Electronic Colloquium on Computational Complexity, Report No. 10 (2020)

Strong Average-Case Circuit Lower Bounds from Non-trivial Derandomization

Lijie Chen*
Massachusetts Institute of Technology

Hanlin Ren†
IIIS, Tsinghua University

February 12, 2020

Abstract

We prove that for all constants a, $\mathsf{NQP} = \mathsf{NTIME}[n^{\mathrm{polylog}(n)}]$ cannot be $(1/2 + 2^{-\log^a n})$-approximated by $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits (ACC^0 circuits with a bottom layer of THR gates). Previously, it was even open whether E^{NP} can be $(1/2 + 1/\sqrt{n})$-approximated by $\mathsf{AC}^0[\oplus]$ circuits. As a straightforward application, we obtain an infinitely often $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable pseudorandom generator for poly-size ACC^0 circuits with seed length $2^{\log^\varepsilon n}$, for all $\varepsilon > 0$.

More generally, we establish a connection showing that, for a typical circuit class $\mathscr{C}$, non-trivial nondeterministic algorithms estimating the acceptance probability of a given S-size $\mathscr{C}$ circuit with an additive error $1/S$ (we call it a CAPP algorithm) imply strong $(1/2 + 1/n^{\omega(1)})$ average-case lower bounds for nondeterministic time classes against $\mathscr{C}$ circuits. Note that the existence of such (deterministic) algorithms is much weaker than the widely believed conjecture $\mathsf{PromiseBPP} = \mathsf{PromiseP}$.

We also apply our results to several sub-classes of TC^0 circuits. First, we show that for all k, NP cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathsf{Sum} \circ \mathsf{THR}$ circuits (exact $\mathbb{R}$-linear combination of threshold gates), improving the corresponding worst-case result in [Williams, CCC 2018]. Second, we establish strong average-case lower bounds and build $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRGs for $\mathsf{Sum} \circ \mathsf{PTF}$ circuits, for various regimes of degrees. Third, we show that non-trivial CAPP algorithms for $\mathsf{MAJ} \circ \mathsf{MAJ}$ indeed already imply worst-case lower bounds for TC^0_3 ($\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$). Since exponential lower bounds for $\mathsf{MAJ} \circ \mathsf{MAJ}$ are already known, this suggests TC^0_3 lower bounds are probably within reach.

Our new results build on a line of recent works, including [Murray and Williams, STOC 2018], [Chen and Williams, CCC 2019], and [Chen, FOCS 2019]. In particular, it strengthens the corresponding $(1/2 + 1/\mathrm{polylog}(n))$-inapproximability average-case lower bounds in [Chen, FOCS 2019].

The two important technical ingredients are techniques from Cryptography in NC^0 [Applebaum et al., SICOMP 2006], and Probabilistic Checkable Proofs of Proximity with NC^1-computable proofs.

*lijieche@mit.edu. Supported by NSF CCF-1741615 and a Google Faculty Research Award.
†rhl16@mails.tsinghua.edu.cn

ISSN 1433-8092

1 Introduction

1.1 Background and Motivation

A holy grail of theoretical computer science is to prove *unconditional* circuit lower bounds for explicit functions (such as $\mathsf{NP} \not\subset \mathsf{P}_{/\mathrm{poly}}$). To approach this notoriously hard central open problem, the first step is to understand the power of various *constant depth* circuit classes. Back in the 1980s, there was a lot of significant progress in proving lower bounds for constant depth circuits. A line of works [Ajt83, FSS84, Yao85, Hås89] established exponential lower bounds for AC^0 (constant depth circuits consisting of AND/OR gates of unbounded fan-in), and [Raz87, Smo87] proved exponential lower bounds for $\mathsf{AC}^0[p]$ (AC^0 circuits extended with MOD_p gates) when p is a prime.

However, the progress had stopped there—the power of $\mathsf{AC}^0[m]$ for a composite m had been elusive, despite that it had been conjectured that they cannot even compute the majority function. In fact, it had been a notorious long-standing open question in computational complexity whether NEXP (nondeterministic exponential time) has polynomial-size ACC^0 circuits[1], until a seminal work by Williams [Wil14b] a few years ago, which proved NEXP does not have polynomial-size ACC^0 circuits, via a new *algorithmic* approach to circuit lower bounds [Wil13].

