker than rETH: It suffices to assume that the Totally Quantified Boolean Formula (TQBF) problem cannot be solved by probabilistic algorithms that run in time $2^{n/\mathrm{polylog}(n)}$ (see

[3] See Section 3.1 for definitions of complexity classes used throughout the paper.

[4] Throughout the paper, when we say that a PRG is ϵ-pseudorandom for *uniform circuits*, we mean that for every efficiently-samplable distribution over circuits, 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 3.6 and 3.7). The existence of such PRGs implies an "average-case" derandomization of $\mathcal{BPP}$ in the following sense: For every $L \in \mathcal{BPP}$ there exists an efficient deterministic algorithm D such that every probabilistic algorithm that gets input 1^n and tries to find $x \in \{0,1\}^n$ such that $D(x) \neq L(x)$ has a small probability of success (see, e.g., [20, Prop. 4.4]).

[5] Another relevant work is that of Goldreich [20]: He showed that if $prBPP = prP$, then there exists a PRG for uniform circuits that suffices for this conclusion (in particular, the PRG runs in polynomial time and works for all input lengths).

Definition 4.6 for a standard definition of TQBF). This hypothesis is weaker than rETH because 3-SAT reduces to TQBF with a linear overhead in the input length. (Indeed, it is a far weaker hypothesis, since TQBF is $\mathcal{PSPACE}$-complete whereas 3-SAT is only $\mathcal{NP}$-complete.)

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

The technical statement of Theorem 1.1 is even stronger: For every $t(n) = n^{\mathrm{polyloglog}(n)}$, the PRG is $(1/t)$-pseudorandom for every distribution over circuits that can be sampled in time t and with $O(\log(t))$ bits of advice (see Theorem 4.14 for details).

Theorem 1.1 establishes for the first time that hardness assumptions for $\mathcal{BPTIME}$ yield a PRG for uniform circuits with seed length as short as $\tilde{O}(\log(n))$ and running time as small as $2^{\tilde{O}(\log(n))}$. The proof of this result is based on careful refinements of the proof framework of [33], using new technical tools that we construct. The latter tools significantly refine and strengthen the technical tools that were used by [56] 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*. (The meaning of "infinitely-often" is "on infinitely many input lengths", and the meaning of "almost-always" that will be used next is "on all but finitely many input lengths". Recall that a hypothesis of the form $L \notin \mathcal{BPTIME}[T]$ only means that every probabilistic time-T algorithm fails to compute L infinitely-often.)

The shortcoming of Theorem 1.1 that the derandomization works only infinitely-often is identical to all previous uniform "hardness-to-randomness" results that used the [33] proof framework.[6][7] However, known techniques (see, e.g., [27]) can nevertheless be adapted to yield an almost-always PRG that uses $O(\log(n))$ *bits of non-uniform advice* (relying on an almost-always lower bound hypothesis).

We are able to significantly improve this: Assuming the "almost-always" version of rETH, we show that $\mathcal{BPP}$ can be derandomized in average-case and almost-always, using only a *triply-logarithmic* number (i.e., $O(\mathrm{logloglog}(n))$) of advice bits. In fact, as in Theorem 1.1, it suffices to assume hardness for TQBF, rather than for 3-SAT.

Theorem 1.2 (aa-rETH $\Rightarrow$ almost-always derandomization in time $n^{\mathrm{polyloglog}(n)}$; informal). *Assume that for some $T(n) = 2^{n/\mathrm{polylog}(n)}$ it holds that $\mathsf{TQBF} \notin \mathsf{i.o.}\mathcal{BPTIME}[T]$, and let $t(n) = n^{\mathrm{polyloglog}(n)}$. Then, for every $L \in \mathcal{BPTIME}[t]$ and every distribution ensemble $\mathcal{X}$ that can be sampled in polynomial time, 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* $\Pr_{x\sim\mathcal{X}_n}[D(x) \neq L(x)] < 1/n$.

Similarly to Theorem 1.2, the conclusion in Theorem 1.2 can be strengthened so that it holds for every distribution $\mathcal{X}$ samplable in time $t(n) = n^{\mathrm{polyloglog}(n)}$, and the derandomization succeeds on all but a $(1/t)$-fraction of the inputs under $\mathcal{X}$ (rather than only on a $1 - 1/n$ fraction).

[6] 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 [8, 26]).

[7] As mentioned above, Gutfreund and Vadhan [27, Section 6] showed that if we settle for average-case derandomization of $\mathcal{RP}$ (rather than of $\mathcal{BPP}$), the derandomization can work almost-always. As in previous results, their derandomization is relatively slow (i.e., it works in sub-exponential time). We show that their ideas can be combined with the techniques underlying Theorem 1.1, to deduce a *fast* average-case derandomization $\mathcal{RP}$ that works almost-always (see Theorem 4.15).

Remark 1.3 (non-deterministic extensions). We note that "scaled-up" versions of Theorems 1.1 and 1.2 for *non-deterministic settings* follow easily 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[8] we can deduce strong circuit lower bounds, and hence also *worst-case* derandomization of $prBPP$ and of $prMA$ (see Appendix A for details and for a related result). Similarly, as shown by Gutfreund, Shaltiel, and Ta-Shma [26], a suitable variant of AMETH implies an average-case derandomization of AM.

1.3 NETH and an equivalence of derandomization and circuit lower bounds

In the previous section we considered the hypothesis rETH, and now we consider the Non-Deterministic Exponential-Time Hypothesis (NETH), which asserts 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 $\epsilon > 0$. This hypothesis is an exponential-time version of $coNP \not\subseteq NP$, and is incomparable to rETH (and weaker than MAETH).

1.3.1 Background and a surprising observation. 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 $prBPP$ is equivalent to circuit lower bounds against $\mathcal{E}$*. Specifically, recall that two-way implications between derandomization and circuit lower bounds have been gradually developing since the early '90s (for surveys see, e.g., [45, 60]), and that it is a long-standing question whether the foregoing implications can be strengthened to show a *complete equivalence* between the two. One well-known implication of such an equivalence would be that any worst-case derandomization of $prBPP$ *necessitates* the construction of PRGs that "fool" non-uniform circuits.[9]

Then, being more concrete, the motivating observation for our results in this section is that NETH *implies an affirmative answer to the foregoing classical question.* In fact, this is not difficult to show, relying on known results (see Section 2.2 for details).

1.3.2 Our results: Even very weak forms of NETH *suffice for the equivalence.* Our main contribution is in showing that, loosely speaking, even a *very weak form of* NETH suffices to answer the question of equivalence in the affirmative, and that this weak form of NETH is in some sense *inherent*. Specifically, we say that $L \subseteq \{0,1\}^*$ has $NTIME[T]$-uniform circuits if there exists a non-deterministic machine M that gets input 1^n, runs in time $T(n)$, and satisfies the following: For some non-deterministic choices M outputs a *single circuit* $C\colon \{0,1\}^n \to \{0,1\}$ that *decides L on all inputs $x \in \{0,1\}^n$*, and whenever M does not output such a circuit, it outputs $\perp$. We also quantify the *size* of the output circuit, when this size is smaller than $T(n)$.

The weak forms of NETH that will suffice to show equivalences between derandomization and circuit lower bounds are of the form "$\mathcal{E}$ does not have $NTIME[T]$-uniform circuits of size $S(n) \ll T(n)$", for values of T and S that will be specified below. In words, this hypothesis rules out a world in which every $L \in \mathcal{E}$ can be computed by *small circuits* that can be *efficiently produced by a uniform* (non-deterministic) *machine*. Indeed, this hypothesis is weaker than the NETH-style hypothesis $\mathcal{E} \not\subseteq NTIME[T]$, and even than the hypothesis

[8]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 [61]), there is currently no evidence against the "non-strong" version MAETH.

[9]The question of equivalence is mostly "folklore", but was mentioned several times in writing. It was asked in [30, Remark 33], who proved an analogous equivalence between non-deterministic derandomization with short advice and circuit lower bounds against non-deterministic classes (i.e., against $NTIME$; see also [11]). It was also mentioned as a hypothetical possibility in [56] (referred to there as a "super-Karp-Lipton theorem"). Following the results of [43], the question was recently raised again as a conjecture in [55].

$\mathcal{E} \not\subseteq (\mathcal{NTIME}[T] \cap \mathcal{SIZE}[T])$. [10] The fact that such a weak hypothesis suffices to deduce that derandomization and circuit lower bounds are equivalent can be seen as appealing evidence that the equivalence indeed holds.

Our results refer both to the "low-end" parameter regime, which connects relatively weak circuit lower bounds to relatively slow derandomization algorithms, and to the "high-end" parameter regime, which connects strong circuit lower bounds to fast derandomizatoin algorithms. Showing an equivalence in the former regime will require weaker hypothesis, compared to the latter regime.

Starting with the "low-end" regime, our first result is that if $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of 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 $\mathcal{EXP}$.

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, even infinitely-often. Then,*

$$pr\mathcal{BPP} \subseteq \mathtt{i.o.}pr\mathcal{SUBEXP} \iff \mathcal{EXP} \not\subseteq \mathcal{P}/\mathrm{poly} \ .$$

The scaling of Theorem 1.4 to the "high-end" regime us not smooth, and uses different proof techniques (see Section 5 for details). Nevertheless, an analogous result holds for the extreme "high-end" setting: Under the stronger hypothesis that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\Omega(n)}]$-uniform circuits, we show that $pr\mathcal{BPP} = pr\mathcal{P}$ is equivalent to lower bounds for exponential-sized circuits against $\mathcal{E}$; that is:

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:*

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

(We remind the reader again that circuit lower bounds as in Theorems 1.4 and 1.5 are known to be equivalent to the existence of corresponding PRGs that fool non-uniform circuits [3, 34, 44, 54, 57]. Thus, the hypotheses in these theorems imply that derandomization requires PRGs.)

The very weak version of NETH *is inherent (for a stronger conclusion that it yields).* Remarkably, as mentioned above, hypotheses such as the ones in Theorems 1.4 and 1.5 actually yield a stronger conclusion, and are also *necessary* for that stronger conclusion. Specifically, the stronger conclusion is that even *non-deterministic derandomization of $pr\mathcal{BPP}$* (such as $pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP}$) yields circuit lower bounds against $\mathcal{E}$, which in turn yield PRGs for non-uniform circuits.

Theorem 1.6 ($\mathcal{NTIME}$-uniform circuits for $\mathcal{E}$, non-deterministic derandomization, and circuit lower bounds). *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,*

$$pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP} \Longrightarrow \mathcal{EXP} \not\subseteq \mathcal{P}/\mathrm{poly} \ . \tag{1.1}$$

In the other direction, if Eq. (1.1) *holds, then $\mathcal{E}$ cannot be decided by $\mathcal{NP}$-uniform circuits.*

Note that in Theorem 1.6 there is a gap between the hypothesis that implies Eq. (1.1) and the conclusion from Eq. (1.1). Specifically, the hypothesis refers to $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of polynomial size, whereas the conclusion refers to $\mathcal{NP}$-uniform circuits. By optimizing the parameters, this gap between sub-exponential and polynomial can be considerably narrowed (see Theorem 5.11).

[10] We stress that our hypothesis refers to lower bounds for *uniform* models of computation, for which strong lower bounds (compared to those for non-uniform circuits) are already known. (For example, $\mathcal{NP}$ is hard for $\mathcal{NP}$-uniform circuits of size n^k for every fixed $k \in \mathbb{N}$ (see [49]), whereas we do not even know if $\mathcal{E}^{\mathcal{NP}}$ is hard for non-uniform circuits of arbitrarily large *linear* size.)

1.4 Disproving a version of rETH requires circuit lower bounds

Our last main result is that *disproving* a weak version of rETH requires breakthrough circuit lower bounds. Recall that rETH assumes hardness of the form $2^{\epsilon \cdot n}$ for solving 3-SAT for n-bit formulas; thus, disproving rETH means constructing a probabilistic algorithm that solves 3-SAT for n-bit formulas in time $2^{\epsilon \cdot n}$.

We consider the stronger assumption, that the problem CircuitSAT for n-bit circuits can be solved in probabilistic time $2^{n/\mathrm{polylog}(n)}$. (Recall that in CircuitSAT we want to solve satisfiability for a given general Boolean circuit, rather than for a given depth-two formula as in 3-SAT.) We show that such an algorithm would yield lower bounds for circuits of quasilinear size against $\mathcal{BPE} = \mathcal{BPTIME}[2^{O(n)}]$. [11]

Theorem 1.7 (circuit lower bounds from 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 $\mathcal{BPE} \not\subset \mathcal{SIZE}[n \cdot (\log n)^c]$.*

Theorem 1.7 can be viewed from another perspective, which reveals that it constitutes progress on a well-known technical challenge. Specifically, we can view Theorem 1.7 as belonging to the family of results asserting that circuit-analysis algorithms imply circuit lower bounds (following Williams [59]). Previous results crucially rely on the hypothesis that the circuit-analysis algorithm is *deterministic*. It is a well-known challenge to obtain analogous results for *randomized* algorithms, and indeed Theorem 1.7 is such a result, albeit one that relies on a relatively-fast algorithm (see Section 2.3 for further details and for comparison with known results).

Since Theorem 1.1 deduces a conclusion from a weak version of rETH, and Theorem 1.7 deduces a conclusion from the *negation* of a weak version of rETH, we can combine the two results to obtain a "win-win" statement. This yields the following unconditional Karp-Lipton style result: If $\mathcal{BPE}$ can be decided by circuits of quasilinear size, then $\mathcal{BPP}$ can be derandomized, in average-case and infinitely-often, in time $2^{\tilde{O}(\log(n))} = n^{\mathrm{polyloglog}(n)}$. (See Corollary 6.6 for details and for a precise statement.)

1.5 Open problems and subsequent work

Our work makes significant progress on several long-standing open problems, but by no means did we resolve them completely. Let us mention a few of these problems.

Uniform hardness vs randomness. As mentioned in Section 1.2, the goal in this classical line of work is to deduce smooth tradeoffs between average-case derandomization and hardness for uniform probabilistic algorithms (which mirror the known tradeoffs between worst-case derandomization and hardness for non-uniform circuits).

The main open problem is to deduce polynomial-time derandomization from the existence of a hard function computable in exponential time (rather than in linear space as in Theorems 1.1 and 1.2); that is:

Open Problem 1. *Deduce average-case derandomization of $\mathcal{BPP}$ that runs in polynomial time from the existence of a function in $\mathcal{E} = \mathcal{DTIME}[2^{O(n)}]$ that is hard for uniform probabilistic algorithms.*

Progress on the foregoing problem was recently made in a work by three of the current authors [12]. They deduced average-case derandomization of $\mathcal{RP}$ that runs in polynomial time from the existence of a function computable by logspace-uniform circuits of size $2^{O(n)}$ and depth $2^{o(n)}$ that is hard for $\mathcal{BPTIME}[2^{\epsilon \cdot n}]$ (for an arbitrary constant $\epsilon > 0$).

Theorem 1.2 (as well as another result in aforementioned work [12]) deduced derandomization of $\mathcal{BPP}$ on all input lengths that relies on a small number of bits of non-uniform advice. A second open problem is to deduce such derandomization without relying on non-uniform advice:

[11] For context, the best known lower bounds for circuits of quasilinear size are against Σ_2 (see [36]) or against $\mathcal{MA}/1$ (i.e., Merlin-Arthur protocols that use one bit of *non-uniform advice*; see [48]).

Open Problem 2. *Deduce fast derandomization of $\mathcal{BPP}$ (ideally, polynomial time) that works for all input lengths and does not rely on any non-uniform advice, from the existence of a function in $\mathcal{DSPACE}[O(n)]$ (or, better yet, in $\mathcal{E}$) that is hard for uniform probabilistic algorithms.*

In a different direction, a subsequent work by two of the authors [13] showed *worst-case* derandomization from strong hardness assumptions for uniform probabilistic algorithms (namely, from the existence of a function f in $\mathcal{P}$ such that every probabilistic algorithm running in a certain fixed polynomial time fails to compute f on *each and every* sufficiently large input). A follow-up work by Liu and Pass [39] showed an *equivalence* between worst-case derandomization and a similar (albeit more complicated) hardness assumption for conditional time-bounded Kolmogorov complexity.

Derandomization vs circuit lower bounds. As mentioned in Section 1.3, it is a classical question whether derandomization of $pr\mathcal{BPP}$ requires the circuit lower bounds in $\mathcal{E} = \mathcal{DTIME}[2^{O(n)}]$ that are known to imply it. The conditional results in Theorems 1.4 and 1.5 suggest that the answer may be positive, yet proving unconditional results is still a major open problem.

Open Problem 3. *Show the implication $pr\mathcal{BPP} = pr\mathcal{P} \implies \mathcal{E} \not\subset \mathcal{P}/\text{poly}$.*

Interestingly, while the foregoing problem has been open for decades, we are not aware of any significant barriers towards solving it.

2 TECHNICAL OVERVIEW

In this section we describe the proofs of our main results, in high level. In Section 2.1 we describe the proofs of Theorems 1.1 and 1.2; in Section 2.2 we describe the proofs of Theorems 1.4, 1.5 and 1.6; and in Section 2.3 we describe the proof of Theorem 1.7, which relies on the proofs from Section 2.1.

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

Recall that in typical "hardness-to-randomness" results, a PRG is based on a hard function, and the proof amounts to showing that an efficient distinguisher for the PRG can be transformed to an efficient algorithm or circuit that computes the hard function.

In high-level, our proof strategy follows this paradigm, and relies on the classic approach of Impagliazzo and Wigderson [33] for transforming a distinguisher into an algorithm for the hard function. Loosely speaking, the latter approach works only when the hard function $f^{\mathsf{ws}} : \{0,1\}^* \to \{0,1\}^*$ is well-structured; the precise meaning of the term "well-structured" differs across different follow-up works, and in the current work it will also take on a new meaning, but for now let us intuitively think of f^{ws} as downward self-reducible and as having properties akin to random self-reducibility. Instantiating the Nisan-Wigderson PRG with a suitable encoding $\mathsf{ECC}(f^{\mathsf{ws}})$ of f^{ws} as the underlying function (again, the precise requirements from ECC differ across works), our goal is to show that if the PRG with stretch $t(n)$ does not "fool" uniform distinguishers even infinitely-often, then f^{ws} is computable in probabilistic time $t'(n) > t(n)$.

The key challenge underlying this approach is the *significant overheads* in the proof, which increase the time complexity t' of computing f^{ws}. In the original proof of [33] 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 [56] (following [6]), yielded $t'(n) = \text{poly}(t(\text{poly}(n)))$. Since the relevant functions f^{ws} in all works are computable in $\mathcal{E}$, proofs with such an overhead can yield at most a sub-exponential stretch $t(n) = 2^{n^{\Omega(1)}}$.

As mentioned in Section 1.2, previous works bypassed this difficulty either by using stronger hypotheses, or by deducing weaker conclusions, or by working in different contexts (e.g., considering derandomization of $\mathcal{AM}$ rather than of $\mathcal{BPP}$). In contrast, we tackle this difficulty directly, and manage to reduce *all* of the polynomial overheads in the input length to *polylogarithmic overheads* in the input length. That is, we will show that for

carefully-constructed f^{ws} and suitably-chosen ECC (and with some variations in the proof approach), if the PRG instantiated with $\mathsf{ECC}(f^{\mathsf{ws}})$ for stretch t does not "fool" uniform distinguishers infinitely-often, then f^{ws} can be computed in time $t'(n) = t(\widetilde{O}(n))^{O(1)}$.

2.1.1 The well-structured function f^{ws}. Let us now be more specific about the properties of the well-structured function f^{ws} that we need in our proof. Our function f^{ws} will satisfy the following:

(1) **(Very efficient $\mathcal{PSPACE}$-completeness:)** The $\mathcal{PSPACE}$-complete problem TQBF is reducible to f^{ws} in *quasilinear time*, and f^{ws} is computable in linear space.[12]
(2) **(Not too inefficient downward self-reducibility:)** The function f^{ws} is downward self-reducible in time $2^{n/\mathrm{polylog}(n)}$ (see Definition 4.1 for a standard definition).
(3) **(A strengthening of random self-reducibility:)** The function f^{ws} is sample-aided worst-case to δ-average-case reducible, for $\delta(n) = 2^{-n/\mathrm{polylog}(n)}$.

The last property, which is implicit in many works and was recently made explicit by Goldreich and G. Rothblum [23], asserts the following: There exists a uniform algorithm T that gets as input a circuit $C\colon \{0,1\}^n \to \{0,1\}^*$ that agrees with f^{ws}_n on at least $\delta(n)$ of the inputs, and *labeled examples* $(x, f^{\mathsf{ws}}(x))$ 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 $C'\colon \{0,1\}^n \to \{0,1\}^*$ that computes f^{ws}_n on all inputs (see Definition 4.2).

(Our construction of f^{ws} will also satisfy an additional property, which will only be used in the proof of Theorem 1.2 (i.e., of the "almost-always" version of the result). We will describe this property in the proof outline for Theorem 1.2 below.)

The construction of f^{ws}. Let us now explain how we construct f^{ws}. Following Trevisan and Vadhan [56], our f^{ws} is an artificial $\mathcal{PSPACE}$-complete problem that we carefully construct. Their goal was to construct a $\mathcal{PSPACE}$-complete problem that will be simultaneously downward self-reducible and randomly self-reducible. Our goal will be to obtain a construction with stronger completeness and random self-reducibility properties, while compromising on a slower downward self-reducibility algorithm (as detailed above). In a gist, we do so by drastically improving the efficiency of parts of their construction; details follow.

The construction in [56] is based on the proof of $\mathcal{IP} = \mathcal{PSPACE}$ [42, 52]. Recall that the latter proof starts with a given 3-SAT formula φ, which represents a fully quantified instance for TQBF (see Definition 4.6 for the standard definition). The proof then arithmetizes the TQBF function on φ by a low-degree polynomial $P^{(\varphi,0)} = Q_1 \circ Q_2 \circ \ldots \circ Q_{\mathrm{poly}(n)} \circ P^{(\varphi)}$, where $P^{(\varphi)}$ is a standard arithmetization of 3-SAT, and the Q_i's are suitable arithmetic operators (i.e., arithmetizations of the $\forall$ and of the $\exists$ operators, as well as an operator that lowers the degree of the intermediary polynomial). Finally, the proof defines a sequence of $\mathrm{poly}(n)$ polynomials $P^{(\varphi,1)}, \ldots, P^{(\varphi,\mathrm{poly}(n))}$, where for $i = 1, \ldots, \mathrm{poly}(n)$, the polynomial $P^{(\varphi,i)}$ applies one less operator to $P^{(\varphi)}$, compared to $P^{(\varphi,i-1)}$. The crucial observation of [56] is that computing each $P^{(\varphi,i)}$ efficiently reduces to computing $P^{(\varphi,i-1)}$, and thus this sequence of polynomials already has a property reminiscent to downward self-reducibility (whereas the polynomials are of low degree, and thus compute functions that are random self-reducible).