Not only being an exciting new development after a long gap, the new circuit lower bound is also remarkable as it surpasses all previous known barriers for proving circuit lower bounds: relativization [BGS75], algebrization [AW09], and natural proofs [RR97][2]. Moreover, the underlying method (the algorithmic method) puts many important classical complexity gems together, ranging from nondeterministic time hierarchy theorem [SFM78, Žák83], $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92], hardness vs randomness [NW94], to PCP Theorem [ALM+98, AS98].

Recent development of the algorithmic approach to circuit lower bounds. Recently, Murray and Williams [MW18] significantly advanced the algorithmic approach by proving that strong enough circuit-analysis (Gap-UNSAT)[3] algorithms can also imply circuit lower bounds for NQP (nondeterministic quasi-polynomial time) or NP, instead of the previous gigantic class NEXP. Building on the new connection and the corresponding algorithms for $\mathsf{ACC}^0 \circ \mathsf{THR}$ [Wil14a], they showed that $\mathsf{NQP} \not\subset \mathsf{ACC}^0 \circ \mathsf{THR}$.

Building on [MW18], [Che19] recently generalized the connection to the *average-case*, by showing that strong enough circuit-analysis algorithms also imply $(1/2 + o(1))$-inapproximability average-case lower bounds for NQP or NP. In particular, it was shown that NQP cannot be $(1/2 + 1/\mathrm{polylog}(n))$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$. This is very interesting for two reasons: first, average-case lower bounds tend to have other applications such as constructing unconditional PRGs; second, the proof techniques do not apply the easy-witness lemma of [Wil14b, MW18], and follows a more direct approach.

Still, the $(1/2 + 1/\mathrm{polylog}(n))$-inapproximability result is not enough to get us a non-trivial (say, with $n^{o(1)}$ seed length) PRG construction for ACC^0, which requires at least a $(1/2 + 1/n^{\omega(1)})$-inapproximability bound.

[1] It had been stressed several times as one of the most *embarrassing* open questions in complexity theory, see [AB09]. ACC^0 denotes the union of $\mathsf{AC}^0[m]$ for all constant m.

[2] We remark that there is no consensus on whether the natural proof barrier applies to ACC^0: *i.e.*, there is no widely accepted construction of PRFs in ACC^0. A candidate construction [BIP+18] is proposed recently, which still needs to be tested. But we can say that *if* there is a natural proof barrier for ACC^0, then this lower bound has surpassed it. (We also remark that there is a recent proposal on getting ACC^0 circuit lower bounds via torus polynomials [BHLR19].)

[3] The Gap-UNSAT problem asks one to distinguish between an unsatisfiable formula and a formula accepting a random input with probability $> 1/2$.

The $1/2+1/\sqrt{n}$ Razborov-Smolensky barrier. Indeed, proving a non-trivial $(1/2+n^{-\omega(1)})$-inapproximability result is even open for $\mathsf{AC}^0[\oplus]$ circuits (AC^0 circuits extended with parity gates). Using the renowned polynomial approximation method, [Raz87, Smo87, Smo93] showed that the majority function cannot be $(1/2+n^{1/2-\varepsilon})$-approximated by $\mathsf{AC}^0[\oplus]$. However, it is even open that whether E^{NP} can be $(1/2+1/\sqrt{n})$-approximated by $(\log n)$-degree $\mathbb{F}_2$-polynomials. Improving the $(1/2+1/\sqrt{n})$-bound (and constructing the corresponding PRGs) is recognized as a significant open question in circuit complexity [Vio09a, Vad12, FSUV13, CHLT19].

1.2 Our Results

In this paper, we significantly improve the circuit-analysis-algorithms-to-average-case-lower-bounds connection in [Che19]. We first define the circuit-analysis task of our interest.

- CAPP[4] **for $\mathscr{C}$ circuits with inverse-circuit-size error**: Given a $\mathscr{C}$ circuit C of size S on n input bits, estimate $\Pr_{x\in\{0,1\}^n}[C(x)=1]$ within an additive error $1/S$.