Loosely speaking, the function from [56] defines, for every integer $n \in \mathbb{N}$, a corresponding interval I_n of $\mathrm{poly}(n)$ input lengths; for simplicity of presentation, let us pretend that this interval is $I_n = [n, \ldots, N = \mathrm{poly}(n)]$. At input length $N = \mathrm{poly}(n)$ the function gets as input a 3-SAT formula φ over n variables and outputs $P^{(\varphi,0)}$. Then, for $i \in [\mathrm{poly}(n)]$, at input length $N - i$, the function gets input (φ, w), where w is a sequence of auxiliary variables, and outputs $P^{(\varphi,i)}(w)$. Given the observation mentioned above, this function is downward self-reducible and randomly self-reducible.

[12] For our derandomization results, it would have sufficed for f^{ws} to be computable in quasiexponential time $2^{\tilde{O}(n)}$ rather than linear space; see the comment in the end of Section 4.1.2.

Going through their proof (with needed adaptations for our "high-end" parameter setting), we encounter four different polynomial overheads in the input length, when reducing from TQBF to their function. The first and obvious one is that inputs of length n are mapped to inputs of length $N = \text{poly}(n)$, corresponding to the number of rounds in the $\mathcal{IP} = \mathcal{PSPACE}$ protocol. The other polynomial overheads in the input length come from their reduction of TQBF to an intermediate problem that takes both φ and w as part of the input and is still amenable to arithmetization,[13] from the field size that is required for the stronger random self-reducibility property that we need, and from the way the $\text{poly}(n)$ polynomials are combined into a single Boolean function.

The main challenge is to eliminate all of the foregoing overheads *simultaneously*. Our first main idea is to use an $\mathcal{IP} = \mathcal{PSPACE}$ protocol with $\text{polylog}(n)$ rounds instead of $\text{poly}(n)$ rounds, so that the first overhead (i.e., the additive overhead in the input length caused by the number of operators) will be only $\text{polylog}(n)$ instead of $\text{poly}(n)$. Indeed, in such a protocol the verification time in each round is high, and therefore our downward self-reducibility algorithm is relatively slow and makes many queries; but we will be able to afford this.

While implementing this idea, we define a different intermediate problem that is both amenable to arithmetization and reducible from TQBF in quasilinear time, reyling on an efficient Cook-Levin theorem (see Claim 4.7.1); we move to an arithmetic setting that will support the strong random self-reducibility property that we want, and arithmetize the intermediate problem in this setting (see Claim 4.7.2); we show how to execute arithmetic operators in a "batch" in this arithmetic setting (see Claim 4.7.3); and we efficiently combine the resulting collection of polynomials into a single Boolean function (see the last part of the proof of Lemma 4.7).

We stress that we are "paying" for all the optimizations above, by the fact that the associated algorithms (for downward self-reducibility and for our notion of random self-reducibility) now run in time $2^{n/\text{polylog}(n)}$, rather than polynomial time; but again, we are able to afford this in our proof.

2.1.2 Instantiating the [33] proof framework with the function f^{ws}. Given this construction of f^{ws}, we now use a variant of the proof framework of Impagliazzo and Wigderson [33], as follows. For simplicity, in this overview we show how to "fool" polynomial-time distinguishers that do not use advice. (The full technical proof appears in Section 4.2, see the proof of Lemma 4.9.)

Let ECC be the Goldreich-Levin [21] (i.e., Hadamard) encoding $\text{ECC}(f^{\text{ws}})(x, r) = \oplus_i f^{\text{ws}}(x)_i \cdot r_i$. Our PRG is the Nisan-Wigderson PRG, instantiated with $\text{ECC}(f^{\text{ws}})$ as the hard function, and with seed length $\tilde{O}(\log(n))$. To analyze it, we rely on the well-known "uniform reconstruction" argument of [33] (following [44]), which shows the following: If for input length n there exists a uniform $\text{poly}(n)$-time distinguisher A for the PRG, then for input length $\ell = \widetilde{O}(\log(n))$ there is a *weak learner* for $\text{ECC}(f^{\text{ws}})$. That is, there exists an algorithm that gets input 1^ℓ and oracle access to $\text{ECC}(f^{\text{ws}})$ on ℓ-bit inputs, runs in time $\text{poly}(n) \approx 2^{\ell/\text{polylog}(\ell)}$, and outputs a small circuit that agrees with $\text{ECC}(f^{\text{ws}})$ on approximately $1/2 + 1/n^2 \approx 1/2 + \delta_0(\ell)$ of the ℓ-bit inputs, where $\delta_0(\ell) = 2^{-\ell/\text{polylog}(\ell)}$.

Thus, 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 the "bootstrapping" idea of [33], we now iteratively construct, for each input length $i = 1, ..., \ell$, a circuit of size $2^{i/\text{polylog}(i)}$ for f_i^{ws}. The base case $i = 1$ is trivial. And, in iteration $i > 1$, having already obtained a circuit C_{i-1} for f_{i-1}^{ws}, we run the weak learner for $\text{ECC}(f^{\text{ws}})$ on input length $2i$, and answer its oracle queries using the downward self-reducibility of f^{ws}, the circuit C_{i-1}, and the fact that $\text{ECC}(f^{\text{ws}})_{2i}$ is easily computable given access to f_i^{ws}.

The weak learner outputs a circuit $C_i^{(0)}$. of size $2^{2i/\text{polylog}(2i)}$ that agrees with $\text{ECC}(f^{\text{ws}})$ on approximately $1/2 + \delta_0(2i)$ of the $2i$-bit inputs, and we want to transform it into a circuit that computes f^{ws} on all i-bit inputs. To do so we first use the list-decoding algorithm of Goldreich and Levin [21] to efficiently transform $C_i^{(0)}$ to a circuit $C_i^{(1)}$ of similar size that computes f^{ws} on a approximately $\delta(i) = \text{poly}(\delta_0(2i))$ of the i-bit inputs; the

[13] Recall that the standard arithmetization of 3-SAT is a polynomial that depends on the input formula, whereas we want a single polynomial that gets both a formula and the assignment as input.

algorithm of [21] succeeds only with probability $\text{poly}(\delta)$, so we run it for $\text{poly}(1/\delta)$ times, and each time test the agreement of the resulting circuit with f^{ws}, using the circuit, C_{i-1} and the downward self-reducibility of f^{ws}.

Our goal now is to transform $C_i^{(1)}$ into a circuit of similar size that computes f^{ws} on all i-bit inputs. Recall that in general, performing such transformations by a *uniform* algorithm is challenging (intuitively, if the truth-table of f^{ws} is a codeword in an error-correcting code, then this task corresponds to uniform list-decoding of a "very corrupt" version of f^{ws}). However, in our specific setting we can produce random *labeled samples* for f^{ws}, using its downward self-reducibility and the circuit C_{i-1}. Relying on the *sample-aided worst-case to average-case reducibility* of f^{ws}, we can transform $C_i^{(1)}$ to a circuit C_i of similar size that computes f_i^{ws} on all inputs.

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

2.1.3 The "almost-always" version: Proof of Theorem 1.2. We now explain how to adapt the proof above in order to get an "almost-always" PRG with near-exponential stretch. For starters, we will use a stronger property of f^{ws}, namely that it is downward self-reducible in a polylogarithmic number of steps; this means that for every input length ℓ there exists an input length $\ell_0 \geq \ell - \text{polylog}(\ell)$ such that f^{ws} is efficiently-computable at input length ℓ_0 (i.e., $f_{\ell_0}^{\text{ws}}$ is computable in time $2^{\ell_0/\text{polylog}(\ell_0)}$ without a "downward" oracle); see Section 4.1.1 for intuition and details about this property.

Now, observe that the transformation of a probabilistic distinguisher A for the PRG to a probabilistic algorithm F that computes f^{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^{ws} correctly at input length $\ell = \ell(n) = \widetilde{O}(\log(n))$; specifically, we want to use the downward self-reducibility argument for f^{ws} at input lengths $\ell, \ell-1, ..., \ell_0$, and S_n is the set of input lengths at which we need a distinguisher for G in order to obtain a weak learner for $\text{ECC}(f^{\text{ws}})$ at input lengths $\ell, \ell-1, ...\ell_0$. Moreover, since f^{ws} is downward self-reducible in polylog steps, we will only need weak learners at inputs $\ell, ..., \ell_0 = \ell - \text{polylog}(\ell)$; 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^{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.[14]

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 $G(1^m)$ (i.e., the PRG at input length m) and truncates the output to n bits. (We can indeed show that truncating the output of our PRG preserves its pseudorandomness in a uniform setting; see Proposition 4.12 for details.) The crucial point is that 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, its advice) depends on the distinguisher that it wants to "fool". Thus, for every $L \in \mathcal{BPP}$ and every efficiently-samplable distribution $\mathcal{X}$ of inputs, there exists a corresponding "almost-always" derandomization algorithm $D_{\mathcal{X}}$ (see Proposition 4.12).

[14] Actually, since f^{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 TQBF can be reduced to f^{ws} with quasilinear overhead, if TQBF is "hard" almost-always then for every $\ell(n)$ there exists $\ell' \leq \widetilde{O}(\ell(n))$ such that f^{ws} is "hard" on ℓ', which allows our argument to follow through, with a similar set $\overline{S_n} \subset [n, n^{\text{polyloglog}(n)}]$ (see Proposition 4.11 for details). For simplicity, we ignore this issue in the overview.

2.2 $\mathcal{NTIME}$-uniform circuits for $\mathcal{E}$ and an equivalence between derandomization and circuit lower bounds

The proofs that we describe in the current section are significantly simpler technically than the proofs described in Sections 2.1 and 2.3. As mentioned in Section 1.3, the motivating observation is that NETH implies an equivalence between derandomization and circuit lower bounds; let us start by proving this statement:

Proposition 2.1 ("warm-up": a weaker version of Theorem 1.4). *Assume that* $\mathcal{EXP} \not\subset \texttt{i.o.}\mathcal{NSUBEXP}$. *Then,* $pr\mathcal{BPP} \subseteq pr\mathcal{SUBEXP} \iff \mathcal{EXP} \not\subset \texttt{i.o.}\mathcal{P}/\text{poly}$.

Proof. The "$\Longleftarrow$" direction follows (without any assumption) from [3]. For the "$\Longrightarrow$" direction, assume that $pr\mathcal{BPP} \subseteq pr\mathcal{SUBEXP}$, and assume towards a contradiction that $\mathcal{EXP} \subset \texttt{i.o.}\mathcal{P}/\text{poly}$. The latter hypothesis implies (using the Karp-Lipton style result of [3]) that $\mathcal{EXP} \subset \texttt{i.o.}\mathcal{MA}$. Combining this with the former hypothesis, we deduce that $\mathcal{EXP} \subset \texttt{i.o.}\mathcal{NSUBEXP}$, a contradiction. ∎

Our proofs of Theorems 1.4 and 1.5 will follow the same logical structure as the proof of Proposition 2.1, and our goal will be to relax the hypothesis $\mathcal{EXP} \not\subset \texttt{i.o.}\mathcal{NSUBEXP}$.[15] We will do so by strengthening the Karp-Lipton style result that uses [3] and asserts that a joint "collapse" hypothesis and derandomization hypothesis implies that $\mathcal{EXP}$ can be decided in small non-deterministic time. We will show two different strengthenings, each referring to a different parameter setting: The first strengthening refers to a "low-end" setting, and asserts that if $\mathcal{EXP} \subset \mathcal{P}/\text{poly}$ and $pr\mathcal{BPP} \subseteq pr\mathcal{SUBEXP}$ then $\mathcal{EXP}$ has $\mathcal{NSUBEXP}$-uniform circuits of polynomial size (see Item (1) of Proposition 5.6); and the second strengthening refers to a "high-end" setting, and asserts that if $\mathcal{E} \subset \texttt{i.o.}\mathcal{SIZE}[2^{\epsilon \cdot n}]$ and $pr\mathcal{BPP} = pr\mathcal{P}$ then $\mathcal{E}$ has $\mathcal{NTIME}[2^{O(\epsilon)\cdot n}]$-uniform circuits (see Proposition 5.7). The proofs of these two different strengthenings rely on different ideas; for high-level descriptions of the proofs see Sections 5.1.2 and 5.1.3, respectively.

For context, recall that (as noted by Fortnow, Santhanam, and Williams [17]), the proof of [3] already supports the stronger result that $\mathcal{EXP} \subset \mathcal{P}/\text{poly} \iff \mathcal{EXP} = \mathcal{OMA}$;[16] and by adding a derandomization hypothesis (e.g., $pr\mathcal{BPP} = pr\mathcal{P}$) we can deduce that $\mathcal{EXP} = \mathcal{ONP}$. Nevertheless, our results above are stronger, because $\mathcal{NP}$-uniform circuits are an even weaker model than $\mathcal{ONP}$: This is since in the latter model the proof is verified on an input-by-input basis, whereas in the former model we only verify *once* that the proof is convincing for *all* inputs. We also stress that some lower bounds for this weaker model (i.e., for $\mathcal{NTIME}$-uniform circuits of small size) are already known: Santhanam and Williams [49] proved that for every $k \in \mathbb{N}$ there exists a function in $\mathcal{NP}$ that cannot be computed by $\mathcal{NP}$-uniform circuits of size n^k.

We also note that our proofs actually show that (conditioned on lower bounds for $\mathcal{NTIME}$-uniform circuits against $\mathcal{E}$) even a *relaxed derandomization hypothesis* is already equivalent to the corresponding circuit lower bounds. For example, in the "high-end" setting, to deduce that $\mathcal{E} \not\subset \mathcal{SIZE}[2^{\Omega(n)}]$ it suffices to assume that CAPP on v-bit circuits of size $n = 2^{\Omega(v)}$ can be solved in time $2^{\epsilon \cdot v}$, for a sufficiently small $\epsilon > 0$.[17] For more details, see Section 5.2.

[15] This high-level proof structure, which combines a non-uniform collapse hypothesis (using a Karp-Lipton-style theorem) and a derandomization hypothesis, dates back to the work of Impagliazzo, Kabanets, and Wigderson [30], underlies the algorithmic method of Williams [59], and has been used in works published in parallel to ours (such as Chen *et al.* [10]).

[16] The notation $\mathcal{OMA}$ stands for "oblivious" $\mathcal{MA}$. It denotes the class of problems that can be decided by an $\mathcal{MA}$ verifier such that for every input length there is a single "good" proof that convinces the verifier on all inputs in the set (rather than a separate proof for each input); see, e.g., [17, 22].

[17] Note that the problem of solving CAPP for v-bit circuits of size $n = 2^{\Omega(v)}$ can be trivially solved in time $2^{O(v)} = \text{poly}(n)$, and thus unconditionally lies in $pr\mathcal{P} \cap pr\mathcal{BPTIME}[\tilde{O}(n)]$. The derandomization problem described above simply calls for a *faster* deterministic algorithm for this problem.

Proof of Theorem 1.6. The first part of Theorem 1.6 asserts that if $\mathcal{E}$ does not have $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of polynomial size, then the conditional statement "$pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP} \implies \mathcal{EXP} \not\subseteq \mathcal{P}/\text{poly}$" holds. The proof of this statement again follows the logical structure from the proof of Proposition 2.1, and relies on a further strengthening of our "low-end" Karp-Lipton style result such that the result only uses the hypothesis that $pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP}$ rather than $pr\mathcal{BPP} \subseteq pr\mathcal{SUBEXP}$.[18]

The second part of Theorem 1.6 asserts that if the conditional statement "$pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP} \implies \mathcal{EXP} \not\subseteq \mathcal{P}/\text{poly}$" holds, then $\mathcal{E}$ does not have $\mathcal{NP}$-uniform circuits. We will in fact prove the stronger conclusion that $\mathcal{E} \not\subseteq (\mathcal{NP} \cap \mathcal{P}/\text{poly})$. (Recall that the class of problems decidable by $\mathcal{NP}$-uniform circuits is a subclass of $\mathcal{ONP} \subseteq \mathcal{NP} \cap \mathcal{P}/\text{poly}$.) The proof itself is very simple: Assume towards a contradiction that $\mathcal{E} \subseteq (\mathcal{NP} \cap \mathcal{P}/\text{poly})$; since $\mathcal{BPP} \subseteq \mathcal{EXP}$, it follows that $pr\mathcal{BPP} \subseteq pr\mathcal{NP}$ (see the proof of Theorem 5.10); and by the hypothesized conditional statement, we deduce that $\mathcal{EXP} \not\subseteq \mathcal{P}/\text{poly}$, a contradiction. Indeed, the parameter choices in the foregoing proof are far from tight, and (as mentioned after the statement of Theorem 1.6) the quantitative gap between the two parts of Theorem 1.6 can be considerably narrowed (see Theorem 5.11).

2.3 Circuit lower bounds from randomized CircuitSAT algorithms

Recall that Theorem 1.7 asserts that if CircuitSAT for n-bit circuits of size $\tilde{O}(n^2)$ can be solved in probabilistic time $2^{n/(\log n)^c}$, then $\mathcal{BPE} \not\subseteq \mathcal{SIZE}[n \cdot (\log n)^{c'}]$, where c' depends on c. The relevant context for this result is the known line of works that deduce circuit lower bounds from "non-trivial" circuit-analysis algorithms, following the celebrated result of Williams [59]. The main technical innovation in Theorem 1.7 is that our hypothesis is only that there exists a *probabilistic* circuit-analysis algorithm, whereas the aforementioned known results crucially rely on the fact that the circuit-analysis algorithm is *deterministic*. On the other hand, the aforementioned known results yield new circuit lower bounds even if the running time of the algorithm is $2^n/n^{\omega(1)}$,[19] whereas Theorem 1.7 only yields new circuit lower bounds if the running time is $2^{n/\text{polylog}(n)}$.

As far as we are aware, Theorem 1.7 is the first result that deduces circuit lower bounds from a near-exponential-time probabilistic algorithm for a natural circuit-analysis task. The closest result that we are aware of is by Oliveira and Santhanam [46, Theorem 14], who deduced lower bounds for circuits of size $n^{O(1)}$ against $\mathcal{BPE}$ from probabilistic algorithms for *learning with membership queries* (rather than for a circuit-analysis task such as CircuitSAT); as explained next, we build on their techniques in our proof.[20]

Our proof strategy is indeed very different from the proof strategies underlying known results that deduce circuit lower bounds from deterministic circuit-analysis algorithms (e.g., from the "easy-witness" proof strategy [9, 11, 14, 30, 43, 59], or from proofs that rely on $\mathcal{MA}$ lower bounds [30, Rmk. 26], [48, 55]). 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 [46, Sec. 5] (following [16, 28, 37]). Loosely speaking, their connection relies on the classical results of [33], and we are able to significantly refine this connection, using our refined version of the [33] 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 $2^{n/\text{polylog}(n)}$ for an arbitrarily large polylogarithmic function. We show that in this case, any

[18] Intuitively, in the "low-end" Karp-Lipton result we only need to derandomize probabilistic decisions made by the non-deterministic machine that constructs the circuit, whereas the circuit itself is deterministic; thus, a non-deterministic derandomization hypothesis suffices for this result. See Section 5.1.2 for details.

[19] For example, from such an algorithm they deduce the lower bound $\mathcal{NEXP} \not\subseteq \mathcal{P}/\text{poly}$; and from an algorithm that runs in time $2^{n/\text{polylog}(n)}$ as in Theorem 1.7, their results yield the lower bound $\mathcal{NP} \not\subseteq \mathcal{SIZE}[n^k]$ for every fixed $k \in \mathbb{N}$.

[20] Another known result, which was communicated to us by Igor Oliveira, asserts that if CircuitSAT for circuits over n variables and of size $\text{poly}(n)$ can be solved in probabilistic sub-exponential time $2^{n^{o(1)}}$, then $\mathcal{BPTIME}[2^{O(n)}] \not\subseteq \mathcal{P}/\text{poly}$. This result can be seen as a "high-end" form of our result (i.e., of Theorem 1.7), where the latter will use a weaker hypothesis but deduce a weaker conclusion.

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).[21] Now, let f^{ws} be the well-structured function from Section 2.1, and recall that f^{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^{ws} does not have circuits of size $n \cdot (\log n)^k$. In this case a Boolean version of f^{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^{ws} has circuits of size $n \cdot (\log n)^k$. Hence, f^{ws} is also learnable (as we concluded above), and so the argument of [33] can be used to show that f^{ws} is computable by an efficient probabilistic algorithm.[22] Now, by a diagonalization argument, there exists $L^{\mathrm{diag}} \in \Sigma_4[n \cdot (\log n)^{2k}]$ that cannot be computed by circuits of size $n \cdot (\log n)^k$. We show that $L^{\mathrm{diag}} \in \mathcal{BPE}$ by first reducing L^{diag} to f^{ws} in time $\widetilde{O}(n)$, and then computing f^{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 [33] argument (which was described in Section 2.1) yields a probabilistic algorithm for f^{ws} with very little overhead, which allows us to indeed show that $L^{\mathrm{diag}} \in \mathcal{BPE}$. Specifically, our implementation of the argument (with the specific well-structured function f^{ws}) shows that if f^{ws} can be learned in time $T(n) = 2^{n/\mathrm{polylog}(n)}$, then f^{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^{ws}, we generate a random sample of $r = \mathrm{poly}(n)$ labeled examples $(x_1, f^{\mathsf{ws}}(x_1)), ..., (x_r, f^{\mathsf{ws}}(x_r))$ for f^{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^{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* circuit of size $n \cdot (\log n)^k$ that agrees with f^{ws} on the sample also agrees with f^{ws} on almost all inputs (i.e., by a union-bound over all circuits of such size).

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 Complexity classes

We will use standard complexity-theoretic notation, which can be found in any standard textbook (such as [2, 19]). As few specific reminders for classes that will be used in our paper, let us recall that:

(1) The class $\mathcal{E} = \mathcal{DTIME}[2^{O(n)}]$ is the set of languages decidable in deterministic time $2^{O(n)}$.
(2) For a function $s\colon \mathbb{N} \to \mathbb{N}$, the class $\mathcal{SIZE}[s]$ is the set of languages decidable by an infinite family $\{C_n\colon \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ of Boolean circuits with fan-in two over the De Morgan basis such that C_n is of size at most $s(n)$.

[21] That is, there exists a probabilistic algorithm that gets input 1^n and oracle access to 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.
[22] Actually, our implementation of the [33] argument shows that if the function $\mathtt{ECC}(f^{\mathsf{ws}})$ (where `ECC` is defined as in Section 2.1) can be learned, then the function f^{ws} can be efficiently computed. For simplicity, we ignore the difference between f^{ws} and $\mathtt{ECC}(f^{\mathsf{ws}})$ in the current high-level description.

(3) For a class C of languages, the notation $\mathtt{i.o.}C$ refers the set of languages $L \subseteq \{0,1\}^*$ that agree with some $L' \in C$ on infinitely many input lengths; that is, there exists an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ it holds $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$.
(4) The notation $prBPP$ refers to the set of promise problems decidable in probabilistic polynomial time; that is, the set of pairs $(\mathsf{Y}, \mathsf{N}) \in \{0,1\}^* \times \{0,1\}^*$ such that there exists a probabilistic polynomial time machine M satisfying $x \in \mathsf{Y} \Rightarrow \Pr[M(x) = 1] \geq 2/3$ and $x \in \mathsf{N} \Rightarrow \Pr[M(x) = 0] \geq 2/3$.
(5) The class $\mathcal{SUBEXP} = \cap_{\epsilon>0}\mathcal{DTIME}[2^{n^\epsilon}]$ it the set of languages decidable in sub-exponential time (i.e., time 2^{n^ϵ} where $\epsilon > 0$ can be an arbitrarily small constant). Similarly, the class $\mathtt{i.o.}prSUBEXP$ is the set of promise problems decidable in sub-exponential time on infinitely many input lengths; and the class $prNSUBEXP$ is the set of promise problems decidable in sub-exponential time.