For simplicity, throughout this paper, we will just refer to the above problem as CAPP. We remark that under the widely believed assumption $\mathsf{PromiseBPP}=\mathsf{PromiseP}$, this problem has a $\mathrm{poly}(S)$ time algorithm even for $\mathscr{C}=\mathsf{P}_{/\mathrm{poly}}$. In the following, we show that indeed a non-trivial improvement on the brute-force $2^n\cdot\mathrm{poly}(S)$-time algorithm already implies strong average-case lower bounds for $\mathscr{C}$.

From Non-trivial CAPP Algorithms to Strong Average-case Circuit Lower Bounds

Theorem 1.1. *Let $\mathscr{C}$ be a typical circuit class[5] such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. The following hold.*

(NP Average-Case Lower Bound) Suppose there is a constant $\varepsilon>0$ such that the CAPP problem of $\mathsf{AND}_4\circ\mathscr{C}$ circuits of size $2^{\varepsilon n}$ can be solved in $2^{n-\varepsilon n}$ time. Then for every constant $k\geq 1$, NP cannot be $(1/2+n^{-k})$-approximated by $\mathscr{C}$ circuits of n^k size.

(NQP Average-Case Lower Bound) Suppose there is a constant $\varepsilon>0$ such that the CAPP problem of $\mathsf{AND}_4\circ\mathscr{C}$ circuits of size 2^{n^ε} can be solved in 2^{n-n^ε} time. Then for every constant $k\geq 1$, NQP cannot be $(1/2+2^{-\log^k n})$-approximated by $\mathscr{C}$ circuits of $2^{\log^k n}$ size.

(NEXP Average-Case Lower Bound) Suppose the CAPP problem of $\mathsf{AND}_4\circ\mathscr{C}$ circuits of size $\mathrm{poly}(n)$ can be solved in $2^n/n^{\omega(1)}$ time. Then NE cannot be $(1/2+1/\mathrm{poly}(n))$-approximated by $\mathscr{C}$ circuits of $\mathrm{poly}(n)$ size.

By the standard Discriminator Lemma [HMP+93], we immediately obtain worst-case lower bounds for $\mathsf{MAJ}\circ\mathscr{C}$ circuits as well.

Corollary 1.2. *Under the algorithmic assumptions of Theorem 1.1, we obtain worst-case lower bounds for $\mathsf{MAJ}\circ\mathscr{C}$ circuits in the corresponding cases: (1) NP not in n^k-size $\mathsf{MAJ}\circ\mathscr{C}$ for all k; (2) NQP not in $2^{\log^k n}$-size $\mathsf{MAJ}\circ\mathscr{C}$ for all k; (3) NE not in $\mathrm{poly}(n)$-size $\mathsf{MAJ}\circ\mathscr{C}$.*

[4] The acronym CAPP denotes the Circuit Acceptance Probability Problem.

[5] A circuit class $\mathscr{C}$ is *typical* if it is closed under both negation and projection.

Remark 1.3. We remark that the conclusions of Theorem 1.1 still hold if the corresponding CAPP algorithms are *non-deterministic*. That is, on any computational branch, it either outputs a correct estimation[6] or rejects, and it does not reject all branches.

Remark 1.4. Theorem 1.1 assumes $\mathscr{C}$ is a sub-class of NC^1 (*e.g.*, $\mathsf{THR} \circ \mathsf{THR}$, TC^0, or ACC^0). On the other hand, if $\mathscr{C}$ is stronger than NC^1 (*e.g.*, NC^2, $\mathsf{P}_{/\mathsf{poly}}$), [Che19, Theorem 1.3] already showed that[7] even CAPP with constant error suffices to prove the stated average-case lower bounds in Theorem 1.1. Although we still left open the possible case that $\mathscr{C}$ is uncomparable to NC^1, our theorem together with [Che19] cover nearly all interesting circuit classes.

Comparison with [Che19]. Our Theorem 1.1 improves on the corresponding connection in [Che19] in two ways: (1) we get a much better inapproximability bound, which is crucial for our construction of nondeterministic PRGs; (2) we only need CAPP algorithms for $\mathsf{AND}_4 \circ \mathscr{C}$, while [Che19] requires algorithms for $\mathsf{AC}^0 \circ \mathscr{C}$. On the other hand, our requirement on the CAPP algorithms is stronger (additive error $1/S$) than that of [Che19] (constant additive error).

More on our definition on CAPP. We remark that our definition of CAPP is a bit non-standard, comparing to the usual definition with a constant error. Nonetheless, such a CAPP algorithm is *much weaker* than a full-power #SAT algorithm, and (as discussed before) is widely believed to exist even for $\mathsf{P}_{/\mathrm{poly}}$ circuits.

Strong Average-Case Lower Bounds for $\mathsf{ACC}^0 \circ \mathsf{THR}$

Applying the non-trivial #SAT algorithms for $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits in [Wil14a], it follows that NQP cannot be even weakly approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, and it is (worst-case) hard for $\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

Theorem 1.5. *For every constant $k \geq 1$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$. Consequently, NQP cannot be computed by $\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$ (in the worst-case), for all $k \geq 1$.*

The same holds for $(\mathsf{N} \cap \mathsf{coN})\mathsf{QP}_{/1}$ in place of NQP.

Very recently, building on [CW19] and the FGLSS reduction [FGL+91], Vyas and Williams [VW20] proved that NQP cannot be computed by $\mathsf{EMAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$[8] circuits of size $2^{\log^k n}$ for all $k \geq 1$. Our result improves on theirs as $\mathsf{EMAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits is a subclass of $\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits (see, e.g. [HP10]).

Nondeterministic PRGs for ACC^0 with Sub-Polynomial Seed Length

As an important application of the above strong average-case lower bound, we also obtain the first PRG with $n^{o(1)}$ seed length for ACC^0 circuits (previous, this was open even for $\mathsf{AC}^0[\oplus]$ circuits), albeit it is nondeterministic and infinitely often.

[6] It is allowed that on different branches it outputs different estimations as long as they are all within an additive error of $1/S$.

[7] [Che19, Theorem 1.3] only states the result with inapproximability $1/2 + n^{-c}$ for a constant c, but it is easy to see that its proof can be generalized to the inapproximability corresponding to Theorem 1.1.

[8] EMAJ is the "exact majority" function which outputs 1 on an n-bit input if and only if the number of ones in the input equals $\lceil \frac{n}{2} \rceil$.

Theorem 1.6. *For every constant $\varepsilon > 0$, there is an infinitely often, $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG fooling polynomial size ACC^0 circuits with seed length $2^{(\log n)^{\varepsilon}}$.*[9]

Remark 1.7. We can indeed optimize the seed length to be the *inverse of any sub-fourth-exponential function*. See Section 7.2 for details.

Previously, the best PRG for ACC^0 is from [COS18], which is $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable and has seed length $n - n^{1-\beta}$ for any constant $\beta > 0$. Our construction significantly improves on that.

Lower Bounds and PRGs for $\mathsf{Sum} \circ \mathscr{C}$ Circuits

For a circuit class $\mathscr{C}$, a $\mathsf{Sum} \circ \mathscr{C}$ circuit is an $\mathbb{R}$-linear combination $C(x) := \sum_{i=1}^{t} \alpha_i C_i(x)$, such that each $\alpha_i \in \mathbb{R}$, each C_i is a $\mathscr{C}$ circuit on n input bits, and $C(x) \in \{0,1\}$ for all $x \in \{0,1\}^n$. We denote t as the *sparsity* of the circuit, and we define the size of C as the total size of all $\mathscr{C}$ sub-circuits C_i's.

We first show that if we have the corresponding non-trivial #SAT algorithms instead of the non-trivial CAPP algorithms, we would have average-case lower bounds for $\mathsf{Sum} \circ \mathscr{C}$ circuits. To avoid repetition, in the following we only state the version for NQP.

Corollary 1.8. *Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. Suppose there is a constant $\varepsilon > 0$ such that the #SAT problem of $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $2^{n^{\varepsilon}}$ can be solved in $2^{n-n^{\varepsilon}}$ time. Then for every constant $k \geq 1$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{Sum} \circ \mathscr{C}$ circuits of $2^{\log^k n}$ size.*

This immediately implies a strong average-case lower bound for $\mathsf{Sum} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$.

Corollary 1.9. *For every constant $k \geq 1$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{Sum} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$. Consequently, NQP cannot be computed by $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$ (in the worst-case), for all $k \geq 1$.*

The same holds for $(\mathsf{N} \cap \mathsf{coN})\mathsf{QP}_{/1}$ in place of NQP.

Now we discuss some applications of our new techniques to some sub-classes of TC^0 circuits.