3.2 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 [15]). 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 [7]). 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.

3.3 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 v input bits of size at most $S(v)$ that accept at least an α fraction of their inputs.*
- *The NO instances are (representations of) circuits over v input bits of size at most $S(v)$ 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$[S]$ *problem. We define* CAPP *to be the problem when there is no restriction on S.*

It is well-known that CAPP is complete for $prBPP$ under deterministic polynomial-time reductions; in particular, CAPP can be solved in deterministic polynomial time *if and only if* $prBPP = prP$. (For a proof see, e.g. [58, Cor. 2.31], [19, 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 [57] (following the line of works initiated by Nisan and Wigderson [44]).

Theorem 3.2 (Umans' PRG; see [57, 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 3.3 (near-optimal non-uniform hardness-to-randomness using Umans' PRG). *There exists a universal constant $\Delta > 1$ such that for every time-computable $S : \mathbb{N} \to \mathbb{N}$ and for $T(n) = 2^{\Delta \cdot S^{-1}(n^\Delta)}$, we have that*

(1) *If $\mathcal{E} \not\subset \mathcal{SIZE}[S]$ then* $\mathsf{CAPP} \in \mathtt{i.o.}pr\mathcal{DTIME}[T]$.
(2) *If $\mathcal{E} \not\subset \mathtt{i.o.}\mathcal{SIZE}[S]$ then* $\mathsf{CAPP} \in pr\mathcal{DTIME}[T]$.

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

Definition 3.4 (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 [25]. (The construction in [25] is of an extractor, rather than of an averaging sampler, but the two are well-known to be essentially equivalent; see, e.g., [19, Sec. D.4.1.2] or [58, Cor. 6.24].)

Theorem 3.5 (the near-optimal extractor of [25], instantiated as a sampler and for specific parameters). *Let $\gamma \geq 1$ and $\beta > \alpha > 0$ be constants. Then, there exists a polynomial-time algorithm that for every m computes an $(m^{-\gamma}, 2^{-(\beta-\alpha)\cdot m})$-averaging sampler $Samp\colon \{0,1\}^{m'} \to (\{0,1\}^m)^D$, where $m' = (1+\beta) \cdot m$ and $D = \mathrm{poly}(m)$.*

3.4 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. Towards this purpose, we consider a generator G that gets input 1^n, a random seed of length $\ell(n)$, and a stretch parameter $\mathtt{str}(n)$, and outputs $\mathtt{str}(n)$ pseudorandom bits.

Definition 3.6 (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 3.6 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 3.7 (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 3.8 (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., [20, 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 3.9 (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 3.8 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 $x \notin L$. ∎

3.5 An $\mathcal{E}$-complete problem with useful properties

Our proofs in Section 5 will rely on the well-known existence of an $\mathcal{E}$-complete problem L^{nice} with the following useful properties: The problem L^{nice} is randomly self-reducible and that has an instance checker with linear-length queries such that both the instance checker and the random self-reducibility algorithm use a linear number of random bits. Let us properly define these notions:

Definition 3.10 (instance checkers). *A probabilistic polynomial-time oracle machine* IC *is an* instance checker *for a set $L \subseteq \{0,1\}^*$ if for every $x \in \{0,1\}^*$ the following holds:*

(1) *(Completeness.) $\mathrm{IC}^L(x) = L(x)$, with probability one.*

(2) *(Soundness.) For every* $L' \subseteq \{0,1\}^*$ *we have that* $\Pr[\mathtt{IC}^{L'}(x) \notin \{L(x), \perp\}] \leq 1/6$. [23]

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

Definition 3.11 (random self-reducible function). *We say that* $f\colon \{0,1\}^* \to \{0,1\}^*$ *is* randomly self-reducible *if there exists a probabilistic oracle machine* Dec *that gets input* $x \in \{0,1\}^n$ *and access to an oracle* $g\colon \{0,1\}^n \to \{0,1\}^*$*, runs in time* $\mathrm{poly}(n)$*, makes oracle queries such that each query is uniformly distributed in* $\{0,1\}^n$*, and if for every oracle query* $q \in \{0,1\}^n$ *it holds that* $g(q) = f(q)$*, then* $\mathtt{Dec}^g(x) = f(x)$*.*

In high-level, the problem $L^{\mathtt{nice}}$ is the low-degree extension of an (arbitrary) $\mathcal{E}$-complete problem. The intuition is that since $L^{\mathtt{nice}}$ is a low-degree extension it is randomly self-reducible, and since $L^{\mathtt{nice}}$ is $\mathcal{E}$-complete we can construct an instance checker for it. (Specifically, the instance checker for $L^{\mathtt{nice}}$ simulates a PCP verifier for $L^{\mathtt{nice}}$, and the problem of answering the verifier's queries reduces to $L^{\mathtt{nice}}$, to the verifier's queries can be answered using an oracle to $L^{\mathtt{nice}}$.) For details and a full proof, see Appendix C.

Proposition 3.12 (an $\mathcal{E}$-complete problem that is random self-reducible and has a good instance checker). *There exists* $L^{\mathtt{nice}} \in \mathcal{DTIME}[\tilde{O}(2^n)]$ *such that:*

(1) *Any* $L \in \mathcal{DTIME}[2^n]$ *reduces to* $L^{\mathtt{nice}}$ *in polynomial time with a constant multiplicative blow-up in the input length; specifically, for every* n *there exists* $n' = O(n)$ *such that any* n*-bit input for* L *is mapped to an* n'*-bit input for* $L^{\mathtt{nice}}$*.*
(2) *The problem* $L^{\mathtt{nice}}$ *is randomly self-reducible by an algorithm* Dec *that on inputs of length* n *uses* $n + \mathrm{polylog}(n)$ *random bits.*
(3) *There is an instance checker* IC *for* $L^{\mathtt{nice}}$ *that on inputs of length* n *uses* $n + O(\log(n))$ *random bits and makes* $O(1)$ *queries of length* $\ell(n) = O(n)$*.*

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 [33] (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 a very large running time (e.g., running time $2^{n/\mathrm{polylog}(n)}$). Secondly, we also require that for any $n \in \mathbb{N}$ there exists an input length m that is not

[23] The standard definition of instance checkers fixes the error probability to 1/3, but we can reduce the error to 1/6 using standard error-reduction.

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 [23], and is implicit in many previous results (see, e.g., the references in [23]).

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:[24]

Definition 4.2 (sample-aided reductions; see [23, 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 $x \in \{0,1\}^n$ (the probability bound of $2/3$ is over the internal randomness of C).*

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 B.

[24] Definition 4.2 is actually a slightly modified version of the definition in [23]. First, we consider reductions of computing f in the worst-case to computing f in "rare-case", whereas [23] 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.

We are now ready to define well-structured functions. Fixing a parameter $\delta > 0$, a function f^{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 $f^{\mathsf{ws}} : \{0,1\}^* \to \{0,1\}^*$ is* (δ, s)-well-structured *if f^{ws} is 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^{ws} is simply downward self-reducible in time $\mathrm{poly}(1/\delta)$), we say that f^{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 $f^{\mathsf{ws}} : \{0,1\}^* \to \{0,1\}^*$. To formalize this we consider both a reduction R, which transforms any input x for L to an input $R(x)$ for f^{ws}, and a "decision algorithm" D, which translates the non-Boolean result $f^{\mathsf{ws}}(R(x))$ 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 $|R(x)| \leq b(|x|)$ 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 $f^{\mathsf{ws}} : \{0,1\}^* \to \{0,1\}^*$ such that TQBF reduces to f^{ws} in *quasilinear 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 [56], which (loosely speaking) transforms the protocol underlying the $\mathcal{IP} = \mathcal{PSPACE}$ proof into a computational problem $L_{TV} : \{0,1\}^* \to \{0,1\}^*$.[25] 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. (The field size is chosen such that both P and related polynomials that are described below will be of low degree.) 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 $N_0 + i - 1$.[26] 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 $\varphi \in$ TQBF (regardless of w); thus,

[25] Actually, in [56] 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.

[26] In more detail, we define three arithmetic operators on functions $\mathbb{F}^{2n} \to \mathbb{F}$, each indexed by a variable $j \in [n]$, and denote these operators by $\{O_k^j\}_{k \in [3], j \in [n]}$. In each recursive step $i \in [r(n)]$, the polynomial corresponding to input length $N_0 + i$ is obtained by applying operator $O_{k(i)}^{j(i)}$, 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).

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 $f^{\text{ws}} : \{0,1\}^* \to \{0,1\}^*$, 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^{ws} such that TQBF is reducible to f^{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^{ws}. Recall that we want f^{ws} to be worst-case to δ-average-case reducible for a tiny $\delta = 2^{-n/\text{polylog}(n)}$; thus, f^{ws} will involve computing polynomials over a field of large size $|\mathbb{F}| \geq \text{poly}(1/\delta)$. Using the approach of [56], 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^{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^{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 [56] 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 TQBF^{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 TQBF^{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 TQBF^{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 [56] (see Claim 4.7.2).

Thirdly, the number of polynomials in the construction of L_{TV} (i.e., the size of the interval I_n) is $r(n) = \text{poly}(n)$, 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 $N_1 \geq \text{poly}(n)$. 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 $\text{polylog}(n)$ rounds and a verifier that runs in time $2^{n/\text{polylog}(n)}$: Intuitively, at each input length, we define f^{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 $\text{poly}(1/\delta)$, 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 $\text{poly}(1/\delta)$); see Claim 4.7.3 for details.

Lastly, we also want our function to be downward self-reducible in $\text{polylog}(n)$ steps (i.e., after $\text{polylog}(n)$ "downward" steps, the function at the now-smaller input length is computable in time $\text{poly}(1/\delta)$ 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^{ws} simply computes the arithmetized version of TQBF^{loc}, which (as mentioned above) is computable in time $\text{poly}(1/\delta)$.

The complexity of f^{ws}. For our derandomization result, it suffices to prove that f^{ws} is computable in time $2^{\tilde{O}(n)}$, rather than in linear space. (This is because our derandomization algorithm enumerates over all choices for a seed of length $\tilde{O}(n)$, and computes the Nisan-Wigderson generator on each choice, with f^{ws} as the hard function.)

However, analogously to Trevisan and Vadhan [56], we prove the stronger statement that f^{ws} is computable in *linear space*.[27] This stronger property may be of independent interest, and in particular may be used in future work for constructions of PRGs that work in small space. (See Remark 4.13 for further details.)

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 $\mathsf{TQBF} \subseteq \{0,1\}^*$ 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, $\varphi \in \mathsf{TQBF}$ 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 a formula φ that is represented by n bits actually has less than n input variables, since the representation length of an m-bit formula is $O(m \cdot \log(m))$. Thus, an n-bit φ actually has at most $n/O(\log(n))$ variable. In Definition 4.6 we assume for simplicity (and to avoid cumbersome notation) that φ has precisely n input variables, but some of these are dummy variables that are ignored.[28]

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^{ws} such that TQBF can be reduced to f^{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 $f^{\mathsf{ws}} : \{0,1\}^* \to \{0,1\}^*$ such that f^{ws} is computable in linear space, and TQBF deterministically reduces to f^{ws} in time $n \cdot \log^{2c+r}(n)$.*

Proof. In high-level, we first reduce TQBF to a problem $\mathsf{TQBF}^{\mathsf{loc}}$ that will have a property useful for arithmetization, and then reduce $\mathsf{TQBF}^{\mathsf{loc}}$ to a function f^{ws} that we will construct as follows. We will first carefully arithmetize a suitable witness-relation that underlies $\mathsf{TQBF}^{\mathsf{loc}}$; then transform the corresponding arithmetic version of $\mathsf{TQBF}^{\mathsf{loc}}$ 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 $\mathcal{IP} = \mathcal{SPACE}$ [42, 52]); and finally "combine" these polynomials to a Boolean function f^{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 $\mathsf{TQBF}^{\mathsf{loc}}$, such that TQBF is reducible to $\mathsf{TQBF}^{\mathsf{loc}}$ with quasilinear blow-up, and $\mathsf{TQBF}^{\mathsf{loc}}$ 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 $\varphi \in \mathsf{TQBF}$ 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 $\mathsf{TQBF}^{\mathsf{loc}}$ 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 $\mathsf{R\text{-}TQBF}^{\mathsf{loc}}$, we will have that $x \in \mathsf{TQBF}^{\mathsf{loc}}$ iff $\exists w_1 \forall w_2 ... \; \mathsf{R\text{-}TQBF}^{\mathsf{loc}}(x, w)$, where $\mathsf{R\text{-}TQBF}^{\mathsf{loc}}$ is a conjunction of local conditions on (x, w). In more detail:

[27] Recall that the downward self-reducibility algorithm for f^{ws} works in time $\mathrm{poly}(1/\delta) = 2^{n/\mathrm{polylog}(n)}$, and thus the existence of this algorithm does not immediately imply that $f^{\mathsf{ws}} \in \mathcal{SPACE}$.

[28] This choice makes our reduction of TQBF to f^{ws} somewhat wasteful, but this waste only causes only a polylogarithmic overhead, which is insignificant for our results. Thus, for simplicity, we assume that the number of variables indeed equals the representation length of φ.

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

(1) *(Length-preserving witnesses.) For any* $(x,w) \in \mathsf{R\text{-}TQBF}^{\mathrm{loc}}$ *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* $(x,w) \in \mathsf{R\text{-}TQBF}^{\mathrm{loc}}$ *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* $\varphi \in \mathsf{TQBF}$ *if and only if* $x \in \mathsf{TQBF}^{\mathrm{loc}}$.

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 $\varphi \in \mathsf{TQBF}$ if and only if

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

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:[29] There exists a linear-space and quasilinear-time algorithm that, on input 1^n, constructs a 3-SAT formula $\Phi_n : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^{\mathrm{ql}(n)} \to \{0,1\}$ of size $\mathrm{ql}(n) = \tilde{O}(n)$ 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 $w' \in \{0,1\}^{\mathrm{ql}(n)}$ satisfying $\Phi_n(\varphi, w, w') = 1$.

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

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

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 $i \in [\mathrm{ql}(n)]$, 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 $\varphi \in \mathsf{TQBF}$ if and only if

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

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

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

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

[29] The algorithm transforms M into an oblivious machine [24, 47], and then applies an efficient Cook-Levin transformation of the oblivious machine to a 3-SAT formula (see, e.g., [2, Sec 2.3.4]).

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 $2^{5n'} = 1/\text{poly}(\delta_0(n))$ 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 $\text{poly}(n') = \text{poly}(n)$ (by Shoup's [53] 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 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 $|H| = 2^{n'}$).[30]

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 $\mathsf{R\text{-}TQBF}^{\mathsf{loc}}$.

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

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

1. *(Low-degree.) The degree of $P^{\mathsf{TQBF}^{\mathsf{loc}}}$ is at most $O(n \cdot 2^{n'})$.*
2. *(Arithmetizes $\mathsf{R\text{-}TQBF}^{\mathsf{loc}}$.) For every $(x, w) \in H^{\ell_0} \times H^{\ell_0}$ it holds that $P^{\mathsf{TQBF}^{\mathsf{loc}}}(x, w) = 1$ if $(x, w) \in \mathsf{R\text{-}TQBF}^{\mathsf{loc}}$, and $P^{\mathsf{TQBF}^{\mathsf{loc}}}(x, w) = 0$ otherwise.*
3. *(Efficiently-computable.) There exists a deterministic algorithm that gets as input $(x, w) \in \mathbb{F}^{2\ell_0}$, runs in time $\text{poly}(|\mathbb{F}|)$, and outputs $P^{\mathsf{TQBF}^{\mathsf{loc}}}(x, w) \in \mathbb{F}$. 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 $P_0^{\mathsf{TQBF}^{\mathsf{loc}}} : \mathbb{F}^{2n' \cdot \ell_0} \to \mathbb{F}$ 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 $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}(x, w) = 1$ if $(x, w) \in \mathsf{R\text{-}TQBF}^{\mathsf{loc}}(x, w)$, and $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}(x, w) = 0$ 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 $(x, w) \in \mathsf{R\text{-}TQBF}^{\mathsf{loc}}$ 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 $\text{poly}(n)$ 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 $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}$ is simply the multiplication of all the p_i's; that is, $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}(x, w) = \prod_{i \in [n]} p_i(x, w)$. Note that $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}$ can indeed be evaluated in time $\text{poly}(n)$ and in linear space, and that the degree of $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}$ is $O(n)$ (since each p_i is a multilinear polynomial in $O(1)$ variables).

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

[30] 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.

$\mathrm{poly}(|H|) = \mathrm{poly}(|\mathbb{F}|)$ and in linear space (again just using Lagrange's interpolation formula and the fact that π is efficiently-computable).

Finally, we define the polynomial $P^{\mathsf{TQBF}^{\mathsf{loc}}} : \mathbb{F}^{2\ell_0} \to \mathbb{F}$. Intuitively, for $(x, w) \in H^{\ell_0} \times H^{\ell_0}$, the polynomial $P^{\mathsf{TQBF}^{\mathsf{loc}}}$ 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 $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}$ on these $2n' \cdot \ell_0$ bit-projections. More formally, for every $(x, w) \in \mathbb{F}^{2\ell_0}$ we define

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

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 $\deg(P_0^{\mathsf{TQBF}^{\mathsf{loc}}}) = O(n)$. The second item in the claim follows immediately from the definition of $P^{\mathsf{TQBF}^{\mathsf{loc}}}$. 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 $P_0^{\mathsf{TQBF}^{\mathsf{loc}}}$ 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 $\mathrm{poly}(n) < \mathrm{poly}(|\mathbb{F}|)$.) □

Constructing a "downward self-reducible" collection of low-degree polynomials. Our goal now is to define a collection of $O(\ell_0^2)$ polynomials $\left\{P_{n,i} : \mathbb{F}^{2\ell_0} \to \mathbb{F}\right\}_{i \in [O(\ell_0^2)]}$ such that the polynomials are of low degree, and $P_{n,1}$ essentially computes $\mathsf{TQBF}^{\mathsf{loc}}$, 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 $\left\{P_{n,i} : \mathbb{F}^{2\ell_0} \to \mathbb{F}\right\}_{i \in [\bar{\ell}_0]}$, 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 $\mathsf{TQBF}^{\mathsf{loc}}$ 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 $x \in \mathsf{TQBF}^{\mathsf{loc}}$, and $P_{n,1}(x, w) = 0$ if $x \notin \mathsf{TQBF}^{\mathsf{loc}}$. (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.

Quantifications operator: Let $i \in [\ell_0]$. Loosely speaking, $\mathsf{Quant}^{(i)}(p)$ 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. In more detail, consider an input $(x, w) \in \mathbb{F}^{2\ell_0}$ for p, and think of $w = w_1, ..., w_{\ell_0} \in \mathbb{F}$. The operator $\mathsf{Quant}^{(i)}(p)$ causes p to ignore w_i, and instead think of a variable $\pi(\sigma 0^{4n'}) \in H$ that is determined a sequence of n' bits $\sigma = \sigma_1, ..., \sigma_{n'}$; then, $\mathsf{Quant}^{(i)}(p)$ will be the arithmetization of the expression "$\exists\sigma_1\forall\sigma_2\exists\sigma_3... : p(x, w_1, ..., w_{i-1}, \pi(\sigma 0^{4n'}), w_{i+1}, ..., w_{\ell_0})$" (obtained by arithmetizing the "$\exists$" and "$\forall$" operations in the usual way). 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 recursively define $n'+1$ functions $\mathsf{Quant}^{(i,0)}, ..., \mathsf{Quant}^{(i,n')} = \mathsf{Quant}^{(i)}(p)$ such that for $j \in \{0, ..., n'\}$ it holds that $\mathsf{Quant}^{(i,j)}(p)$ is a function $\mathbb{F}^{2\ell_0} \times \{0,1\}^{n'-j} \to \mathbb{F}$. The function $\mathsf{Quant}^{(i,0)}(p)$ 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 $\mathsf{Quant}^{(i,0)}(x,w,\sigma) = p(x, w_1...w_{i-1}\pi(\sigma 0^{4n'}))$. Then, for $j \in [n']$, if j is odd then we define

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

and if j is even then we define

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

Note that the function $\mathsf{Quant}^{(i)}(p)$ can be evaluated at any input in linear space with oracle access to p (since each $\mathsf{Quant}^{(i,j)}(p)$ can be evaluated in linear space with oracle access to $\mathsf{Quant}^{(i,j-1)}(p)$). Also observe the following property of $\mathsf{Quant}^{(i)}(p)$, 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 $\mathsf{Quant}^{(i)}(p)(x,w) = 1$ 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 $\mathsf{Quant}^{(i)}(p)(x,w) = 0$ otherwise.*

Degree-reduction operator: 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

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

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

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

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

Fact 4.7.3.2. *For $i \in [2\ell_0]$, let v be the variable whose degree $\mathsf{DegRed}^{(i)}$ 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 $\mathsf{DegRed}^{(i)}(p)$ is $|H|-1$, and the individual degree of any other input variable to $\mathsf{DegRed}^{(i)}(p)$ 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 $\mathsf{DegRed}^{(i)}(p)(x,w) = p(x,w)$.*

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 $\mathsf{Quant}^{(i)}$ to p, then applying $\mathsf{DegRed}^{(j)}$ 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 $\mathtt{Quant}^{(i)}(p)(x,w') = 1$ 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 $\mathtt{Quant}^{(i)}(p)(x,w') = 0$ otherwise. Now, let $p^{(0)} = \mathtt{Quant}^{(i)}(p)$, and for every $j \in [2\ell_0]$ recursively define $p^{(j)} = \mathtt{DegRed}^{(j)}(p^{(j-1)})$. 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 $p'(x,w') = \mathtt{Quant}^{(i)}(x,w')$. □

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 $P_{n,\ell_0(2\ell_0+1)+1}(x,w) = P^{\mathsf{TQBF}^{\mathsf{loc}}}(x,w)$. 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 $P_{n,i(2\ell_0+1)}(x,w) = \mathtt{Quant}^{(i)}(P_{n,i(2\ell_0+1)+1})(x,w)$; and for each $j = 1,...,2\ell_0$, we define $P_{n,i(2\ell_0+1)-j}(x,w) = \mathtt{DegRed}^{(j)}(P_{n,i(2\ell_0+1)-j+1})(x,w)$.

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 $P^{\mathsf{TQBF}^{\mathsf{loc}}}$, and relying on Claim 4.7.2). Using a suitable composition lemma for space-bounded computation (see, e.g., [19, 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 $P_{n,\ell_0(2\ell_0+1)+1} = P^{\mathsf{TQBF}^{\mathsf{loc}}}$ 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 $x \in \mathsf{TQBF}^{\mathsf{loc}}$ and $P_{n,1}(x,w) = 0$ otherwise (regardless of w). To do so, recall that $P_{n,\bar{\ell}_0} = P^{\mathsf{TQBF}^{\mathsf{loc}}}$, and hence for any $w' \in H^{\ell_0}$ it holds that $P_{n,\bar{\ell}_0}(x,w') = 1$ if $(x,w') \in \mathsf{R\text{-}TQBF}^{\mathsf{loc}}$ 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 $\exists\sigma_1\forall\sigma_2\exists\sigma_3...\forall\sigma_{n'}(x, w'_1...w'_{\ell_0-1}\pi(\sigma_1...\sigma_{n'})) \in \mathsf{R\text{-}TQBF}^{\mathsf{loc}}$, 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 $\exists\sigma_1^{(1)}\forall\sigma_2^{(1)}...\forall\sigma_{n'}^{(1)}...\exists\sigma_1^{(\ell_0)}...\forall\sigma_{n'}^{(\ell_0)} : (x,w') \in \mathsf{R\text{-}TQBF}^{\mathsf{loc}}$, 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 $x \in \mathsf{TQBF}^{\mathsf{loc}}$ and $P_{n,1}(x,w) = 0$ otherwise, as we wanted. □

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 B.1). Our goal

now is simply to "combine" these polynomials into a single Boolean function $f^{\mathsf{ws}} : \{0,1\}^* \to \{0,1\}^*$ 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^{ws} on input length $N + i$ such that it computes (a Boolean version of) $P_{n,\bar{\ell}_0 - i}$. Specifically, $f^{\mathsf{ws}} : \{0,1\}^{N+i} \to \{0,1\}^{N+i}$ 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^{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^{ws} is well-defined.[31] In addition, since the input length to f^{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^{ws} is computable in linear space. To see that TQBF reduces to f^{ws}, recall that by Claim 4.7.1 we can reduce TQBF to $\mathsf{TQBF}^{\mathsf{loc}}$ in time $n \cdot (\log n)^r$ (for some universal constant $r \in \mathbb{N}$); and note that we can then further reduce $\mathsf{TQBF}^{\mathsf{loc}}$ to f^{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^{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^{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^{ws}_N corresponds to the polynomial $P_{n,\bar{\ell}_0(n)-i}$, and f^{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 $f^{\mathsf{ws}}_{N_0}$ 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 $f^{\mathsf{ws}}_{N_0}$ can be computed in time $\mathrm{poly}(2^{n'}) < \mathrm{poly}(1/\delta(N_0))$.

To see that f^{ws} is sample-aided worst-case to δ-average-case reducible, first note that computing f^{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 $n' = \lceil n/\ell_0(n) \rceil$. [32] We use Proposition B.1 with parameter $\rho(\log(|\mathbb{F}^{2\ell_0(n)}|)) = \delta_0(n) < \delta(N)$, 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:

[31] 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 $10 \lceil (n+1)/\ell_0(n+1) \rceil \cdot \ell_0(n+1) + 11(n+1) \cdot \bar{\ell}_0(n+1) \geq 10n + 11(n+1)\bar{\ell}_0(n+1) + 10$.

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

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 $\delta(n) = 2^{-n/\log(n)^c}$ 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 $2^{O(n/\log(n)^c)}$. 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^{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^{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^{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^{ws}. Loosely speaking, the lemma asserts that for any $n \in \mathbb{N}$, if G is *not* pseudorandom for A on every input length in a small set of input lengths surrounding n, then F computes f^{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^{ws} ⇒ computing f^{ws}). *Let $c \in \mathbb{N}$ be an arbitrary constant, let $\delta(n) = 2^{-n/\log(n)^c}$, 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 $f^{\mathsf{ws}} : \{0,1\}^* \to \{0,1\}^*$ 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.*
(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 $S_n \subset \lceil n, n^2 \rceil$ 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 $t_F(\ell) = 2^{O(\ell/\log(\ell)^c)}$ 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^{ws} on strings of length $\ell(n)$.*

Moreover, for any function $\mathtt{str} : \mathbb{N} \to \mathbb{N}$ such that $\mathtt{str}(n) \leq n$, 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 $2^{O(\ell/\log(\ell)^c)}$ fails to compute f^{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. We prove the "moreover" part, and it implies the foregoing statement using the function $\mathtt{str}(n) = n$.

Construction: The generator G_0. For any p, s, δ, k, t, and f^{ws} that satisfy our hypothesis, let $f^{\mathsf{GL}(\mathsf{ws})} : \{0,1\}^* \to \{0,1\}$ be defined as follows: For any $(x, r) \in \{0,1\}^n \times \{0,1\}^n$ we let $f^{\mathsf{GL}(\mathsf{ws})}(x, r) = \sum_{i \in [n]} f^{\mathsf{ws}}(x)_i \cdot r_i$, where the

arithmetic is over $\mathbb{F}_2$.[33] (We use the notation $f^{\mathsf{GL}(\mathsf{ws})}$ since we will use the algorithm of Goldreich and Levin [21] to transform a circuit that agrees with $f^{\mathsf{GL}(\mathsf{ws})}$ on $1/2+\epsilon$ of the inputs into a circuit that computes f^{ws} on $\mathrm{poly}(\epsilon)$ of the inputs.) We will need the following standard definition:

Definition 4.9.1 (combinatorial designs). *An (ℓ, a)-combinatorial design is a collection of sets $S_1, ..., S_n \subseteq [d]$ such that for every $i \in [n]$ it holds that $|S_i| = \ell$, and for every distinct $i, j \in [n]$ it holds that $|S_i \cap S_j| \leq a$. We call n the* number of sets*, and d the* universe size*, and a the* pairwise-intersection size*.*

Consider a combinatorial design that has n sets of size $\ell(n) = \lceil \log(n) \cdot (\mathrm{loglog} n)^b \rceil$ (where b is a sufficiently large constant that depends on k) with pairwise-intersection size $\gamma \cdot \log(n)$, where $\gamma > 0$ is a sufficiently small constant, in a universe of size $\ell_G(n) = \tilde{O}(\ell(n)) = \tilde{O}(\log(n))$ (see, e.g., [58, Prob 3.2] for a polynomial-time construction of such a design).

The algorithm G_0 is the Nisan-Wigderson generator, instantiated with $f^{\mathsf{GL}(\mathsf{ws})}$ as the hard function and with the foregoing design. Since f^{ws} is computable in linear space, the function $f^{\mathsf{GL}(\mathsf{ws})}(x, r)$ is computable in time $n^{\mathrm{polyloglog}(n)}$, and hence G_0 is computable in time $n^{\mathrm{polyloglog}(n)}$ and has seed length $\ell_G(n)$.

Analysis: Transforming a distinguisher A into an algorithm F for f^{ws}. Let us first fix some parameters that will be useful below. Denote $\ell'(n) = \ell(n)/\log(\ell(n))^{c+1}$, and fix a sufficiently small universal constant $\epsilon > 0$. We assume that $\ell(n)$ is sufficiently large such that $t(n) = n^{\log\log(n)^k} \leq 2^{\epsilon \cdot \ell'(n)}$. Recall that, since f^{ws} is downward self-reducible in s steps, there exists an input length $\ell_0(n) \geq \ell(n) - s(\ell(n))$ such that $f^{\mathsf{ws}}_{\ell_0(n)}$ is computable in time $\mathrm{poly}(1/\delta(\ell_0(n)))$. For $L_n = \{\ell_0(n), ..., \ell(n)\}$, we define $S_n = \{\ell^{-1}(2i) : i \in L_n\}$; see Figure 1 for an illustration. Note that indeed $|S_n| \leq s(\ell(n)) = s(\tilde{O}(\log(n)))$; and relying on the fact that $s(\ell(n)) \leq \ell(n)/2$, we have that $S_n \subset [n_0, n_1]$ where $n_0 = \ell^{-1}(2\ell_0) \geq \ell^{-1}(\ell(n)) = n$ and $n_1 = \ell^{-1}(2\ell(n)) < n^2$.

Let A be a probabilistic algorithm that gets input 1^n and $\log(t(n))$ bits of non-uniform advice and runs in time $t(n)$, fix a corresponding advice sequence, and fix a function $\mathtt{str}(n) \leq n$. Recall that we denote $G(1^n, s) = G_0(1^n, s)_{1,...,\mathtt{str}(n)}$.

We call $n \in \mathbb{N}$ distinguishable if for every $m \in S_n$, when A is given input $1^{\mathtt{str}(m)}$ and the advice bits, 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. We will construct a probabilistic algorithm F^{ckt} that gets input $1^{\ell(n)}$, runs in time $2^{O(\ell(n)/\log(\ell(n))^c)}$, and if n is distinguishable, with high probability F^{ckt} outputs a circuit $\{0,1\}^{\ell(n)} \to \{0,1\}$ that correctly computes f^{ws} on $\ell(n)$-bit inputs. (It follows that a probabilistic algorithm F can decide f^{ws} on $\{0,1\}^{\ell(n)}$ in time at most $2^{O(\ell(n)/\log(\ell(n))^c)}$, by running F^{ckt} and evaluating the circuit at the given input.)

Construction and analysis of F^{ckt}. Given as input $1^{\ell(n)}$, the algorithm F^{ckt} iteratively constructs circuits for f^{ws}_i, for increasing values of $i \in L_n = \{\ell_0(n), ..., \ell(n)\}$. The construction for the base case $i = \ell_0(n)$ relies on the fact that $f^{\mathsf{ws}}_{\ell_0(n)}$ is computable in time $\mathrm{poly}(1/\delta(\ell_0(n)))$ (i.e., the circuit for $f^{\mathsf{ws}}_{\ell_0(n)}$ simply implements this algorithm). For subsequent iterations, the algorithm F^{ckt} will rely on the following procedure:

Claim 4.9.2. *There exists an algorithm F^{step} that gets as input $i \in L_n \setminus \{\ell_0(n)\}$ and a circuit $C_{i-1} : \{0,1\}^{i-1} \to \{0,1\}^{i-1}$ that computes f^{ws}_{i-1}, runs in time $2^{O(i/\log(i)^c)} \cdot \mathrm{poly}(|C_{i-1}|)$, and if n is distinguishable, then with probability at least $1 - \exp(-i/\log(i)^{c+1})$ the algorithm F^{step} outputs a circuit $C_i : \{0,1\}^i \to \{0,1\}^i$ of size $2^{O(i/\log(i)^c)}$ that computes f^{ws}_i.*

Before proving Claim 4.9.2, let us see how is suffices for the construction of F^{ckt}. The algorithm F^{ckt} uses F^{step} with inputs $i = \ell_0(n)+1, ..., \ell(n)$, and thus it runs in time $2^{O(\ell(n)/\log(\ell(n))^c)}$. (Note that the size of the output

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

circuit C_i in Claim 4.9.2 does not depend on the size of the input circuit C_{i-1}.) The probability that it outputs a circuit that correctly computes $f^{\mathsf{ws}}_{\ell(n)}$ is at least $1 - \sum_{i=\ell'}^{\ell} \exp(i/\log(i)^{c+1}) \geq 2/3$, assuming that ℓ is sufficiently large. Thus, it remains to prove Claim 4.9.2.

Preliminary step: Constructing a weak learner. Towards constructing $F^{\mathtt{step}}$ and proving Claim 4.9.2, our first step is to construct an efficient algorithm $F^{\mathtt{lrn}}$ that gets input $1^{\ell(m)}$ and oracle acess to $f^{\mathsf{GL}(\mathsf{ws})}$ on $\ell(m)$-bit inputs, uses a small amount of non-uniform advice, and if $m \in S_n$ for a distinguishable n, then the algorithm prints a circuit that computes $f^{\mathsf{GL}(\mathsf{ws})}$ on noticeably more than half of the $\ell(m)$-bit inputs. The construction and proof follow the standard efficient uniform reconstruction argument for the Nisan-Wigderson PRG, from [33] (following [44]).

Claim 4.9.3. *There exists a probabilistic algorithm $F^{\mathtt{lrn}}$ that gets input $1^{\ell(m)}$, and oracle access to $f^{\mathsf{GL}(\mathsf{ws})}$ on $\ell(m)$-bit inputs, and $3\epsilon \cdot \ell'(m)$ bits of non-uniform advice, runs in time $2^{\ell'(m)}$, and satisfies the following. If $m \in S_n$ for a distinguishable n, then with probability more than $2^{-\ell'(m)}$ the algorithm outputs a circuit $\{0,1\}^{\ell(m)} \to \{0,1\}$ that computes $f^{\mathsf{GL}(\mathsf{ws})}$ correctly on more than $1/2 + 2^{-\ell'(m)}$ of the inputs.*

Proof. Let $\ell = \ell(m)$, let $\ell' = \ell'(m)$, and let $m' = \mathtt{str}(m) \leq m$. Let us first assume that $m' = m$ (i.e., $\mathtt{str}$ is the identity function and $G_0 = G$). In this case, a standard argument (based on [44] and first noted in [33]) shows that there exists a probabilistic polynomial time algorithm Rec^{NW} that satisfies the following: When given as input a circuit $D_m : \{0,1\}^m \to \{0,1\}$ that $(1/m^{\mathrm{loglog}(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and also given oracle access to $f^{\mathsf{GL}(\mathsf{ws})}$ on ℓ-bit inputs, with probability at least $1/O(m)$ the algorithm Rec^{NW} outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{x\in\{0,1\}^\ell}[C_\ell(x) = f^{\mathsf{GL}(\mathsf{ws})}(x)] \geq 1/2 + 1/O(m^{\mathrm{loglog}(m)^k})$.

Towards extending this claim to the setting of an arbitrary $m' = \mathtt{str}(m) \leq m$, let us quickly recap the original construction of Rec^{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 $\mathrm{poly}(m)$ values for $f^{\mathsf{GL}(\mathsf{ws})}$ (these are potential values for the output indices whose sets in the design 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 $x \in f^{\mathsf{GL}(\mathsf{ws})}$. Now, note that if the output of the PRG is truncated to length $m' < m$, 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, there is an algorithm Rec^{NW} that gets as input 1^m and a circuit $D_{m'} : \{0,1\}^{m'} \to \{0,1\}$ that $(1/m^{\mathrm{loglog}(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and oracle access to $f^{\mathsf{GL}(\mathsf{ws})}_\ell$, and with probability at least $1/O(m)$ outputs a circuit $C_\ell : \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{x\in\{0,1\}^\ell}[C_\ell(x) = f^{\mathsf{GL}(\mathsf{ws})}(x)] \geq 1/2 + 1/O(m^{\mathrm{loglog}(m)^k})$.

Now, let n be distinguishable, let $m \in S_n$, let $\ell = \ell(m)$, and let $m' = \mathtt{str}(m)$. Our probabilistic algorithm $F^{\mathtt{lrn}}$ is given as input 1^ℓ and non-uniform advice (a, m', m) such that $|a| = \log(t(m)) = \log(m) \cdot \mathrm{loglog}(m)^k = \epsilon \cdot \ell'$; note that, since $m' \leq m$, the total length of the advice is at most $\epsilon \cdot \ell' + 2\log(m) < 2\epsilon \cdot \ell'$. The algorithm $F^{\mathtt{lrn}}$ simulates the algorithm A on input $1^{m'}$ with the advice a, feeds the output of A as input for Rec^{NW} along with 1^m, and outputs the circuit given by Rec^{NW}.

Our algorithm $F^{\mathtt{lrn}}$ runs in time $m^{O(\mathrm{loglog}(m)^k)} = 2^{\ell'}$. With probability more than $(1/m^{\mathrm{loglog}(m)^k})$, the algorithm A outputs $D_{m'} : \{0,1\}^{m'} \to \{0,1\}$ that $(1/m^{\mathrm{loglog}(m)^k})$-distinguishes $G(1^m, \mathbf{u}_{\ell_G(m)})$ from uniform, and conditioned on this event, with probability at least $1/O(m)$ the algorithm $F^{\mathtt{lrn}}$ outputs $C_\ell : \{0,1\}^\ell \to \{0,1\}$ that correctly computes $f^{\mathsf{GL}(\mathsf{ws})}$ on $1/2 + 1/O(m^{\mathrm{loglog}(m)^k}) > 1/2 + 2^{-\ell'}$ of the ℓ-bit inputs. □

Claim 4.9.3 implies that for any distinguishable n, when $F^{\mathtt{lrn}}$ gets input 1^r where $r \in 2L_n = \{2i : i \in L_n\}$, it succeeds (with probability $\geq 2^{-\ell'(n)}$) in printing a circuit that approximates $f^{\mathsf{GL}(\mathsf{ws})}$ on r-bit inputs. (This is

because, by the definition of S_n, any such input length is of the form $\ell(m)$ for $m \in S_n$.) 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.

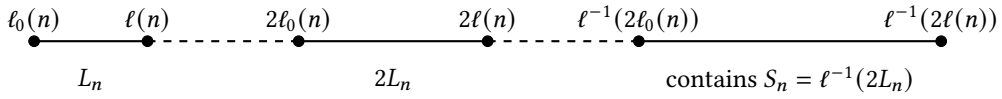

Fig. 1. We want to compute f^{ws} on inputs of length $\ell(n)$. We define a corresponding interval $L_n = \{\ell_0(n), ..., \ell(n)\}$ of input lengths, where $\ell_0(n) \geq \ell(n) - s(\ell(n))$, in which we will use the downward self-reducibility of f^{ws}. We assume that there is a uniform distinguisher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

Proof of Claim 4.9.2. Let $i' = 2i/\log(2i)^{c+1}$, and let $S = |C_{i-1}|$. First note that the algorithm can compute f_i^{ws} in time $\mathrm{poly}(1/\delta(i), S)$ (using the downward self-reducibility of f^{ws} and the circuit C_{i-1}) and also compute $f_{2i}^{\mathsf{GL(ws)}}$ in time $\mathrm{poly}(1/\delta(i), S)$ (using the fact that $f^{\mathsf{GL(ws)}}(x, r) = \sum_{j\in[i]} f_i^{\mathsf{ws}}(x)_j \cdot r_j$). We will construct C_i in a sequence of steps:

1. Simulating the learner for $f_{2i}^{\mathsf{GL(ws)}}$. We enumerate over all $2^{3\epsilon\cdot i'}$ possible advice strings for F^{lrn}. For each fixed advice string $a \in \{0,1\}^{3\epsilon\cdot i'}$, we simulate F^{lrn} on input 1^{2i} with advice a for $2^{O(i')}$ times (using independent randomness in each simulation), while answering its queries to $f_{2i}^{\mathsf{GL(ws)}}$ using C_{i-1}.

Analysis: When a is the "good" advice, each simulation of F^{lrn} is successful with probability at least $2^{-i'}$. Thus, with probability at least $1 - \exp(-i')$ we obtained a list of $2^{O(i')}$ circuits, at least one of which correctly computes $f_{2i}^{\mathsf{GL(ws)}}$ on at least $1/2 + 2^{-i'}$ of its inputs.

2. Weeding the list to find a circuit for $f_{2i}^{\mathsf{GL(ws)}}$. We enumerate over the list of $2^{O(i')}$ circuits. For each circuit, we randomly sample $2^{O(i')}$ inputs, compute $f_{2i}^{\mathsf{GL(ws)}}$ at each of these inputs using C_{i-1}, and compare the value of $f_{2i}^{\mathsf{GL(ws)}}$ to the output of the candidate circuit. If the circuit agrees with $f_{2i}^{\mathsf{GL(ws)}}$ on at least $1/2 + 2^{-i'} - 2^{-2i'}$ of the inputs in the sample, we denote this circuit by $C_i^{(1)}$ and move on to Step 3; otherwise, we continue to the next circuit in the list. If we enumerated over the entire list and did not find a suitable circuit $C_i^{(1)}$, we abort.

Analysis: For each circuit, with probability at least $1 - 2^{-O(i')}$ over the sampled inputs, we correctly estimate its agreement with $f_{2i}^{\mathsf{GL(ws)}}$ up to error $2^{-2i'-1}$. Union-bounding over the $2^{O(i')}$ circuits, with probability at least $1 - 2^{-O(i')}$, in this step we obtained a circuit $C_i^{(1)}$ that has agreement at least $1/2 + 2^{-2i'}$ with $f_{2i}^{\mathsf{GL(ws)}}$.

3. Conversion to a probabilistic circuit that computes f_i^{ws} with success $\mathrm{poly}(\delta_0)$. We use the algorithm of Goldreich and Levin [21] to convert the deterministic circuit $C_i^{(1)}$ into a probabilistic circuit $C_i^{(2)} : \{0,1\}^i \to \{0,1\}^i$ of size $2^{O(i')}$ such that $\Pr[C_i^{(2)}(x) = f_i^{\mathsf{ws}}(x)] \geq 2^{-O(i')}$, where the probability is taken both over a random choice of $x \in \{0,1\}^i$ and over the internal randomness of $C_i^{(2)}$. Specifically, the circuit $C_i^{(2)}$ gets input $x \in \{0,1\}^i$, and simulates the algorithm from [19, Theorem 7.8] with parameter $\delta_0 = 2^{-2i'}$, while resolving the oracle queries of the algorithm using the circuit $C_i^{(1)}$; then, the circuit $C_i^{(2)}$ outputs a random element from the list that is produced by the algorithm from [19].

Analysis: Since $\mathbb{E}_x[\Pr_r[C_i^{(1)}(x, r) = f_{2i}^{\mathsf{GL(ws)}}(x, r)]] \geq 1/2 + \delta_0$, it follows that for at least $\delta_0/2$ of the inputs $x \in \{0,1\}^i$ it holds that $\Pr_r[C_i^{(1)}(x, r) = f_{2i}^{\mathsf{GL(ws)}}(x, r)] \geq 1/2 + \delta_0/2$. For each such input x, with probability at

least $1/2$ the algorithm of [21] outputs a list of size $\mathrm{poly}(1/\delta_0)$ that contains $f_i^{\mathsf{ws}}(x)$, and thus the circuit $C_i^{(2)}$ outputs $f_i^{\mathsf{ws}}(x)$ with probability $\mathrm{poly}(\delta_0)$.

4. Fixing randomness for the probabilistic circuit. For $t = 2^{O(i')}$ attempts we choose a random string for $C_i^{(2)}$, hard-wire it into the circuit, and estimate the agreement between the resulting deterministic circuit and f_i^{ws}, with an additive error of $\delta_1 = \mathrm{poly}(\delta_0)$ and confidence $1 - 1/\mathrm{poly}(t)$. (The estimation in each attempt is done using random sampling of inputs, the downward self-reducibility of f_i^{ws} and the circuit C_{i-1}, similarly to Step 2.) We proceed to the next step if in one of these attempts yields a deterministic circuit that (according to our estimations) agrees with f_i^{ws} on at least $2\delta_1$ of the inputs.

Analysis: With probability at least $1 - \exp(-i')$ at least one choice of random string yields a deterministic circuit that agrees with f_i^{ws} on at least $3\delta_1$ of the inputs, and with probability at least $1 - \exp(-i')$ all of our t estimates are correct up to an additive error of δ_1. Thus, with probability at least $1 - \exp(-i')$ we proceed to the next step with a deterministic circuit $C_i^{(3)}$ of size $2^{O(i')}$ that agrees with f_i^{ws} on $\delta_1 = 2^{-O(i')} = 2^{-O(i/\log(i)^{c+1})} > \delta(i)$ of the inputs.