We begin with some notation. Recall that a degree-d PTF gate is a function defined by $\mathsf{sign}(p(x))$, where p is a degree-d polynomial on x over $\mathbb{R}$, and $\mathsf{sign}(z)$ outputs 1 if $z \geq 0$ and 0 otherwise. Clearly, a THR gate is simply a degree-1 PTF gate.

[Wil18] proved that NP cannot be computed by n^k-size $\mathsf{Sum} \circ \mathsf{THR}$ circuits for all $k > 0$. With our improved connection, we apply the #SAT algorithm for $\mathsf{AND}_4 \circ \mathsf{THR}$ of [Wil18] to improve it to a corresponding average-case lower bound.

Theorem 1.10. *For all constants k, NP cannot be $(1/2 + 1/n^k)$-approximated by n^k-size $\mathsf{Sum} \circ \mathsf{THR}$ circuits. Consequently, NP cannot be computed by n^k-size $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{THR}$ circuits for all constants k.*[10]

We remark that $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{THR}$ is a sub-class of $\mathsf{THR} \circ \mathsf{THR}$ with no previous known lower bounds. So Theorem 1.10 can be viewed as progress toward resolving the notorious open question of proving super-polynomial $\mathsf{THR} \circ \mathsf{THR}$ lower bounds.

Applying the non-trivial *zero-error* #SAT algorithm for PTF in [BKK+19], we also obtain NQP (NE) average-case lower bounds for $\mathsf{Sum} \circ \mathsf{PTF}_d$ circuits.

[9] That is, this PRG G is computable by a nondeterministic machine M with one bit of advice such that for a seed $s \in \{0,1\}^{2^{(\log n)^{\varepsilon}}}$, $M(s)$ either outputs $G(s)$ or rejects on any computational branch, and it outputs $G(s)$ on some computational branches. See Definition 2.7 for a formal definition.

[10] This average-case lower bound can also be extended to against $\mathsf{Sum} \circ \mathsf{ReLU}$ circuits, similar to the exact $\mathsf{Sum} \circ \mathsf{ReLU}$ lower bounds in [Wil18].

Theorem 1.11. *The following hold.*

- *For every constants $d, k \geq 1$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{Sum} \circ \mathsf{PTF}_d$ circuits of sparsity $2^{\log^k n}$. Consequently, NQP does not have $2^{\log^k n}$-size $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{PTF}_d$ circuits.*
- *Let $d(n) = 0.49\frac{\log n}{\log\log n}$, then NE cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by $\mathsf{Sum} \circ \mathsf{PTF}_{d(n)}$ circuits of sparsity $\mathrm{poly}(n)$. Consequently, $\mathsf{NE} \not\subset \mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{PTF}_{d(n)}$.*

From the above theorem, we can also obtain non-trivial nondeterministic PRGs for $\mathsf{Sum} \circ \mathsf{PTF}$ circuits.

Theorem 1.12. *For every constants $d, k \geq 1$ and $\varepsilon > 0$, there is an $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable i.o. PRG with seed length $O(2^{\log^\varepsilon n})$ that $(1/n^k)$-fools $\mathsf{Sum} \circ \mathsf{PTF}_d$ circuits of sparsity n^k.*[11]

Previously, the best (constant-error) PRG for degree-d PTF has seed length $O(\log n \cdot 2^{O(d)})$ [MZ13]. Our construction has a worse seed-length, is nondeterministic and infinitely often, but works for the larger class $\mathsf{Sum} \circ \mathsf{PTF}$.

Towards TC^0_3 Lower Bounds

In [CW19], it is shown that non-trivial CAPP algorithms for $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits with inverse-polynomial additive error would already imply $\mathsf{THR} \circ \mathsf{THR}$ circuit lower bounds. We significantly improve that connection by showing it would indeed imply TC^0_3 lower bounds!

Theorem 1.13. *If there is a $2^n/n^{\omega(1)}$ time CAPP algorithm for $\mathrm{poly}(n)$-size $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits. Then $\mathsf{NEXP} \not\subset \mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$.*

We remark that $\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ is actually equivalent to $\mathsf{MAJ} \circ \mathsf{THR} \circ \mathsf{THR}$ (since $\mathsf{MAJ} \circ \mathsf{MAJ} = \mathsf{MAJ} \circ \mathsf{THR}$ [GHR92]). Since exponential-size (worst-case) lower bounds against $\mathsf{MAJ} \circ \mathsf{MAJ}$ are already known. If only we can "mine" a non-trivial CAPP algorithm (which is widely believed to exist) for $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits from these lower bounds, we would have worst-case lower bounds against TC^0_3.