5. Worst-case to δ-average-case reduction for f_i^{ws}. We use the sample-aided worst-case to δ-average-case reduction for f^{ws}, generating random labeled samples $(r, f_i^{\mathsf{ws}}(r))$ by using the downward self-reducibility of f^{ws} and the circuit C_{i-1} to compute $f_i^{\mathsf{ws}}(r)$.

Analysis: With probability at least $1 - \delta(i)$, the uniform reduction outputs a probabilistic circuit $C_i^{(4)}$ of size $\mathrm{poly}(1/\delta(i))$ such that for every $x \in \{0,1\}^i$ it holds that $\Pr_r[C_i^{(4)}(x, r) = f_i^{\mathsf{ws}}(x)] \geq 2/3$. [34]

6. Fixing randomness for the final circuit. Applying naive error-reduction to $C_i^{(4)}$, we obtain a circuit $C_i^{(5)}$ of size $\mathrm{poly}(1/\delta(i))$ that correctly computes f_i^{ws} at any input with probability $1 - 2^{-O(i)}$. Then we uniformly choose randomness for $C_i^{(5)}$ 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_i^{ws} correctly on all inputs.

Having proved Claim 4.9.2, this concludes the proof of Lemma 4.9. ■

In the last part of the proof of Lemma 4.9, after we converted a distinguisher for $f^{\mathsf{GL}(\mathsf{ws})}$ into a weak learner for $f^{\mathsf{GL}(\mathsf{ws})}$ (i.e., after Claim 4.9.3), we used the existence of the weak learner for $f^{\mathsf{GL}(\mathsf{ws})}$ on $2L_n$ to obtain a circuit that computes f^{ws} on L_n. This part of the proof immediately implies the following, weaker corollary. (The corollary 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^{ws} on a corresponding input length.)

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

[34] 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 $C_i^{(3)}$.

thus it cannot be that f^{ws} is hard almost-always.[35] However, since TQBF can be reduced to f^{ws} with a quasilinear blow-up $b : \mathbb{N} \to \mathbb{N}$, we can still deduce the following: If TQBF is "hard" almost-always, then for every $n \in \mathbb{N}$ there exists $n' \leq b(n)$ such that f^{ws} is "hard" on input length n' (i.e., this holds for the smallest $n' \geq n$ 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 $f^{\mathsf{ws}} \Rightarrow$ "almost everywhere" derandomization of $\mathcal{RP}$ "on average"). *Assume that for some constant $c \in \mathbb{N}$ and for $\delta(n) = 2^{-n/\log(n)^c}$ 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^{ws} on input length $\overline{n} = \min\{b(n_0) \geq n : n_0 \in \mathbb{N}\}$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\log\log(n)^k}$ 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^{ws}, the parameter $2k$ instead of k (i.e., the parameter t in Lemma 4.9 is $t(n) = n^{\log\log(n)^{2k}}$) and with $\mathtt{str}(n) = n$ ($\mathtt{str}$ 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^{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 $\text{polyloglog}(n)$ 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'.*

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

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^{ws} on input length $\overline{\ell}(\overline{n}) = \overline{\ell}(n)$, which contradicts our hypothesis. □

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 $\tilde{O}(\log(n^{\text{polyloglog}(n)})) + \log(|\overline{S_n}|) = \tilde{O}(\log(n))$, and that H is computable in time at most $n^{\text{polyloglog}(n)}$.

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

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

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 $f^{\mathsf{ws}} \Rightarrow$ "almost everywhere" derandomization of $\mathcal{BPP}$ "on average" with short advice). *Assume that for some constant $c \in \mathbb{N}$ and for $\delta(n) = 2^{-n/\log(n)^c}$ 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 $2^{O(n/\log(n)^c)}$, and every sufficiently large $n \in \mathbb{N}$, the algorithm fails to compute f^{ws} on input length $\overline{n} = \min\{b(n_0) \geq n : n_0 \in \mathbb{N}\}$.*

For $k \in \mathbb{N}$ and $t(n) = n^{\log\log(n)^k}$, 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^{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^{ws} on input length $\overline{\ell}(n)$. This contradicts our hypothesis regarding f^{ws}. □

The machine D gets input $x \in \{0,1\}^n$ and advice of length $O(\text{loglogloglog}(n))$, 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.[36] 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 $L^{\mathsf{pad}} = \{(x, 1^{t(|x|)}) : x \in L\}$, and note that L^{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^{pad} there exists an algorithm D^{pad} that runs in time $t_{D^{\mathsf{pad}}}(m) = m^{\text{polyloglog}(m)}$ such that for all sufficiently large $m \in \mathbb{N}$ it holds that $\Pr_{z \leftarrow F^{\mathsf{pad}}(1^m)}[D^{\mathsf{pad}}(z) \neq L^{\mathsf{pad}}(z)] \leq 1/t(m)$.

We define the algorithm D in the natural way, i.e. $D(x) = D^{\mathsf{pad}}(x, 1^{t(|x|)})$, and note that this algorithm runs in time $n^{\text{polyloglog}(n)}$. 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^{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^{pad} fails and halts), and let $B_{F^{\mathsf{pad}}} = \{n + t(n) : n \in B_F\}$. For any $m \in B_{F^{\mathsf{pad}}}$ we have that

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

which yields a contradiction. ■

Remark 4.13 (A PRG that runs in quasilogarithmic space). The PRG constructed in Lemma 4.9 actually works in *quasilogarithmic space* (since f^{ws} is computable in linear space), except for one crucial part: The construction of combinatorial designs. Combinatorial designs with parameters as in our proof actually *can* be constructed in logarithmic space, but these combinatorial designs work only for values of ℓ that are of a specific form (since the constructions are algebraic).[37] However, in our downward self-reducibility argument we need

[36] 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).

[37] This can be done using an idea from [29, Lemma 5.5] (attributed to Salil Vadhan), essentially "composing" Reed-Solomon codes over $GF(n)$ of degree $n/\text{polylog}(n)$ with standard designs (a-la Nisan and Wigderson [44]; see [29, Lemma 2.2]) with set-size $\ell = \text{polylog}(n)$.

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 $f^{\mathsf{GL}(\mathsf{ws})}$ 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 4.14 (rETH $\Rightarrow$ i.o.-PRG for uniform circuits). *Assume that there exists $i \geq 1$ such that $\mathsf{TQBF} \notin \mathcal{BPTIME}[2^{n/\log(n)^i}]$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\log\log(n)^k}$ 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 $(\delta, O(\log(n)^{6c}))$-well-structured function f^{ws} that is computable in linear space, and such that TQBF reduces to f^{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^{ws} cannot be computed in probabilistic time $2^{n/\log(n)^{3c-1}} > 2^{O(n/\log(n)^{3c})}$; this is the case since otherwise, 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 4.15 (a.a.-rETH $\Rightarrow$ almost-always HSG for uniform circuits and almost-always "average-case" derandomization of $\mathcal{BPP}$). *Assume that there exists $i \geq 1$ such that $\mathsf{TQBF} \notin$ i.o.-$\mathcal{BPTIME}[2^{n/\log(n)^i}]$. Then, for every $k \in \mathbb{N}$ and for $t(n) = n^{\log\log(n)^k}$:*

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 $1/t(n)$.*

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 4.14, let $\delta(n) = 2^{-n/\log(n)^{3c}}$ for a sufficiently large constant $c \in \mathbb{N}$, and let f^{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 $2^{n/\log(n)^{3c-1}} > 2^{O(n/\log(n)^{3c})}$ and every sufficiently large $n_0 \in \mathbb{N}$, the algorithm fails to compute f^{ws} on input length $n = \mathtt{ql}(n_0)$; this is because otherwise we could have computed TQBF on infinitely-often n_0's in time $2^{n/\log(n)^{c-r-1}} \leq 2^{n_0/\log(n_0)^k}$, where the calculation is as in Eq. (4.4). This implies the hypothesis of Propositions 4.11 and 4.12. ∎

5 NETH AND THE EQUIVALENCE OF DERANDOMIZATION AND CIRCUIT LOWER BOUNDS

In this section we prove Theorems 1.4, 1.5, and 1.6. Recall that these results 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 5.1 ($\mathcal{NTIME}[T]$-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 $C\colon \{0,1\}^n \to \{0,1\}$ of size at most $S(n)$ that decides $L_n = L \cap \{0,1\}^n$.*
(2) *For every $n \in \mathbb{N}$ and non-deterministic choices, $M(1^n)$ either outputs a circuit $C\colon \{0,1\}^n \to \{0,1\}$ that decides L_n, or outputs $\perp$.*

*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 $S(n) = T(n)$.*

The class $\mathcal{ONTIME}[T]$, which was defined in [17, 22] and stands for "oblivious $\mathcal{NTIME}[T]$", consists of all sets decidable by non-deterministic time-T machines such that for every input length $n \in \mathbb{N}$ there exists a *single* witness w_n that convinces the non-deterministic machine on *all* n-bit inputs in the set. As mentioned in Section 2.2, the class of problems decidable by $\mathcal{NTIME}[T]$-uniform circuits is a subclass of $\mathcal{ONTIME}[T]$, which is in turn a subclass of $\mathcal{NTIME}[T] \cap \mathcal{SIZE}[T]$. That is:

Fact 5.2. *For $T : \mathbb{N} \to \mathbb{N}$, if $L \subseteq \{0,1\}^*$ can be decided by $\mathcal{NTIME}[T]$-uniform circuits, then $L \in \mathcal{ONTIME}[T'] \subseteq (\mathcal{NTIME}[T'] \cap \mathcal{SIZE}[T'])$, for $T'(n) = \tilde{O}(T(n))$.*

Proof. Fix L, and let M be a non-deterministic machine that uniformly constructs circuits for L as in Definition 5.1. For every $n \in \mathbb{N}$, let $w_n \in \{0,1\}^{T(n)}$ be non-deterministic choices such that $M(1^n, w_n)$ is a circuit for L_n. Then, L can be decided by a non-deterministic machine that gets input $x \in \{0,1\}^n$ and witness w_n, constructs a circuit for L_n using w_n, and evaluates this circuit at input x. The same witness w_n leads this non-deterministic machine to accept all $x \in L_n$, and the running time is quasilinear in the size of the circuit (i.e., in T). ∎

Since we will be repeating some technical non-degeneracy conditions on functions throughout the section, let us define these conditions concisely at this point:

Definition 5.3 (size functions and time functions). *We say that $S\colon \mathbb{N} \to \mathbb{N}$ is a* size function *if S is time-computable, increasing, satisfies $S(n) = o(2^n/n)$, and for every $n \in \mathbb{N}$ satisfies $S(n) > n$ and $S(n+1) \leq 2S(n)$. We say that $T : \mathbb{N} \to \mathbb{N}$ is a* time function *if T is time-computable, increasing, and for every $n \in \mathbb{N}$ satisfies $T(n) > n$.*

We will first prove, in Section 5.1, the key technical results that underlie the main theorems; these technical results will be strengthenings of classical Karp-Lipton style theorems. Then, in Section 5.2, we will prove Theorems 1.4, 1.5, and 1.6.

5.1 Strengthened Karp-Lipton style results

Recall that Babai *et al.* [3] proved that if $\mathcal{EXP} \subset \mathcal{P}/\text{poly}$ then $\mathcal{EXP} = \mathcal{MA}$; if we also use an additional hypothesis that $pr\mathcal{BPP} = pr\mathcal{P}$, then we can deduce the stronger conclusion $\mathcal{EXP} = \mathcal{NP}$. In the current section we will prove two strengthenings of this result, which further strengthen the foregoing conclusion: Instead of deducing that $\mathcal{EXP} = \mathcal{NP}$, we will deduce that $\mathcal{EXP}$ can be decided by $\mathcal{NTIME}[T]$-uniform circuits of size S, for small values of T, S.

We first prove, in Section 5.1.1 a lemma that will be used in one of our proofs; we present this lemma and the underlying question in a separate section since they might be of independent interest. The two strengthened Karp-Lipton style results will be subsequently proved in Sections 5.1.2 and 5.1.3, respectively.

5.1.1 Solving $(1, 1/3)$-CAPP *using many untrusted* CAPP *algorithms.* Recall that in the problem (α, β)-CAPP, we get as input a description of a circuit, and our goal is to distinguish between circuits with acceptance probability at least $\alpha > 0$ and circuits with acceptance probability at most $\beta > 0$; we also denote CAPP $= (2/3, 1/3)$-CAPP (see Definition 3.1). Assume that we want to solve CAPP on an input circuit C of description length n, and that we are guaranteed that an algorithm A solves CAPP on *some* input length (unknown to us) in the interval $[n, S(n)]$, for some function S. This problem arises, for example, if we assume that $pr\mathcal{BPP} \subset \mathtt{i.o.}pr\mathcal{NP}$ (which implies that CAPP $\in \mathtt{i.o.}pr\mathcal{NP}$), and want to derandomize $\mathcal{MA}$ infinitely-often. (This is because when the $\mathcal{MA}$ verifier gets an input of length m, the derandomization of the verifier corresponds to a CAPP problem on some input length $n = m^k$, but we are not guaranteed that the CAPP algorithm works on input length n.)[38] How can we solve this problem?

If we invoke the algorithm A on each input length in the interval $[n, S(n)]$, while feeding it C as input each time (i.e., C is padded up to the appropriate length), then we obtain a variety of answers, and it is not clear a-priori how we can distinguish the correct answer from possibly-misleading ones. In this section we show a solution for this problem in the setting where we only need to solve CAPP with *one-sided error*, and when A solves a problem in $pr\mathcal{BPP}$ that slightly generalizes CAPP. Intuitively, since we only need to solve $(1, 1/3)$-CAPP, it will be possible to *prove* to us that C is *not* a YES instance (i.e., that C does not accept all of its inputs); and since A solves a problem that slightly generalizes CAPP, we will be able to modify it to an algorithm that is able to provide such a proof when C is not a YES instance. Details follow.

We first define the aforementioned variation of (α, β)-CAPP, denoted pCAPP (for "parametrized CAPP"), in which α and β are specified as part of the input.

Definition 5.4 (parametrized CAPP). *In the promise problem* pCAPP$[S, \ell]$*, the input is a triplet* (C, α, β)*, where* C *is a Boolean circuit over* v *variables and of size* $S(v)$ *and* $1 > \alpha > \beta > 0$ *are rational numbers specified with* $\ell(v)$ *bits. The YES instances are such that* $\Pr_x[C(x) = 1] \geq \alpha$ *and the NO instances are such that* $\Pr_x[C(x) = 1] \leq \beta$*.*

Note that if $\ell(v) = O(\log(S(v)))$, then pCAPP$[S, \ell] \in pr\mathcal{BPP}$. (This is since we can uniformly sample ϵ^{-2} inputs for C, where $\epsilon = \beta - \alpha \geq 1/\mathrm{poly}(S(v))$, and estimate $\Pr_x[C(x) = 1]$ with accuracy $(\alpha - \beta)/2$, with high probability). We now show that solving $(1, 1/3)$-CAPP for circuits of size $S(n)$ infinitely-often reduces to solving pCAPP infinitely-often (i.e., on an arbitrary infinite set of input lengths).

Lemma 5.5 (solving CAPP with one-sided error on a fixed input length reduces to solving pCAPP on an unknown "close" input length). *For any two size functions* $S^{(n)}, S^{(v)} : \mathbb{N} \to \mathbb{N}$ *and time function* $T : \mathbb{N} \to \mathbb{N}$*, assume that* pCAPP$[S^{(v)}, \ell] \in \mathtt{i.o.}\mathcal{DTIME}[T]$*, where* $\ell(v) = 4 \cdot \log(v)$*. Then, there exists an algorithm* $M^{co\mathcal{RP}}$ *that for infinitely-many values of* $n \in \mathbb{N}$*, when given as input* $(1^n, C)$ *such that* C *a* v*-bit circuit of size at most* $\max\left\{S^{(n)}(n), S^{(v)}(v)\right\}$*, the algorithm* $M^{co\mathcal{RP}}$ *solves* $(1, 1/3)$-CAPP *on* C *in time* $\mathrm{poly}(n) \cdot v \cdot \tilde{O}(S(n)) \cdot T(\tilde{O}(S^{(n)}(n)))$*.*

Proof. Let $\mathtt{ql}(S) = \tilde{O}(S)$ such that circuits of size S can be described by strings of length $\mathtt{ql}(S)$. For any $n \in \mathbb{N}$, we consider inputs of length $S^{(n)}(n)$ that describe v-bit circuits of size $S^{(v)}(v)$. Let $I_n = [2\mathtt{ql}(S^{(n)}(n)), 2\mathtt{ql}(S^{(n)}(n+1)) - 1]$, and note that any sufficiently large integer belongs to a unique interval I_n. Let M^{pCAPP} be a time-T algorithm that solves pCAPP$[S^{(v)}, \ell]$ infinitely-often. We will use M^{pCAPP} to construct the following search algorithm:

Claim 5.5.1 (search-to-decision reduction that preserves the input length). *There exists an algorithm* F *that gets as input* $(1^n, C, m)$*, where* C *is a* v*-bit circuit of size at most* $\max\left\{S^{(n)}(n), S^{(v)}(v)\right\}$ *and* $m \in I_n$*, runs in time* $\mathrm{poly}(n) \cdot v \cdot T(m)$*, and if* M^{pCAPP} *correctly solves* pCAPP$[S^{(v)}, \ell]$ *on input length* m *and* $\Pr_x[C(x) = 1] \leq 1/3$ *then* $F(1^n, C, m) \in C^{-1}(0)$*.*

[38] Also, in this setting the function S represents "how far ahead" (beyond n) we are willing to look in our search for the "good" input length.

Proof. In the following we will construct a set of m-bit inputs and run M^{pCAPP} on each of those inputs. Since all of our inputs will be of the form (C, α, β) where α and β can be specified with $4 \cdot \log(v)$ bits, each input will be of size less than $2\mathtt{ql}(S^{O(n)}(n)) \leq m$; we will therefore pad each input to be of length exactly m.

First we run M^{pCAPP} on input $(C, 1/2, 1/3)$, and if M^{pCAPP} accepts then we output 0^v. Otherwise, when M^{pCAPP} rejected, we have that $\Pr_x[C(x) = 1] \leq 1/2$; in this case our goal will be to construct a string in $C^{-1}(0)$, bit-by-bit. Let $\neg C$ be the circuit that computes C and negates the output, let σ_0 be the empty string, and for $i \in [v]$, in iteration i we act as follows:

(1) We start with a prefix $\sigma_{i-1} \in \{0,1\}^{i-1}$, and with the guarantee that the circuit $\neg C_{\sigma_{i-1}}$, which is obtained by fixing the first $i-1$ input variables of $\neg C$ to σ_{i-1}, satisfies $\Pr_x[\neg C_{\sigma_{i-1}}(x) = 1] \geq 1/2 - (i-1) \cdot v^{-2}$.
(2) We run M^{pCAPP} at input $(\neg C_{\sigma_{i-1}0}, 1/2 - (i-1) \cdot v^{-2}, 1/2 - i \cdot v^{-2})$. If M^{pCAPP} accepts then we define $\sigma_i = \sigma_{i-1}0$, and otherwise we define $\sigma_i = \sigma_{i-1}1$.
(3) To see that the guarantee on $\neg C_{\sigma_i}$ is preserved for iteration $i+1$, note that if M^{pCAPP} accepted then $\Pr_x[\neg C_{\sigma_i}(x) = 1] > 1/2 - i \cdot v^{-2}$; and otherwise we have that $\Pr_x[\neg C_{\sigma_{i-1}1}(x) = 0] \leq 1/2 - (i-1) \cdot v^{-2}$, which implies (by the guarantee on $\neg C_{\sigma_{i-1}}$ from the beginning of the iteration) that $\Pr_x[\neg C_{\sigma_i}(x) = 1] \geq 1/2 - (i-1) \cdot v^{-2}$.

After the v iterations we have that $\Pr_x[\neg C_{\sigma_i}(x) = 1] > 0$, and therefore $\sigma_i \in (\neg C^{-1})(1) = C^{-1}(0)$ and we output σ_i. The running time of each iteration is $\text{poly}(n) \cdot v \cdot T(m)$. □

Our algorithm $M^{co\mathcal{RP}}$ runs F at inputs $\{(1^n, C, k)\}_{k \in I_n}$, and evaluates C at the outputs of F; if for some $k \in I_n$ it holds that $C(F(C,k)) = 0$ then $M^{co\mathcal{RP}}$ rejects, and otherwise $M^{co\mathcal{RP}}$ accepts. The running time of $M^{co\mathcal{RP}}$ is $\text{poly}(n) \cdot v \cdot T(2\mathtt{ql}(S^{(n)}(n+1))) \cdot |I_n| = \text{poly}(n) \cdot \tilde{O}(S^{(n)}(n)) \cdot v \cdot T(\tilde{O}(S^{(n)}(n)))$.

Now, fix $n \in \mathbb{N}$ such that for some $m \in I_n$ it holds that M^{pCAPP} decides $\text{pCAPP}[S^{(v)}, \ell]$ on inputs of length m. To see that $M^{co\mathcal{RP}}$ correctly solves $(1, 1/3)$-CAPP on an input circuit C over v bits of size at most $\max\left\{S^{(n)}(n), S^{(v)}(v)\right\}$, note that if C accepts all its inputs then $M^{co\mathcal{RP}}$ always accepts C; and if C accepts at most $1/3$ of its inputs then for the "good" $m \in I_n$ it holds that $F(1^n, C, m) \in C^{-1}(0)$, in which case $M^{co\mathcal{RP}}$ rejects. ■

5.1.2 A strengthened Karp-Lipton style result for the "low-end" setting. To prove our first strengthening of [3], let $L \in \mathcal{EXP}$, and note that by our assumption $L \in \mathcal{P}/\text{poly}$. Consider an $\mathcal{MA}$ verifier V that gets input 1^n, guesses a circuit $C_L : \{0,1\}^n \to \{0,1\}$, and tries to decide if C_L correctly computes $L_n = L \cap \{0,1\}^n$. The key observation is that since this decision problem (of deciding whether or not a given n-bit circuit computes L_n) is in $\mathcal{EXP}$, we can apply the original Karp-Lipton style result of [3] to it. The latter result implies that there exists an $\mathcal{MA}$ verifier M that decides whether or not C_L computes L_n correctly. Our verifier V guesses C_L and a witness for M, simulates M, and if M confirms that C_L computes L_n then V outputs C_L.

We will derandomize the foregoing $\mathcal{MA}$ verifier in one of two ways. The first relies on a hypothesis of the form $pr\mathcal{BPP} \subseteq pr\mathcal{NSUBEXP}$, which immediately implies that $\mathcal{MA} \subseteq \mathcal{NSUBEXP}$. The second relies on a hypothesis of the form $pr\mathcal{BPP} \subset \mathtt{i.o.}pr\mathcal{SUBEXP}$; in this case we derandomize the $\mathcal{MA}$ verifier infinitely-often, relying on the fact that the $\mathcal{MA}$ verifier can be assumed to have perfect completeness [18] and on Lemma 5.5 (which was presented in Section 5.1.1). Note that in both cases, the running time of the resulting non-deterministic machine is sub-exponential, but the size of the output circuit C_L is nevertheless still polynomial.

The following statement and proof generalize the above, using parametrized "collapse" and derandomization hypotheses. Specifically, if we assume that $\mathcal{E} \subset \mathcal{SIZE}[S]$ and that $pr\mathcal{BPP}$ can be derandomized in time T, we deduce that $\mathcal{E}$ has $\mathcal{NTIME}[T']$ uniform circuits of size $S(n)$, where $T'(n) \approx T(S(S(n)))$.

Proposition 5.6 (a strengthened "low-end" Karp-Lipton style result). *There exist two constants $k, k' > 1$ such that for any size function $S : \mathbb{N} \to \mathbb{N}$ and time function $T : \mathbb{N} \to \mathbb{N}$ satisfying $T(n) \geq n^{k'}$ the following holds. Let $T'(n) = T\left(\bar{S}(n)\right)^{O(1)}$ where $\bar{S}(n) = \tilde{O}(S(\tilde{O}(S(n))))$.*

(1) *If* $\mathcal{DTIME}[2^n] \subset \mathcal{SIZE}[S]$ *and* $\mathrm{pCAPP}[v^k \cdot S(v), 4 \cdot \log(v)] \in \mathtt{i.o.}pr\mathcal{DTIME}[T]$*, then any* $L \in \mathcal{DTIME}[2^n]$ *can be decided on infinitely-many input lengths by* $\mathcal{NTIME}[T']$*-uniform circuits of size* $S(n)$*.*
(2) *If* $\mathcal{DTIME}[2^n] \subset \mathcal{SIZE}[S]$ *and* $(1, 1/3)\text{-CAPP}[v^k \cdot S(v)] \in pr\mathcal{NTIME}[T]$*, then any* $L \in \mathcal{DTIME}[2^n]$ *can be decided (on all input lengths) by* $\mathcal{NTIME}[T']$*-uniform circuits of size* $S(n)$*.*

Proof. We first prove Item (1). Fix $L \in \mathcal{DTIME}[2^n]$, and recall that by our hypothesis $L \in \mathcal{SIZE}[S]$. We define a corresponding problem L-$\mathtt{Ckts}$ as the set of size-S circuits that decide L; that is, denoting by $\mathtt{ql}(S) = \tilde{O}(S)$ the description length of size-S circuits, on inputs of length $N = n + \mathtt{ql}(S(n))$ we define L-$\mathtt{Ckts}$ by

$$L\text{-}\mathtt{Ckts}_N = \{(1^n, C) : |C| = \mathtt{ql}(S(n)) \wedge \forall x \in \{0,1\}^n, C(x) = L(x)\} \ ,$$

and on inputs of length N that cannot be parsed as $N = n + \mathtt{ql}(S(n))$ we define L-$\mathtt{Ckts}$ trivially. Note that $L\text{-}\mathtt{Ckts} \in \mathcal{DTIME}[2^N]$, since we can enumerate the $2^n < 2^{o(N)}$ inputs, and for each $x \in \{0,1\}^n$ compute $C(x)$ and $L(x)$ in time $2^n + \mathrm{poly}(|C|) < 2^{o(N)}$.

Given input 1^n, we first guess a circuit $C_n^{(L)}$ of size $S(n)$, in the hope that $C_n^{(L)}$ decides L_n; note that a suitable circuit exists by our hypothesis. Now we consider the problem of deciding if $x = (1^n, C_n^{(L)}) \in L\text{-}\mathtt{Ckts}$, where $x \in \{0,1\}^{N=n+\mathtt{ql}(S(n))}$. Since $L\text{-}\mathtt{Ckts} \in \mathcal{DTIME}[2^N]$, we can reduce L-$\mathtt{Ckts}$ to the problem $L^{\mathtt{nice}}$ from Proposition 3.12; that is, we compute in time $\mathrm{poly}(N)$ an input $x' \in \{0,1\}^{N'=O(N)}$ for $L^{\mathtt{nice}}$ such that $x \in L\text{-}\mathtt{Ckts} \iff x' \in L^{\mathtt{nice}}$.

Now, let $\bar{N} = \ell(N') = O(N)$, where ℓ is the query length of the instance checker $\mathtt{IC}$ for $L^{\mathtt{nice}}$. We guess another circuit, which is of size $S(2\bar{N})$ and denoted $C_{\bar{N}}^{L^{\mathtt{nice}}} : \{0,1\}^{\bar{N}} \to \{0,1\}$, in the hope that $C_{\bar{N}}^{L^{\mathtt{nice}}}$ decides $L_{\bar{N}}^{\mathtt{nice}}$; again, a suitable circuit exists by our hypothesis.[39] We then construct a circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}} : \{0,1\}^{O(\bar{N})} \to \{0,1\}$ that computes the decision of $\mathtt{IC}$ at input x' and with oracle $C_{\bar{N}}^{L^{\mathtt{nice}}}$, as a function of the $O(\bar{N})$ random coins of $\mathtt{IC}$, and maps the outputs $\{0, \perp\}$ of $\mathtt{IC}$ to 0, and the output 1 of $\mathtt{IC}$ to 1.

Note that the circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ is over $v = O(\bar{N})$ input bits and of size $S^{(n)}(n) \stackrel{\mathrm{def}}{=} \mathrm{poly}(N) \cdot S(2\bar{N})$. Also, measuring the size of $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ as a function of its number of input bits (i.e., of v), the size is upper-bounded by $S^{(v)}(v) \stackrel{\mathrm{def}}{=} v^k \cdot S(v)$, where $k \in \mathbb{N}$ is a sufficiently large universal constant (and we assume without loss of generality that $v \geq 2\bar{N}$). By the properties of the instance checker, and using the fact that a suitable circuit $C_{\bar{N}}^{L^{\mathtt{nice}}}$ for $L_{\bar{N}}^{\mathtt{nice}}$ exists, we have that:

(1) If $C_n^{(L)}$ decides L then $x' \in L^{\mathtt{nice}}$, and hence for some guess of $C_{\bar{N}}^{L^{\mathtt{nice}}}$ the circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ will have acceptance probability one.
(2) If $C_n^{(L)}$ does not decide L then $x' \notin L^{\mathtt{nice}}$, and hence for all guesses of $C_{\bar{N}}^{L^{\mathtt{nice}}}$ the circuit $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ accepts at most $1/6$ of its inputs.

Using our hypothesis about pCAPP and Lemma 5.5, there exists an algorithm $M^{co\mathcal{RP}}$ that for infinitely-many values of $n \in \mathbb{N}$ gets input $(1^n, \mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}})$ and solves $(1, 1/3)$-CAPP on $\mathtt{IC}_{x'}^{C_{\bar{N}}^{L^{\mathtt{nice}}}}$ in time $\mathrm{poly}(n) \cdot v \cdot \tilde{O}(S(n)) \cdot$

[39] To see this more formally, let $L^{\mathtt{pad}} = \left\{(x, 1^{O(\log(|x|))}) : x \in L^{\mathtt{nice}}\right\}$. Since $L^{\mathtt{nice}} \in \mathcal{DTIME}[\tilde{O}(2^n)]$, we have that $L^{\mathtt{pad}} \in \mathcal{DTIME}[2^n]$. Using our hypothesis, $L^{\mathtt{pad}}$ on inputs of length $N' = \bar{N} + O(\log(\bar{N}))$ has circuits of size $S(N')$, and these circuits can be converted (by hardwiring the last $N' - \bar{N}$ input bits) to $\bar{N}$-bit circuits for $L^{\mathtt{nice}}$ of size $S(N') < S(2\bar{N})$.

$T\left(\tilde{O}(S^{(n)}(n)\right)$. We run this algorithm on $(1^n, \mathtt{IC}^{C^{L^{\mathrm{nice}}}_{\tilde{N}}}_{x'})$, and if it accepts (i.e., asserts that the acceptance probability of $\mathtt{IC}^{C^{L^{\mathrm{nice}}}_{\tilde{N}}}_{x'}$ is larger than 1/3) then we output the circuit $C^{(L)}_n$; otherwise we output $\perp$.

Note that the size of the circuit that we output is $S(n)$, and that our running time is at most

$$\begin{aligned}\mathrm{poly}(n) \cdot v \cdot \tilde{O}(S(n)) \cdot T\left(\tilde{O}(S^{(n)}(n)\right) &= \mathrm{poly}(n) \cdot \tilde{O}(S(n))^2 \cdot T\left(\tilde{O}(S(\tilde{O}(S(n))))\right) \\ &\leq T\left(\tilde{O}(S(\tilde{O}(S(n))))\right)^{O(1)},\end{aligned}$$

where the last inequality relied on the fact that $T(n) \geq n^{k'}$ for a sufficiently large constant k'.

Let us now explain how to prove Item (2). We guess $C^{(L)}_n$ and $C^{L^{\mathrm{nice}}}_{\tilde{N}}$ and construct $\mathtt{IC}^{C^{L^{\mathrm{nice}}}_{\tilde{N}}}_{x'}$ as above. However, instead of using Lemma 5.5, we run the hypothesized non-deterministic $(1, 1/3)$-CAPP$[v^k \cdot S(v)]$ machine, denoted $M^{co\mathcal{RP}}$, on input $\mathtt{IC}^{C^{L^{\mathrm{nice}}}_{\tilde{N}}}_{x'}$ (the advantage in the current setting being that, in contrast to the proof of Item (1), the machine $M^{co\mathcal{RP}}$ is guaranteed to work on *all* input lengths). When $C^{(L)}_n$ decides L_n there are some non-deterministic choices that will cause $M^{co\mathcal{RP}}$ to accept, whereas when $C^{(L)}_n$ does not decide L_n, all non-deterministic choices will cause $M^{co\mathcal{RP}}$ to reject. Our running time is $T(\tilde{O}(S^{(n)}(n)))$, which can be bounded as above by $T\left(\tilde{O}(S(\tilde{O}(S(n))))\right)^{O(1)}$. ■

Note that in the proof of Proposition 5.6 we did not use the fact that L^{nice} is randomly self-reducible, but only the facts that L^{nice} is complete for $\mathcal{E}$ under linear-time reductions (such that all n-bit inputs are mapped to n'-bit inputs, for $n' = O(n)$) and that it has an instance checker with query length $\ell(n) = O(n)$.

5.1.3 A strengthened Karp-Lipton style result for the "high-end" setting. The result presented next asserts that if $\mathcal{E} \in \mathcal{SIZE}[S]$ and $pr\mathcal{BPP}$ can be derandomized in time T, then $\mathcal{E}$ has $\mathcal{NTIME}[T']$ uniform circuits (with a trivial size bound of $T'(n)$), where $T' \approx T(S(n))$. The main difference between this result and the result presented in Section 5.1.3, other than the differences in parameters, is that for this result we will need to assume that $pr\mathcal{BPP}$ can be derandomized *deterministically*, rather than only non-deterministically.

Let us briefly describe the proof idea. We construct a circuit for an $\mathcal{E}$-complete problem L^{nice} that has an instance checker and that is randomly self-reducible (see Section 3.5 for definitions and details). We guess a circuit $C^{L^{\mathrm{nice}}}$ for L^{nice}, which exists by our "collapse" hypothesis, and randomly check whether or not this circuit "convinces" the instance checker on almost all inputs; if it does, we instantiate the instance checker with $C^{L^{\mathrm{nice}}}$ as an oracle, to obtain a "corrupt" version of L^{nice}, denoted $\tilde{L}$. We then construct a probabilistic circuit C' that decides L^{nice}, with high probability, using the random self-reducibility of L^{nice} and oracle access to $\tilde{L}$.

Now, under the hypothesis $pr\mathcal{BPP} \subseteq pr\mathcal{DTIME}[T]$, we can derandomize the two probabilistic steps in the foregoing construction. Specifically, we derandomize the probabilistic verification that the circuit $C^{L^{\mathrm{nice}}}$ "convinces" the instance checker on almost all inputs, and we also derandomize the probabilistic circuit itself (i.e., we actually output a deterministic circuit that constructs the probabilistic circuit C' and applies a deterministic CAPP algorithm to C'). Details follow.

Proposition 5.7 (a strengthened "high-end" Karp-Lipton style result). *There exist two constants $k, k' > 1$ such that for any size function $S\colon \mathbb{N} \to \mathbb{N}$ and time function $T\colon \mathbb{N} \to \mathbb{N}$ the following holds. Assume that $\mathcal{DTIME}[2^n] \subset \mathtt{i.o.}\mathcal{SIZE}[S]$ and that* CAPP$[v^{k'} \cdot S(v)] \in pr\mathcal{DTIME}[T]$. *Then any $L \in \mathcal{DTIME}[2^n]$ can be decided on infinitely-many input lengths by $\mathcal{NTIME}[T']$-uniform circuits, where $T'(n) = \tilde{O}\left(T\left(n^k \cdot S(k \cdot n)\right)\right)$.*

Note that the actual hypothesis of Proposition 5.7 is weaker than the hypothesis $prBPP \in prDTIME[T]$, since we only require an algorithm for CAPP for *large* circuits (i.e., for v-bit circuits of size $\mathrm{poly}(v) \cdot S(v)$).

Proof of Proposition 5.7. Fixing any $L \in DTIME[2^n]$, we prove that there exist $NTIME[T']$-uniform circuits that solve L infinitely-often. In what follows, it will be important to distinguish between the non-deterministic machine M, and the deterministic circuit $C\colon \{0,1\}^n \to \{0,1\}$ that M constructs. The machine M gets input 1^n and constructs C as follows.

Step 1: Reduce L to L^{nice}. As its first step, the circuit C computes the linear-time reduction from L to the problem L^{nice} from Proposition 3.12; that is, C maps its input $x \in \{0,1\}^n$ into $x' \in \{0,1\}^{n'}$, where $n' = O(n)$, such that $x \in L$ if and only if $x' \in L^{\mathsf{nice}}$.

Step 2: Guess-and-verify a circuit for $L^{\mathsf{nice}}_{\bar{n}}$. Let $\mathtt{IC}$ be the instance checker for L^{nice} and let $\bar{n} = \ell(n')$ be the length of queries that $\mathtt{IC}$ makes to its oracle on inputs of length n'.

Claim 5.7.1. *For infinitely-many input lengths n there exists a circuit $C^{L^{\mathsf{nice}}}_{\bar{n}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ that decides $L^{\mathsf{nice}}_{\bar{n}}$.*

Proof. For every $n \in \mathbb{N}$ let $I_n = [2\alpha \cdot n, 2\alpha \cdot (n+1) - 1]$, where $\alpha \in \mathbb{N}$ is the constant such that $\bar{n} = \ell(n') = \alpha \cdot n$. Note that every sufficiently large integer $m \in \mathbb{N}$ belongs to a unique interval I_n (i.e., $n = \lfloor m/2\alpha \rfloor$). We define L' to be the language that on input length $m \in I_n$ considers only its first $\bar{n} = \alpha \cdot n$ input bits and decides $L^{\mathsf{nice}}_{\bar{n}}$ on those input bits. Since L' on input length m can be decided in time $\tilde{O}(2^n) < 2^m$, by our hypothesis there exist an infinite set $\mathcal{M} \subseteq \mathbb{N}$ of input lengths such that for every $m \in \mathcal{M}$ there exist size-$S(m)$ circuits for L'_m. For every such $m \in I_n$, we hard-wire the last $m - \bar{n}$ input bits (to be all-zeroes), and obtain a circuit of size $S(m) < S(4\alpha \cdot n) = S(4\bar{n})$ that decides $L^{\mathsf{nice}}_{\bar{n}}$. □

Thus, if n is one of the infinitely-many input lengths mentioned in Claim 5.7.1, then there exists $C^{L^{\mathsf{nice}}}_{\bar{n}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ that decides $L^{\mathsf{nice}}_{\bar{n}}$. The machine M non-deterministically guesses such a circuit. We define the corruption of $C^{L^{\mathsf{nice}}}_{\bar{n}}$ by

$$\mathtt{Crpt}(C^{L^{\mathsf{nice}}}_{\bar{n}}) = \Pr_{z \in \{0,1\}^{n'}}\left[\Pr[\mathtt{IC}^{C^{L^{\mathsf{nice}}}_{\bar{n}}}(z) = \perp] > 1/6\right],$$

where the internal probability is over the random choices of the machine $\mathtt{IC}$. Let $\mathtt{Dec}$ be the machine underlying the random self-reducibility of L^{nice}, and let $c \in \mathbb{N}$ such that the number of queries that $\mathtt{Dec}$ makes on inputs of length n' is at most $(n')^c$. Consider the following promise problem Π:

- **The input** is guaranteed to be a circuit $C^{L^{\mathsf{nice}}}_{\bar{n}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$.
- **YES instances:** The circuit $C^{L^{\mathsf{nice}}}_{\bar{n}}$ decides $L^{\mathsf{nice}}_{\bar{n}}$, in which case $\mathtt{Crpt}(C^{L^{\mathsf{nice}}}_{\bar{n}}) = 0$.
- **NO instances:** It holds that $\mathtt{Crpt}(C^{L^{\mathsf{nice}}}_{\bar{n}}) > (n')^{-2c}$.

Now, note that $\Pi \in pr\text{-}co\mathcal{RP}$, since a probabilistic algorithm that gets $C^{L^{\mathsf{nice}}}_{\bar{n}}$ as input can decide whether $C^{L^{\mathsf{nice}}}_{\bar{n}}$ is a YES instance or a NO instance by sampling z's and estimating $\Pr\left[\mathtt{IC}^{C^{L^{\mathsf{nice}}}_{\bar{n}}}(z) = \perp\right]$ for each z. Moreover, using the sampler from Theorem 3.5, there is a probabilistic $co\mathcal{RP}$ algorithm for Π that on input $C^{L^{\mathsf{nice}}}_{\bar{n}}\colon \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ uses $m = O(n)$ random bits and runs in time $\mathrm{poly}(n) \cdot S(4\bar{n})$. [40]

[40] Specifically, the algorithm uses the sampler from Theorem 3.5 (with a sufficiently large $\beta, \gamma > 1$ and sufficiently small $\alpha > 0$) to sample $D = \mathrm{poly}(n)$ strings $z_1, \ldots, z_D \in \{0,1\}^{n'}$, and then uses this sampler again to sample D strings $r_1, \ldots, r_D \in \{0,1\}^{n+O(\log(n))}$ to be used as randomness for the machine $\mathtt{IC}$. The algorithm rejects $C^{L^{\mathsf{nice}}}_{\bar{n}}$ if and only if $\Pr_{i \in [D]}\left[\Pr_{j \in [D]}\left[\mathtt{IC}^{C^{L^{\mathsf{nice}}}_{\bar{n}}}(z, r_j) = \perp\right] \geq .01\right] \geq 1/2(n')^{-2c}$, where $\mathtt{IC}^{C^{L^{\mathsf{nice}}}_{\bar{n}}}(z, r_j)$ denotes the simulation of $\mathtt{IC}^{C^{L^{\mathsf{nice}}}_{\bar{n}}}(z)$ with the fixed randomness r_j. This algorithm always accepts YES instances.

Hence, the problem Π is reducible to an instance of $(1, 1/3)$-CAPP with a circuit C_Π on $v = O(n)$ input bits and of size $n^{O(1)} \cdot S(4\bar{n}) = v^{O(1)} \cdot S(v)$. The machine M runs the hypothesized $\mathsf{CAPP}[v^{k'} \cdot S(v)]$ algorithm on C_Π, which takes time $T\left(n^{O(1)} \cdot S(O(n))\right)$, and rejects iff the CAPP algorithm rejects. Thus, from now on we can assume that $C_{\bar{n}}^{L^{\mathsf{nice}}}$ is not a NO instance of Π, or in other words that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathsf{nice}}}) \leq (n')^{-2c}$.

Step 3: Transforming a non-corrupt $C_{\bar{n}}^{L^{\mathsf{nice}}}$ into a probabilistic circuit for L. Given that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathsf{nice}}}) \leq (n')^{-2c}$, the machine M now transforms $C^{L^{\mathsf{nice}}}$ into a probabilistic circuit C' that computes L. In high-level, the circuit C' simulates the random self-reducibility algorithm $\mathtt{Dec}$ for L, while resolving the random queries of $\mathtt{Dec}$ by instantiating the instance checker with oracle $C^{L^{\mathsf{nice}}}$. Details follow.

Lemma 5.7.2 (non-corrupt $C_{\bar{n}}^{L^{\mathsf{nice}}} \Rightarrow$ probabilistic circuit for L^{nice}). *There exists an algorithm that gets as input 1^n and a circuit $C_{\bar{n}}^{L^{\mathsf{nice}}} : \{0,1\}^{\bar{n}} \to \{0,1\}$ of size $S(4\bar{n})$ such that $\mathtt{Crpt}(C_{\bar{n}}^{L^{\mathsf{nice}}}) \leq (n')^{-2c}$, and outputs a probabilistic circuit $C' : \{0,1\}^{n'} \to \{0,1\}$ of size $\mathrm{poly}(n) \cdot S(4\bar{n})$ that uses $O(n)$ random coins such that for every $x' \in \{0,1\}^{n'}$, with high probability over choice of random coins r for C' it holds that $C'(x', r) = L^{\mathsf{nice}}(x')$.*

Proof. We consider an instantiation of $\mathtt{IC}$ on inputs of length n' and with oracle to $C_{\bar{n}}^{L^{\mathsf{nice}}}$, and as a first step we reduce the error of this algorithm. Let $m = O(n)$ be the number of random bits that $\mathtt{IC}$ uses on inputs of length n'. Consider the following probabilistic algorithm $\hat{\mathtt{IC}} : \{0,1\}^{n'} \to \{0,1,\perp\}$. Given input $z \in \{0,1\}^{n'}$, the algorithm $\hat{\mathtt{IC}}$ uses the sampler from Theorem 3.5, instantiated for output length m and with accuracy $1/n$, to obtain a sample of $D = \mathrm{poly}(n)$ strings $r_1, ..., r_D \in \{0,1\}^m$; then $\hat{\mathtt{IC}}$ outputs the majority vote among the values $\{v_i\}_{i \in [D]}$, where v_i is the output of $\mathtt{IC}$ when instantiated on input z with oracle $C_{\bar{n}}^{L^{\mathsf{nice}}}$ and fixed randomness r_i.

Note that $\hat{\mathtt{IC}}$ uses $O(n)$ random bits and runs in time $\mathrm{poly}(n) \cdot S(4\bar{n})$. We claim that there exists a set $G \subseteq \{0,1\}^{n'}$ of density $1 - (n')^{-2c}$ such that for every $z \in G$, with probability at least $1 - \exp(-n)$ over the randomness of $\hat{\mathtt{IC}}$ it holds that $\hat{\mathtt{IC}}(z) = L^{\mathsf{nice}}(z)$. To see this, let G be the set of z's such that $\Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathsf{nice}}}}(z) = \perp] \leq 1/6$, and recall that the density of G is at least $1 - (n')^{-2c}$. Note that for any $z \in G$ we have that $\Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathsf{nice}}}}(z) = L^{\mathsf{nice}}(z) \geq 2/3$, because $\Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathsf{nice}}}}(z) \neq L^{\mathsf{nice}}(z)] \leq \Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathsf{nice}}}}(z) = \perp] + \Pr[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathsf{nice}}}}(z) = \neg C_{\bar{n}}^{L^{\mathsf{nice}}}(z)] \leq 1/3$. Thus, for any fixed $z \in G$, the probability (over the random choices of $\hat{\mathtt{IC}}$) that the majority vote of the v_i's will not equal $L^{\mathsf{nice}}(z)$ is at most $\exp(-n)$.