1.3 Intuition

In the following, we sketch the intuitions for our new average-case lower bounds.

In this section, we will aim for a simpler version that NQP cannot be $(1/2 + n^{-k})$-approximated by ACC^0 for a large constant k (say, $k = 10^3$) for simplicity. We believe this version already captures all important technical ideas of our new average-case circuit lower bounds.

1.3.1 Review of [Che19] and the Bottleneck

First, since our work crucially builds on [Che19] (which proved NQP cannot be $(1/2 + 1/\mathrm{polylog}(n))$-approximated by ACC^0), it would be very instructive to review the proof structure of [Che19], and understand what is the bottleneck of extending [Che19] to prove a $(1/2 + n^{-k})$-inapproximability bound.

[11] We did not attempt to optimize this seed length.

A high-level overview of [Che19]: three steps. Suppose we are proving NQP cannot be $(1-\delta)$-approximated by ACC^0 for now, where δ is a small constant. On a very high level, the proof of [Che19] involves the following three steps.[12]

Step I (Conditional collapse from NC^1 to ACC^0.)

Assuming NQP can be $(1-\delta)$-approximated by ACC^0, [Che19] shows that NC^1 collapses to ACC^0, using the existence of self-reducible NC^1-complete languages [Bab87, Kil88, Bar89].

Step II (An NE algorithm certifying low depth hardness.)

Next, making use of the non-trivial SAT algorithm for ACC^0 circuits [Wil14b], [Che19] shows that there is an NE algorithm $V(\cdot,\cdot)$ certifying n^ε-depth hardness. Formally, $V(x,y)$ takes inputs such that $|y| = 2^{|x|}$; for infinitely many n's, $V(1^n,\cdot)$ is satisfiable, and $V(1^n,y) = 1$ implies y, interpreted as a function $f_y : \{0,1\}^n \to \{0,1\}$, does not have n^ε-depth circuits.

Step III (Certifying low depth hardness implies average-case lower bounds for low depth circuits.)

Finally, [Che19] shows that the above algorithm V would be sufficient to imply that NQP cannot be $(1-\delta)$-approximated by NC^1 (and also ACC^0).

The bottleneck of the argument: Step I. Suppose we are going to prove NQP cannot be $(1/2+n^{-k})$-approximated by ACC^0, let us examine which one of the above three steps would break.

Clearly, Step II is unaffected (assuming Step I works). Another observation is that since NC^1 can compute majority[13], we can use the XOR Lemma [Yao82, IJKW10, GL89] to show that NQP cannot be $(1/2+n^{-k})$-approximated by NC^1 circuits.[14] Therefore, Step III still works if we want to prove the stronger $(1/2+n^{-k})$-inapproximability result.

However, Step I completely breaks. Assuming NQP can be $(1/2+n^{-k})$-approximated by ACC^0 circuits, it seems hopeless to show that NC^1 collapse to ACC^0 using some random self-reducible languages. This is because the given circuit only $(1/2+n^{-k})$-approximates the given random self-reducible language, and to the best of our knowledge, all known correcter for such languages in this error regime requires computing at least some variants of the majority function, while ACC^0 is conjectured not to be able to compute majority [Smo87]!

1.3.2 A Detour: Chen and Williams [CW19] and $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ Circuit Lower Bounds

So it seems unlikely that we can show a collapse theorem from NC^1 to ACC^0 under the assumption that NQP can be $(1/2+n^{-k})$-approximated by ACC^0. A natural idea to avoid this obstacle is to show NC^1 collapses to some other larger classes under the same assumption. Examining the proof idea of [Che19], it seems at least we can show NC^1 collapses to $\mathsf{MAJ} \circ \mathsf{ACC}^0$ under the assumption. However, the issue is that then we don't know how to implement Step II, as we don't have a non-trivial SAT (or even Gap-UNSAT) algorithm for $\mathsf{MAJ} \circ \mathsf{ACC}^0$ circuits.