Now, consider a probabilistic circuit $C' : \{0,1\}^{n'} \to \{0,1\}$ that chooses $O(n)$ random bits to be used as randomness for $\hat{\mathtt{IC}}$, and simulates the random self-reducibility algorithm $\mathtt{Dec}$ on its input $x' \in \{0,1\}^{n'}$, while answering its queries using the algorithm $\hat{\mathtt{IC}}$ with the fixed random bits chosen in advance. Note that the circuit C' is of size $\mathrm{poly}(n) \cdot S(4\bar{n})$. We claim that for every $x' \in \{0,1\}^{n'}$, with high probability $C'(x) = L^{\mathsf{nice}}(x')$. To see this, recall that $\mathtt{Dec}$ makes at most $(n')^c$ queries such that each query is uniformly-distributed, and thus the probability that all queries of $\mathtt{Dec}$ lie in the set G is at least $1 - (n')^{-c}$. Conditioned on this event, for each fixed query z, the probability over choice of randomness for $\hat{\mathtt{IC}}$ that $\hat{\mathtt{IC}}(z)$ does not output $L^{\mathsf{nice}}(z)$ is at most $\exp(-n)$. Hence, by another union-bound, with high probability all the queries of $\mathtt{Dec}$ are answered correctly, in which case $C'(x') = L^{\mathsf{nice}}(x')$. □

Now, assume that $C_{\bar{n}}^{L^{\mathsf{nice}}}$ is a NO instance, and let us call $z \in \{0,1\}^{n'}$ is bad if $\Pr\left[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathsf{nice}}}}(z) = \perp\right] \geq 1/6$. By the properties of the sampler, with high probability over the choice of $z_1, ..., z_D$, the fraction of bad z's in our sample is at least $1/2(n')^{-2c}$; and for any (fixed) bad z, the probability that $\Pr_{j \in [D]}\left[\mathtt{IC}^{C_{\bar{n}}^{L^{\mathsf{nice}}}}(z, r_j) = \perp\right] < .01$ is $\exp(-n)$. Hence, $C_{\bar{n}}^{L^{\mathsf{nice}}}$ will be rejected with high probability. The bound on the algorithm's running time follows from standard quasilinear-time algorithms for the Circuit Eval problem (see, e.g., [38, Thm 3.1]) and since $\tilde{O}(S(4\bar{n})) < \mathrm{poly}(n) \cdot S(2\bar{n})$.

Step 4: Derandomizing C'. The non-deterministic machine guessed-and-verified a circuit $C^{L^{\mathrm{nice}}}_{\bar{n}} : \{0,1\}^{\bar{n}} \to \{0,1\}$ such that $\mathtt{Crpt}(C^{L^{\mathrm{nice}}}_{\bar{n}}) \leq (n')^{-2c}$, and transformed it (using the algorithm from Proposition 5.7.2) into a probabilistic circuit C'. The machine M then constructs the final circuit C, which gets input $x \in \{0,1\}^n$ and acts as follows:

(1) Computes the reduction from L to L^{nice} to obtain $x' \in \{0,1\}^{n'}$.
(2) Hard-wires x' into C' to obtain a description of a circuit $C'_{x'} : \{0,1\}^{O(n)} \to \{0,1\}$ such that $C'_{x'}(r) = C'(x', r)$.
(3) Runs the hypothesized $\mathsf{CAPP}[v^{k'} \cdot S(v)]$ algorithm on C'_x and outputs its decision.

Note that C'_x is a circuit with $v = O(n)$ input bits and of size $\mathrm{poly}(n) \cdot S(4\bar{n}) = v^{O(1)} \cdot S(v)$, and therefore for an appropriate choice of constant k', the $\mathsf{CAPP}[v^{k'} \cdot S(v)]$ algorithm distinguishes between the case that C' accepts x' with high probability and the case that C' rejects x' with high probability. Thus, for every $x \in \{0,1\}^n$ it holds that $C(x) = L(x)$. Finally, both the size of the circuit C and the running time of our non-deterministic machine are bounded by $\tilde{O}\big(T\left((n^{O(1)} \cdot S(O(n)))\right)\big)$. ■

5.2 Proof of Theorems 1.4, 1.5, and 1.6

We now prove the main theorems from Section 1.3. We will first prove Theorem 1.4, which refers to the "low-end" parameter setting: Subexponential-time derandomization of $pr\mathcal{BPP}$ and lower bounds for polynomial-sized circuits against $\mathcal{EXP}$.

Theorem 5.8 (Theorem 1.4, restated). *Assume that there exists $\delta > 0$ such that $\mathcal{DTIME}[2^n]$ cannot be decided by $\mathcal{NTIME}[2^{n^\delta}]$-uniform circuits of an arbitrarily large polynomial size, even infinitely-often. Then, denoting $pr\mathcal{SUBEXP} = \cap_{\epsilon>0} pr\mathcal{DTIME}[2^{n^\epsilon}]$, we have that*

$$\cup_c \mathsf{pCAPP}[v^c, 4 \cdot \log(v)] \in \mathtt{i.o.}pr\mathcal{SUBEXP} \iff \mathcal{EXP} \not\subset \mathcal{P}/\mathrm{poly} .$$

Proof. Let us first prove the first statement. The "$\Longleftarrow$" direction follows from [3], relying on the fact that $\cup_c \mathsf{pCAPP}[v^c, 4 \cdot \log(v)] \in pr\mathcal{BPP}$. For the "$\Longrightarrow$" direction, assume that for every $c \in \mathbb{N}$ and every $\epsilon > 0$ it holds that $\mathsf{pCAPP}[v^c, 4 \cdot \log(v)] \in \mathtt{i.o.}pr\mathcal{DTIME}[2^{n^\epsilon}]$. Assuming towards a contradiction that $\mathcal{EXP} \subset \mathcal{P}/\mathrm{poly}$, we have that $\mathcal{DTIME}[2^n] \subset \mathcal{SIZE}[n^c]$ for some $c \in \mathbb{N}$. We use Item (1) of Proposition 5.6 with parameters $S(n) = n^c$ and $T(n) = 2^{n^\epsilon}$, where $\epsilon > 0$ is sufficiently small. We deduce that $\mathcal{DTIME}[2^n]$ can be decided infinitely-often by $\mathcal{NTIME}[T']$-uniform circuits of size n^c, where

$$T'(n) \leq T(\tilde{O}(S(\tilde{O}(S(n)))))^{O(1)} < T(n^{O_c(1)})^{O(1)} = 2^{n^{\epsilon \cdot O_c(1)}} ,$$

which contradicts our hypothesis if ϵ is sufficiently small. ■

We now prove Theorem 5.9, which refers to a "high-end" parameter setting (i.e., faster derandomization and lower bounds for larger circuits). We will in fact show that, conditioned on the hypothesis that $\mathcal{E}$ cannot be decided by $\mathcal{NTIME}[2^{\Omega(n)}]$-uniform circuits, even a weaker derandomization hypothesis is already equivalent to circuit lower bounds. For example, instead of assuming that $pr\mathcal{BPP} = pr\mathcal{P}$, we will only need to assume that CAPP for v-bit circuits of size $2^{\Omega(v)}$ can be solved deterministically in time $2^{\alpha \cdot v}$, for some small constant $\alpha > 0$. [41]

Theorem 5.9 (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 even infinitely-often. Then:*

[41] This is reminiscent of the recent results of Murray and Williams [43], who showed that solving CAPP for v-bit circuits of size $2^{\Omega(v)}$ in time $2^{.99 \cdot v}$ suffices to deduce circuit lower bounds. Note that the foregoing CAPP problem can be solved in *deterministic* polynomial time, since the input length is $2^{\Omega(v)}$ (i.e., this CAPP problem lies in $pr\mathcal{BPTIME}[\tilde{O}(n)] \cap pr\mathcal{P}$).

(1) *There exists a universal constant $c > 1$ such that*

$$\exists \epsilon > 0 : \mathrm{CAPP}[2^{\epsilon \cdot v}] \in pr\mathcal{DTIME}[n^{(\delta/c)/\epsilon)}] \iff \exists \epsilon > 0 : \mathcal{E} \not\subset \mathrm{i.o.}\mathcal{SIZE}[2^{\epsilon \cdot n}] \ .$$

(2) *For every fixed constant $c > 1$ it holds that*

$$\exists \alpha > 1 : \mathrm{CAPP}[2^{v^{1/c}}] \in pr\mathcal{DTIME}[2^{\alpha \cdot (\log n)^c}] \iff \exists \epsilon > 0 : \mathcal{E} \not\subset \mathrm{i.o.}\mathcal{SIZE}[2^{\epsilon \cdot n^{1/c}}] \ .$$

Proof. We first prove Item (1). The "$\Longleftarrow$" direction follows from [34] (or, alternatively, from the more general Corollary 3.3). Specifically, the hypothesized circuit lower bound implies that $pr\mathcal{BPP} = pr\mathcal{P}$, and in particular that $\mathrm{CAPP} \in pr\mathcal{DTIME}[n^{c'}]$ for some $c' \in \mathbb{N}$. The conclusion then holds for $\epsilon < \frac{\delta}{c \cdot c'}$. For the "$\Longrightarrow$" direction, let $k, k' \in \mathbb{N}$ be as in Proposition 5.7, and let $c = 2k$. Assume that for some $\epsilon > 0$ it holds that $\mathrm{CAPP}[S'] \in pr\mathcal{DTIME}[T]$, where $T(n) = n^{(\delta/c)/\epsilon)}$, and $S(n) = 2^{\epsilon \cdot n}/n^{k'}$, and $S'(v) = v^{k'} \cdot S(v) = 2^{\epsilon \cdot v}$. Assuming towards a contradiction that $\mathcal{E} \subset \mathrm{i.o.}\mathcal{SIZE}[S]$, Proposition 5.7 implies that $\mathcal{DTIME}[2^n]$ can be decided infinitely-often by $\mathcal{NTIME}[T']$-uniform circuits, where $T'(n) = \tilde{O}\left(T(n^k \cdot S(k \cdot n))\right) < 2^{\delta \cdot n}$; this is a contradiction.

The proof of Item (2) is similar. The "$\Longleftarrow$" follows from Corollary 3.3, instantiated with $S(n) = 2^{\epsilon \cdot n^{1/c}}$, to deduce that $\mathrm{CAPP} \in pr\mathcal{DTIME}[T]$ for $T(n) = 2^{\Delta \cdot S^{-1}(n^{\Delta})} = 2^{(\Delta/\epsilon)^c \cdot (\log n)^c}$. For the "$\Longrightarrow$" direction, let $\epsilon < (\delta/k\alpha)^{1/c}$ be sufficiently small, let $S(n) = 2^{\epsilon \cdot n^{1/c}}/n^{k'}$, let $S'(v) = v^{k'} \cdot S(v) = 2^{v^{1/c}}$, and let $T(n) = 2^{\alpha \cdot (\log n)^c}$. We use Proposition 5.7 as above, and rely on the fact that $T'(n) = \tilde{O}\left(T(n^k \cdot S(k \cdot n))\right) < 2^{\delta \cdot n}$. ■

Next, we prove Theorem 1.6, which asserts that if non-deterministic derandomization implies lower bounds against $\mathcal{EXP}$, then $\mathcal{EXP}$ does not have $\mathcal{NP}$-uniform circuits. We will actually prove a stronger result: First, we will use a weaker hypothesis than in Theorem 1.6, namely that $pr\mathcal{BPP} \subseteq pr\mathcal{NP}$ implies circuit lower bounds against $\mathcal{EXP}$; and secondly, we will deduce the stronger conclusion that $\mathcal{EXP} \not\subseteq (\mathcal{NP} \cap \mathcal{P}/\mathrm{poly})$. (This conclusion is stronger because the class of problems decidable by $\mathcal{NP}$-uniform circuits is a subclass of $\mathcal{NP} \cap \mathcal{P}/\mathrm{poly}$.)

Theorem 5.10 (Theorem 1.6, restated). *Assume that there exists $\delta > 0$ such that $\mathcal{E}$ does not have $\mathcal{NTIME}[2^{n^{\delta}}]$-uniform circuits of an arbitrarily large polynomial size. Then,*

$$pr\mathcal{BPP} \subset pr\mathcal{NSUBEXP} \Longrightarrow \mathcal{EXP} \not\subset \mathcal{P}/\mathrm{poly} \ , \tag{5.1}$$

where $pr\mathcal{NSUBEXP} = \cap_{\epsilon > 0} pr\mathcal{NTIME}[2^{n^{\epsilon}}]$. In the other direction, if Eq. (5.1) holds,[42] then $\mathcal{EXP} \not\subseteq (\mathcal{NP} \cap \mathcal{P}/\mathrm{poly})$, and in particular $\mathcal{EXP}$ does not have $\mathcal{NP}$-uniform circuits.

Proof. The proof of the first statement is similar to the proof of Theorem 5.8. We assume that $\mathcal{EXP} \subset \mathcal{P}/\mathrm{poly}$, and use Item (2) of Proposition 5.6 with parameters $S(n) = n^c$ and $T(n) = 2^{n^{\epsilon}}$, where $\epsilon > 0$ is sufficiently small; we deduce that any $L \in \mathcal{E}$ can be decided on all input lengths by $\mathcal{NTIME}[T']$-uniform circuits of size n^c, where $T'(n) < 2^{O(n^{3\epsilon \cdot c})} < 2^{n^{\delta}}$, which is a contradiction (the last inequality relied on $\epsilon > 0$ being sufficiently small).

To prove the "in the other direction" statement, first recall that $pr\mathcal{EXP} \subseteq pr(\mathcal{NP} \cap \mathcal{P}/\mathrm{poly}) \iff \mathcal{EXP} \subseteq (\mathcal{NP} \cap \mathcal{P}/\mathrm{poly})$, because every exponential-time machine that solves a promise problem also induces a language.[43] Now, assume towards a contradiction that $pr\mathcal{EXP} \subseteq pr(\mathcal{NP} \cap \mathcal{P}/\mathrm{poly})$. Since $pr\mathcal{BPP} \subseteq pr\mathcal{EXP}$, we have that $pr\mathcal{BPP} \subseteq pr(\mathcal{NP} \cap \mathcal{P}/\mathrm{poly})$. By the hypothesized conditional statement, it follows that $\mathcal{EXP} \not\subset \mathcal{P}/\mathrm{poly}$, a contradiction. ■

[42] In fact, for this statement it suffices to assume that $pr\mathcal{BPP} \subseteq pr\mathcal{NP} \Longrightarrow \mathcal{EXP} \not\subset \mathcal{P}/\mathrm{poly}$. However, since we will show a result with tighter relations between the parameters below (see Theorem 5.11), in the current statement we ignore this issue for simplicity.

[43] In more detail, the "$\Longrightarrow$" direction is trivial, so we prove the "$\Longleftarrow$" direction. For every $\Pi \in pr\mathcal{EXP}$, let M be an exponential-time machine that solves Π, and let L_M be the set of inputs that M accepts. Since $L_M \in \mathcal{EXP}$, there exists an $\mathcal{NP}$-machine that decides L_M and a polynomial-sized circuit family that decides L_M, and the foregoing machine and circuit family also solve Π.

As mentioned in the introduction, by optimizing the parameters we can show 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 this result, we define the following class of growth functions, which lie "in between" quasipolynomial functions and sub-exponential functions. 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 $\mathsf{e}^{(2,c)}(n) \in 2^{2^{\mathrm{loglog}(n)^c}}$. 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 sub-exponential function. Then, we have that:

Theorem 5.11 (Theorem 1.6, a tighter version). *For any constant $k \in \mathbb{N}$ we have that:*

$$\exists \delta > 0 : \mathcal{DTIME}[2^n] \text{ does not have } \mathcal{NTIME}[T]\text{-uniform circuits, for } T = 2^{\mathsf{e}^{(k,\delta)}} \tag{5.2}$$

$$\Downarrow$$

$$pr\mathcal{BPP} \subseteq \cap_{\epsilon>0} pr\mathcal{NTIME}[2^{\mathsf{e}^{(k,\epsilon)}}] \implies \mathcal{DTIME}[2^n] \not\subset \cup_{c_0 \in \mathbb{N}} \mathcal{SIZE}[\mathsf{e}^{(k,c_0)}] \tag{5.3}$$

$$\Downarrow$$

$$\forall c_0 \in \mathbb{N}, \mathcal{DTIME}[2^n] \not\subset (\mathcal{NTIME}[T] \cap \mathcal{SIZE}[T]), \text{ for } T(n) = \mathsf{e}^{(k,c_0)} \tag{5.4}$$

that is, statement (5.2) *implies statement* (5.3)*, which in turn implies statement* (5.4)*.*

We stress that the gap between the values of T in statements (5.2) and (5.4) is substantial, but nevertheless much smaller than an exponential gap. This is since in statement (5.2) the hypothesis is for T that is exponential in $\mathsf{e}^{(k,\delta)}$ where $\delta > 0$ is an *arbitrarily small constant*, whereas in statement (5.4) 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 5.11. To see that statement (5.2) implies statement (5.3), 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 $\mathsf{e}^{(k,c)}(\mathsf{e}^{(k,c')}(n)) = \mathsf{e}^{(k,cc')}(n)$. Now, assuming that $pr\mathcal{BPP} \subseteq \cap_\epsilon pr\mathcal{NTIME}[2^{\mathsf{e}^{(k,\epsilon)}}]$ and that $\mathcal{DTIME}[2^n] \subset \cup_{c_0} \mathcal{SIZE}[\mathsf{e}^{(k,c_0)}]$, we will show that Eq. (5.2) does not hold. To do so we use Item (2) of Proposition 5.6 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 for some $b \in \mathbb{N}$ it holds that $T'(n) < T(S(S(n)^b)^b)^b < T(\mathsf{e}^{(k,2b^2 \cdot c_0)}(n))^b = 2^{\mathsf{e}^{(k,2\epsilon b^3 \cdot c_0)}(n)}$.

To see that statement (5.3) implies statement (5.4), assume towards a contradiction that for some $c_0 \in \mathbb{N}$ it holds that $pr\mathcal{DTIME}[2^n] \subseteq pr(\mathcal{NTIME}[T] \cap \mathcal{SIZE}[T])$, where $T(n) = \mathsf{e}^{(k,c_0)}(n)$. Hence, CAPP $\in \mathcal{DTIME}[\tilde{O}(2^n)] \subseteq pr(\mathcal{NTIME}[T(\tilde{O}(n))] \cap \mathcal{SIZE}[T(\tilde{O}(n))])$, and it follows that

$$\begin{aligned} pr\mathcal{BPP} &\subseteq \cup_{c \in \mathbb{N}} pr\mathcal{NTIME}[T(n^c)] \\ &\subseteq \cup_{c \in \mathbb{N}} pr\mathcal{NTIME}\left[\mathsf{e}^{(k,c)}\right] \\ &\subseteq \cap_{\epsilon>0} pr\mathcal{NTIME}\left[2^{\mathsf{e}^{(k,\epsilon)}}\right] . \end{aligned}$$

By our hypothesis (i.e., by Eq. (5.3)) it follows that $\mathcal{DTIME}[2^n] \not\subset \cup_{c_0 \in \mathbb{N}} \mathcal{SIZE}\left[\mathsf{e}^{(k,c_0)}\right]$, which is a contradiction. Finally, to deduce the statement (i.e. bridge the gap between $pr\mathcal{DTIME}[2^n]$ and $\mathcal{DTIME}[2^n]$), we use the same argument as in Footnote 43. ∎

6 NOT-RETH AND CIRCUIT LOWER BOUNDS FROM RANDOMIZED ALGORITHMS

In this section we prove Theorem 1.7. We first show the desired $\mathcal{BPE}$ lower bounds follow from a weak learning algorithm for general circuits of quasi-linear size, and then show such an algorithm follows from the $2^{n/\mathrm{polylog}(n)}$-time randomized CircuitSAT algorithm for roughly quadratic-size circuits.

We first generalize the definition of weak learning algorithms, so that the algorithm is now required to learn any possible small oracle circuits.

Definition 6.1 (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.*[44]

Next, we need the following standard diagonalization argument.

Proposition 6.2 (diagonalization against circuits in Σ_4). *Let $\delta = 2^{-n/\mathrm{polylog}(n)}$, k_{ckt} be a constant, and f^{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^{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 $C(y) \neq D(y)$. 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^{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 6.3. *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 $C(x) \neq L(x)$, 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 weak learning algorithms imply non-trivial circuit lower bounds for $\mathcal{BPE}$.

Theorem 6.4 (weak 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 $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$.*

[44] In Section 3.1 we defined $\mathcal{SIZE}$ as referring to languages, whereas here we apply this notation to a fixed n-bit function. The meaning of $\mathcal{SIZE}(O)$ here is the size of the smallest circuit computing O.

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

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

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

Let L^{diag} be the language guaranteed by Proposition 6.2 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^{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 $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathsf{ckt}}}]$. ∎

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

We now prove Theorem 1.7, which asserts that randomized algorithms that solve CircuitSAT in time $2^{n/\mathrm{polylog}(n)}$ imply circuit lower bounds against $\mathcal{BPE}$. As explained in Section 2.3, we do so by showing that the foregoing algorithms for CircuitSAT imply the weak learner for quasi-linear size circuits, which enables us to apply Theorem 6.4.

Reminder of Theorem 1.7. *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 6.4 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 $x_1, x_2, \ldots, x_t$ 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 $\le s(n)$ 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. ∎

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

One shortcoming of Theorem 1.7 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.[45] To address this shortcoming, we now prove a different version of Theorem 1.7, which asserts that 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 6.5 (randomized Σ_2-SAT algorithms imply circuit lower bounds against $\mathcal{BPE}$). *For any constant $k_{\mathrm{ckt}} > 0$, there is another constant $k_{\mathrm{sat}} = k_{\mathrm{sat}}(k_{\mathrm{ckt}})$ such that if Σ_2-SAT with n variables and n clauses can be decided in randomized $2^{n/(\log n)^{k_{\mathrm{sat}}}}$ time, then $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathrm{ckt}}}]$.*

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 $s(n) = n \cdot (\log n)^{k_{\mathrm{ckt}}}$.

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 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 $\mathtt{TQBF}^{\mathtt{loc}}{}_{m+1}$. 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 6.2, and d be a constant such that L^{diag} is $n \cdot (\log n)^d$-time reducible to $\mathtt{TQBF}^{\mathtt{loc}}$. By setting k_{sat} large enough, we can compute $\mathtt{TQBF}^{\mathtt{loc}}{}_{n\cdot(\log n)^d}$ (and therefore also L^{diag}_n) in $2^{o(n)}$ time, Therefore, it follows that $\mathcal{BPTIME}[2^n] \not\subset \mathcal{SIZE}[n \cdot (\log n)^{k_{\mathrm{ckt}}}]$. ∎

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 6.6 (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 $n^{\mathrm{polyloglog}(n)}$.*

[45] 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.

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 6.5 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 $\mathsf{TQBF} \notin \mathcal{BPTIME}[2^{n/(\log n)^{k'+1}}]$. In this case, Item (2) follows from Theorem 4.14. ∎

We note that to prove Corollary 6.6 we do not have to use Theorem 6.5. An alternative proof relies on the fact that the Σ_4 formula from the proof of Proposition 6.2 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 6.2 in polynomial time, and decide its satisfiability in probabilistic time $2^{o(n)}$, which implies that $L^{\mathrm{diag}} \in \mathcal{BPE}$; Item (1) of Corollary 6.6 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 4.14 to deduce Item (2) of Corollary 6.6.

ACKNOWLEDGMENTS

We are grateful to Igor Oliveira for pointing us to the results in [46, Sec. 5], which serve as a basis for the proof of Theorem 1.7. We 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 1.7 can be proved, and for feedback on the manuscript. Finally, we thank an anonymous reviewer for pointing out a bug in the initial proof of Theorem 1.6, which we fixed.

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, NSF CCF-2127597, a Google Faculty Research Award, an IBM Fellowship, and a Miller Research Fellowship. Part of this work was done while Lijie Chen was at MIT. Ron Rothblum is supported in part by a Milgrom family grant, by the Israeli Science Foundation (Grant No. 1262/18), the Technion Hiroshi Fujiwara cyber center and by the European Union (ERC, FASTPROOF, 101041208). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. 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), and by the National Science Foundation under grant number CCF-1445755 and under grant number CCF-1900460. Part of this work was done while Roei Tell was at the Weizmann Institute of Science and at MIT. Eylon Yogev is supported by an Alon Young Faculty Fellowship, by the Israel Science Foundation (Grant No. 2893/22), and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

REFERENCES

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

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

[3] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. 1993. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity* 3, 4 (1993), 307–318.

[4] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. 2013. On the concrete efficiency of probabilistically-checkable proofs. In *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC)*. 585–594.

[5] Charles H. Bennett and John Gill. 1981. Relative to a random oracle A, $\mathbf{P}^A \neq \mathbf{NP}^A \neq \text{co} - \mathbf{NP}^A$ with probability 1. *SIAM Journal of Computing* 10, 1 (1981), 96–113.
[6] Jin-Yi Cai, Ajay Nerurkar, and D. Sivakumar. 1999. Hardness and hierarchy theorems for probabilistic quasi-polynomial time. In *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC))*. 726–735.
[7] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. 2016. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proc. 7th Conference on Innovations in Theoretical Computer Science (ITCS)*. 261–270.
[8] Marco L. Carmosino, Russell Impagliazzo, and Manuel Sabin. 2018. Fine-grained derandomization: from problem-centric to resource-centric complexity. In *Proc. 45th International Colloquium on Automata, Languages and Programming (ICALP)*. Art. No. 27, 16.
[9] Lijie Chen. 2019. Non-deterministic Quasi-Polynomial Time is Average-case Hard for ACC Circuits. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.
[10] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. 2019. Relations and Equivalences Between Circuit Lower Bounds and Karp-Lipton Theorems. In *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 30:1–30:21.
[11] Lijie Chen and Hanlin Ren. 2020. Strong Average-Case Circuit Lower Bounds from Non-trivial Derandomization. In *Proc. 52th Annual ACM Symposium on Theory of Computing (STOC)*.
[12] Lijie Chen, Ron D. Rothblum, and Roei Tell. 2022. Unstructured Hardness to Average-Case Randomness. In *Proc. 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.
[13] Lijie Chen and Roei Tell. 2021. Hardness vs randomness, revised: uniform, non-black-box, and instance-wise. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 125–136.
[14] Lijie Chen and R. Ryan Williams. 2019. Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity. In *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 19:1–19:43.
[15] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. 2014. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms* 10, 4 (2014), Art. 21, 32.
[16] Lance Fortnow and Adam R. Klivans. 2009. Efficient learning algorithms yield circuit lower bounds. *Journal of Computer and System Sciences* 75, 1 (2009), 27–36.
[17] Lance Fortnow, Rahul Santhanam, and Ryan Williams. 2009. Fixed-polynomial size circuit bounds. In *Proc. 24th Annual IEEE Conference on Computational Complexity (CCC)*. 19–26.
[18] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. 1989. On Completeness and Soundness in Interactive Proof Systems. *Advances in Computing Research* 5 (1989), 429–442.
[19] Oded Goldreich. 2008. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, New York, NY, USA.
[20] Oded Goldreich. 2011. In a World of P=BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*. 191–232.
[21] Oded Goldreich and Leonid A. Levin. 1989. A Hard-core Predicate for All One-way Functions. In *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*. 25–32.
[22] Oded Goldreich and Or Meir. 2015. Input-oblivious proof systems and a uniform complexity perspective on P/poly. *ACM Transactions on Computation Theory* 7, 4 (2015), Art. 16, 13.
[23] Oded Goldreich and Guy N. Rothblum. 2017. Worst-case to Average-case reductions for subclasses of P. *Electronic Colloquium on Computational Complexity: ECCC* 26 (2017), 130.
[24] Yuri Gurevich and Saharon Shelah. 1989. Nearly linear time. In *Logic at Botik, Symposium on Logical Foundations of Computer Science*. 108–118.
[25] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. 2009. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *Journal of the ACM* 56, 4 (2009), Art. 20, 34.
[26] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. 2003. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity* 12, 3-4 (2003), 85–130.
[27] Dan Gutfreund and Salil Vadhan. 2008. Limitations of hardness vs. randomness under uniform reductions. In *Proc. 12th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 469–482.
[28] Ryan C. Harkins and John M. Hitchcock. 2013. Exact learning algorithms, betting games, and circuit lower bounds. *ACM Transactions on Computation Theory* 5, 4 (2013), Art. 18, 11.
[29] Tzvika Hartman and Ran Raz. 2003. On the distribution of the number of roots of polynomials and explicit weak designs. *Random Structures & Algorithms* 23, 3 (2003), 235–263.
[30] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. 2002. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences* 65, 4 (2002), 672–694.
[31] Russell Impagliazzo and Ramamohan Paturi. 2001. On the complexity of k-SAT. *Journal of Computer and System Sciences* 62, 2 (2001), 367–375.

[32] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63, 4 (2001), 512–530.
[33] R. Impagliazzo and A. Wigderson. 1998. Randomness vs. Time: De-Randomization Under a Uniform Assumption. In *Proc. 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 734–.
[34] Russell Impagliazzo and Avi Wigderson. 1999. P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*. 220–229.
[35] Valentine Kabanets. 2001. Easiness assumptions and hardness tests: trading time for zero error. Vol. 63. 236–252.
[36] R. Kannan. 1982. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control* 55, 1-3 (1982), 40–56.
[37] Adam Klivans, Pravesh Kothari, and Igor Oliveira. 2013. Constructing Hard Functions Using Learning Algorithms. In *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 86–97.
[38] Richard J. Lipton and Ryan Williams. 2013. Amplifying circuit lower bounds against polynomial time, with applications. *Computational Complexity* 22, 2 (2013), 311–343.
[39] Yanyi Liu and Rafael Pass. 2022. Characterizing Derandomization Through Fine-Grained Hardness of Levin-Kolmogorov Complexity. In *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*.
[40] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2011. Lower bounds based on the exponential time hypothesis. *Bulletin of the European Association for Theoretical Computer Science (EATCS)* 105 (2011), 41–71.
[41] Chi-Jen Lu. 2001. Derandomizing Arthur-Merlin games under uniform assumptions. *Computational Complexity* 10, 3 (2001), 247–259.
[42] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. 1992. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery* 39, 4 (1992), 859–868.
[43] Cody Murray and Ryan Williams. 2018. 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)*.
[44] Noam Nisan and Avi Wigderson. 1994. Hardness vs. randomness. *Journal of Computer and System Sciences* 49, 2 (1994), 149–167.
[45] Igor C. Oliveira. 2013. Algorithms versus Circuit Lower Bounds. *Electronic Colloquium on Computational Complexity: ECCC* 20 (2013), 117.
[46] Igor C. Oliveira and Rahul Santhanam. 2017. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 79. Art. No. 18, 49.
[47] Nicholas Pippenger and Michael J. Fischer. 1979. Relations among complexity measures. *Journal of the ACM* 26, 2 (1979), 361–381.
[48] Rahul Santhanam. 2009. Circuit lower bounds for Merlin-Arthur classes. *SIAM Journal of Computing* 39, 3 (2009), 1038–1061.
[49] Rahul Santhanam and Ryan Williams. 2013. On medium-uniformity and circuit lower bounds. In *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 15–23.
[50] Ronen Shaltiel and Christopher Umans. 2005. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM* 52, 2 (2005), 172–216.
[51] Ronen Shaltiel and Christopher Umans. 2007. Low-end uniform hardness vs. randomness tradeoffs for AM. In *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*. 430–439.
[52] Adi Shamir. 1992. IP = PSPACE. *Journal of the ACM* 39, 4 (1992), 869–877.
[53] Victor Shoup. 1990. New algorithms for finding irreducible polynomials over finite fields. *Math. Comp.* 54, 189 (1990), 435–447.
[54] Madhu Sudan, Luca Trevisan, and Salil Vadhan. 2001. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences* 62, 2 (2001), 236–266.
[55] Roei Tell. 2019. Proving that $prBPP = prP$ is as hard as proving that "almost NP" is not contained in P/poly. *Information Processing Letters* 152 (2019), 105841.
[56] Luca Trevisan and Salil P. Vadhan. 2007. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity* 16, 4 (2007), 331–364.
[57] Christopher Umans. 2003. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences* 67, 2 (2003), 419–440.
[58] Salil P. Vadhan. 2012. *Pseudorandomness.* Now Publishers.
[59] Ryan Williams. 2013. Improving Exhaustive Search Implies Superpolynomial Lower Bounds. *SIAM Journal of Computing* 42, 3 (2013), 1218–1244.
[60] Ryan Williams. 2014. Algorithms for circuits and circuits for algorithms: Connecting the tractable and intractable. In *Proc. International Congress of Mathematicians (ICM)*. 659–682.
[61] Richard Ryan Williams. 2016. 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. Art. No. 2, 17.
[62] Virginia V. Williams. 2015. 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. 17–29.
[63] Virginia Vassilevska Williams. 2018. On some fine-grained questions in algorithms and complexity. Accessed at https://people.csail.mit.edu/virgi/eccentri.pdf, October 17, 2019.

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

A ON IMPLICATIONS OF MAETH

Consider the hypothesis MAETH, which asserts that *co*-3SAT cannot be solved by Merlin-Arthur protocols running in time $2^{\epsilon \cdot n}$, for some $\epsilon > 0$. Recall that the "strong" version of this hypothesis is false (since Williams [61] showed that #CircuitSAT can be solved by a Merlin-Arthur protocol in time $\tilde{O}(2^{n/2})$), but there is currently no evidence against the "non-strong" version.

As mentioned in Section 1.3, 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) result relies on a standard Karp-Lipton-style argument, which originates in [3]. We note in advance that after the proof of this result 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 A.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 $\mathcal{DTIME}[2^n] \not\subseteq \mathcal{MATIME}[S']$, where $S'(n) = S(k \cdot n)^k$. Then, $L \notin \mathcal{SIZE}[S]$.*

Note that, using Corollary 3.3, under the hypothesis of Theorem A.1 we have that CAPP $\in$ i.o.$pr\mathcal{DTIME}[T]$, 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 $pr\mathcal{BPP} \subseteq$ i.o.$pr\mathcal{DTIME}[n^{O(\log\log(n))}]$.

Proof of Theorem A.1. Let L be the problem from Proposition 3.12. Assuming towards a contradiction that $L \in \mathcal{SIZE}[S]$, we show that $\mathcal{DTIME}[2^n] \subseteq \mathcal{MATIME}[S']$.

Let $L_0 \in \mathcal{DTIME}[2^n]$. 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 3.12 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 $\text{poly}(n) \cdot \text{poly}(S(n')) = S(O(n))^{O(1)}$. ∎

In the following result, instead of assuming strong (e.g., super-polynomial) lower bounds for $\mathcal{MATIME}$ against $\mathcal{E}$, we assume fixed polynomial lower bounds for $\mathcal{MATIME}$ against $\mathcal{P}$, and deduce both a sub-exponential derandomization of $\mathcal{BPP}$, and a polynomial-time derandomization of $\mathcal{BPP}$ with n^ϵ advice, for an arbitrarly small constant $\epsilon > 0$.[46]

Theorem A.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 $\mathcal{P} \not\subseteq$ i.o.$\mathcal{MATIME}[n^k]$. Then, for every $\epsilon > 0$ it holds that $pr\mathcal{BPP} \subseteq (pr\mathcal{P}/n^\epsilon \cap pr\mathcal{DTIME}[2^{n^\epsilon}])$.*

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

[46] Recall that, by Adleman's theorem [1, 5], we can derandomize $pr\mathcal{BPP}$ with $\text{poly}(n)$ bits of non-uniform advice (and even with $O(n)$ bits, using Theorem 3.5). However, an unconditional derandomization of $pr\mathcal{BPP}$ with $o(n)$ bits of non-uniform advice is not known.

speaking, the following claim, whose proof is a refinement of on an argument from [10], 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 A.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, $\mathcal{P} \subseteq \mathtt{i.o.}\mathcal{MATIME}[n^{O(k)}]$.*

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

For every $x \in \{0,1\}^n$, let $T_x : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}$ 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 $\Pi \in \{0,1\}^{\mathrm{poly}(n)}$ 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 $\mathcal{P} \not\subseteq \mathtt{i.o.}\mathcal{MATIME}[n^k]$, and taking the counter-positive of Claim A.2.1, we deduce that:

Corollary A.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 A.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 3.2 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 $L \in pr\mathcal{DTIME}[2^{n^{2\epsilon}}]$. 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 $\mathrm{poly}(n)$. 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 3.2 with seed length $n^\epsilon + O(\log(n))$ and for circuits of size n^c to "fool" the circuit R_x. ∎

B 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 [23]. 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 B.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 [54, Thm 29], instantiated with degree parameter $d = d(n)$ and agreement parameter $\rho = \rho(n)$. The algorithm runs in time $\text{poly}(\ell(n), d, \log(q(n)), 1/\rho) = \text{poly}(n, 1/\rho)$ 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 [54] for $\text{poly}(1/\rho)$ times, we obtain a list of $t = \text{poly}(1/\rho)$ circuits $C_1, ..., C_t$ such that with probability at least $1 - \text{poly}(\rho)$ 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 $t' = \text{poly}(1/\rho)$ *deterministic* circuits $D_1, ..., D_{t'}$ such that with probability $1 - \text{poly}(\rho)$ 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 $1 - \text{poly}(\rho)$, 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., [2, Thm 19.19]), while resolving its oracle queries using the circuit D_i. The standard algorithm runs in time $\text{poly}(q, \ell, d)$, 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 $d/q < \delta = o(1)$. ∎

C AN $\mathcal{E}$-COMPLETE PROBLEM WITH USEFUL PROPERTIES

In this appendix we prove Proposition 3.12, which asserts the existence of an $\mathcal{E}$-complete problem (under linear-time reductions) that is randomly self-reducible, has an instance checker with linear-length queries, and such that both the random self-reducibility algorithm and the instance checker use a linear number of random bits.

Proposition C.1 (an $\mathcal{E}$-complete problem that is random self-reducible and has a good instance checker). *For every $\eta > 0$ there exists $L^{\text{nice}} \in \mathcal{DTIME}[\tilde{O}(2^n)]$ such that:*

1. *Any $L \in \mathcal{DTIME}[2^n]$ reduces to L^{nice} in polynomial time with a multiplicative blow-up of at most $1 + \eta$ in the input length. Specifically, for every n there exists $n' \leq (1 + \eta) \cdot n$ such that any n-bit input for L is mapped to an n'-bit input for L^{nice}.*
2. *The problem L^{nice} is randomly self-reducible by an algorithm $\texttt{Dec}$ that on inputs of length n uses $n + \text{polylog}(n)$ random bits.*
3. *There is an instance checker $\texttt{IC}$ for L^{nice} that on inputs of length n uses $n + O(\log(n))$ random bits and makes $O(1)$ queries of length $\ell(n)$, where $\ell(n) < (2 + \eta) \cdot n$.*

Proof. For a sufficiently small $\delta \leq \eta/7$, let $L^{\mathcal{E}} = \{(\langle M \rangle, x) : M \text{ accepts } x \text{ in } 2^{|x|} \text{ steps}\}$. Let $f_{L^{\mathcal{E}}} : \{0,1\}^* \to \{0,1\}^*$ be the low-degree extension of $L^{\mathcal{E}}$ such that inputs of length n_0 for $L^{\mathcal{E}}$ are mapped to inputs in $\mathbb{F}^m$, where $m = \delta \cdot \frac{n_0}{\lceil \log(n_0) \rceil}$ and $|\mathbb{F}| = 2^{(1/\delta+1)\cdot\lceil\log(n_0)\rceil}$, for a polynomial of individual degree $d = \lceil (n_0)^{1/\delta} \rceil$. Note that $(d+1)^m \geq 2^{n_0}$ (i.e., there is a unique extension of $L^{\mathcal{E}}$ with these parameters), and that $|\mathbb{F}| > m \cdot d$ (i.e., the polynomial is indeed of low degree). Finally, let L^{nice} be the set of pairs $(z, i) \in \{0,1\}^{m \cdot \log(|\mathbb{F}|)} \times \{0,1\}^{\lceil \text{loglog}(|\mathbb{F}|) \rceil}$, such that $f_{L^{\mathcal{E}}}(z)_i = 1$ (i.e., the i^{th} bit in the binary representation of $f_{L^{\mathcal{E}}}(z) \in \mathbb{F}$ equals one).

Note that $L^{\mathcal{E}}$ is reducible in polynomial time to $f_{L^{\mathcal{E}}}$, which is in turn reducible in polynomial time to L^{nice}; and that inputs of length $n_0 \in \mathbb{N}$ for $L^{\mathcal{E}}$ are mapped to inputs of length $n = m \cdot \log(|\mathbb{F}|) + \lceil \text{loglog}(|\mathbb{F}|) \rceil + 1 < (1 + 2\delta) \cdot n_0$ for L^{nice}. Thus any $L \in \mathcal{DTIME}[2^n]$ is reducible in polynomial time to L^{nice} with a multiplicative overhead of at most $1 + 3\delta$ in the input length. Also note that $L^{\text{nice}} \in \mathcal{DTIME}[\tilde{O}(2^n)]$, since the polynomial $f_{L^{\mathcal{E}}}$ can be evaluated in such time.

Let us now prove that L^{nice} is randomly self-reducible with at most $(1+\delta)\cdot n$ random bits. Let $\mathtt{Dec}_0$ be the standard random self-reducibility algorithm for $f_{L^{\mathcal{E}}}$, which uses less than n random bits.[47] Given input $(z,i) \in \{0,1\}^{m\cdot\lceil\log(|\mathbb{F}|)\rceil+\lceil\log\log(|\mathbb{F}|)\rceil}$ and oracle access to some $L' \subseteq \{0,1\}^n$, we simulate $\mathtt{Dec}_0$ at input z and with oracle access to a function induced by L' (as detailed below), and then output the i^{th} bit of its answer. Specifically, we initially choose a random permutation π of $\{0,1\}^{\log\log(|\mathbb{F}|)}$, using $\mathrm{polylog}(n) < \delta\cdot n$ random coins, and whenever $\mathtt{Dec}_0$ makes a query $q_1 \in \mathbb{F}^m$, we query L' at all inputs $\{(q_1,q_2)\}_{q_2\in\{0,1\}^{\lceil\log\log(|\mathbb{F}|)\rceil}}$, ordered according to π, and answer $\mathtt{Dec}_0$ accordingly. Note that each of our queries is uniformly distributed: This is since for every query (q_1,q_2) we have that q_1 is uniform (because $\mathtt{Dec}_0$'s queries are uniform) and that q_2 is uniform and independent from q_1 (because we chose a random π). Also note that if $L'(q_1,q_2) = L^{\mathsf{nice}}(q_1,q_2)$ for every query (q_1,q_2), then each query q_1 of $\mathtt{Dec}_0$ is answered by $f_{L^{\mathcal{E}}}(q_1)$, in which case we output $f_{L^{\mathcal{E}}}(z)_i = L^{\mathsf{nice}}(z,i)$.

Finally, to see that L^{nice} has an instance checker that uses $n+O(\log(n))$ random bits and issues $O(1)$ queries of length $(2+7\delta)\cdot n$, fix a PCP system for $\mathcal{DTIME}[T]$, where $T(n)=\tilde{O}(2^n)$, with the following specifications: The verifier V runs in polynomial time, uses $n+O(\log(n))$ bits of randomness, issues $O(1)$ queries, and has perfect completeness and soundness error $1/6$; and there is an algorithm P that gets an input $x\in\{0,1\}^n$ and outputs a proof for x in this PCP system (or $\perp$, if $x\notin L$) in deterministic time $\tilde{O}(2^n)$ (for a suitable PCP system, see [4, Thm 1]). We will instantiate this PCP system for the set $L_1^{\mathsf{nice}} = \left\{(z,i,b) : L^{\mathsf{nice}}(z,i)=b\right\}$, which is in $\mathcal{DTIME}[\tilde{O}(2^n)]$.

The instance checker $\mathtt{IC}$ for L^{nice} gets input $(z,i)\in\{0,1\}^n$ and simulates the verifier V for L_1^{nice} on inputs $(z,i,0)$ and $(z,i,1)$. Whenever $V(z,i,b)$ queries its proof at location $j\in[\tilde{O}(2^n)]$, the instance checker $\mathtt{IC}$ uses its oracle to try and decide the problem Π at input (z,i,b,j), where $\Pi = \left\{((z,i,b),j) : P(z,i,b)_j = 1\right\}$. Specifically, since $\Pi\in\mathcal{DTIME}[\tilde{O}(2^{n/2})]\subseteq\mathcal{DTIME}[\tilde{O}(2^n)]$ it holds that Π reduces to L^{nice} in polynomial time and with multiplicative blow-up of $1+3\delta$ in the input length; hence, $\mathtt{IC}$ reduces $((z,i,b),j)$ to an input for L^{nice} of length $\ell(n)\le(1+3\delta)\cdot(2n+1)<(2+7\delta)\cdot n$ and uses its oracle to try and obtain $\Pi((z,i,b),j)$. For $\sigma\in\{0,1\}$, the instance checker $\mathtt{IC}$ outputs σ if and only if $V(z,i,\sigma)=1$ and $V(z,i,1-\sigma)=0$, and otherwise outputs $\perp$. Note that $\mathtt{IC}^{L^{\mathsf{nice}}}(z,i) = L^{\mathsf{nice}}(z,i)$, with probability one; and that $\mathtt{IC}$ errs when given oracle $L'\neq L^{\mathsf{nice}}$ (i.e., $\mathtt{IC}^{L'}(z,i) = 1-L^{\mathsf{nice}}(z,i)$) only when V accepts $(z,i,1-L^{\mathsf{nice}}(z,i))\notin L_1^{\mathsf{nice}}$, which happens with probability at most $1/6$ for any L'. ∎

[47] Recall that $\mathtt{Dec}_0$ chooses a random vector $\vec{u}\in\mathbb{F}^m$, which requires $m\cdot\log(|\mathbb{F}|) < n$ random bits, and queries its oracle on a set of points on the line corresponding to $\vec{u}$; see, e.g., [19, Sec. 7.2.1.1].