So we indeed want a *collapse theorem which would collapse* NC^1 *to a circuit class* $\mathscr{C}$ *for which we at least know some non-trivial algorithms for, and of course* $\mathscr{C}$ *also has to contain* ACC^0. Perhaps the best

[12] Actually, in [Che19], Step III is much more complicated than the previous two steps, and Step II just follows from [Wil16]. In the presentation of [Che19], Step III is decomposed into several sub-steps [Che19, Section 6.2, 7-9]. We choose to give the overview in this way because we essentially make use of Step III as a black box, and our improvement is mostly focusing on the first two steps. In particular, our improved Step II is much more involved than that of [Che19], and crucially builds on [CW19].

[13] It is proved that black-box hardness amplification requires majority [SV10, GSV18].

[14] Precisely speaking, we have to start with our $(\mathsf{N}\cap\mathsf{coN})\mathsf{QP}_{/1}$ lower bounds for that purpose.

choice for us is the $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits which has recently been studied by [CW19]. So let us take a detour into this circuit class and the corresponding lower bounds in [CW19].

$\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ Circuits. Let $\mathscr{C}$ be a class of functions from $\{0,1\}^n \to \{0,1\}$ and $\delta \in [0, 0.5)$. We say $f : \{0,1\}^n \to \{0,1\}$ admits a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of sparsity S, if there are S functions $C_1, C_2, \ldots, C_S$ from $\mathscr{C}$, together with S coefficients $\alpha_1, \alpha_2, \ldots, \alpha_S$ in $\mathbb{R}$, such that for all $x \in \{0,1\}^n$,

$$\left| \sum_{i=1}^{S} \alpha_i \cdot C_i(x) - f(x) \right| \leq \delta.$$

Given a valid $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuit C, we say $C(x) = 1$ if the corresponding output value $|\sum_i \alpha_i C_i(x) - 1| \leq \delta$, and $C(x) = 0$ otherwise. [CW19] gives a 2^{n-n^ε}-time Gap-UNSAT (in fact, constant-error CAPP) algorithm for $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ of 2^{n^ε}-size when δ is small (the algorithm is indeed already implicit in [Wil18]). Building on this algorithm (and more importantly, PCP of proximity), [CW19] proves that $\mathsf{NQP} \not\subset \widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ for any constant $\delta \in [0, 1/2)$.

1.3.3 Key Technical Ingredient: A $\oplus\mathsf{L}$-complete Language CMD with a $\widetilde{\mathsf{Sum}}_\delta$ Error Correcter

So given the result of [CW19], the question becomes:

> **A New Collapse Theorem?**: Can we show a collapse from NC^1 to $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits, assuming NQP can be $(1/2 + n^{-k})$-approximated by ACC^0 circuits?

Our improvement of Step I answers the question affirmatively, by making use of a $\oplus\mathsf{L}$-complete[15] language CMD [IK02, AIK06, GGH+07] with very nice reducibility properties. We remark that the underlying techniques play a crucial part in the famous construction of NC^0-computable one-way functions (and low-stretch PRGs) [AIK06] (see also the book [App14]).

1. ($\oplus\mathsf{L}$-completeness under projections.) That is, for every language $L \in \oplus\mathsf{L}$, there is a polynomial-time computable projection P such that $L(x) = \mathsf{CMD}(P(x))$.

2. (Single-query error correctability with a randomized image DCMD.) For technical reasons, we also have to introduce another $\oplus\mathsf{L}$-complete language DCMD, which is a "randomized image" of CMD under projections (when randomness is fixed) [GGH+07, Claim 2.19].

   That is, given $n \in \mathbb{N}$, there is $m = \mathrm{poly}(n)$ and a randomized reduction $P(x, r)$ (r is the random bits) from CMD on input length n to DCMD on input length m, such that:

   (a) For all $x \in \{0,1\}^n$, $P(x, \mathcal{U}_\ell)$ distributes uniformly on $\{0,1\}^m$, where ℓ is the number of random bits involved, and $\mathcal{U}_\ell$ is the uniform distribution over $\{0,1\}^\ell$.

   (b) For all fixed random bits r, $P(x, r)$ is a projection of x.

   (c) For all $x \in \{0,1\}^n$, $\mathsf{CMD}_n(x) = \mathsf{DCMD}_m(P(x, r)) \oplus r_0$ for all r, where r_0 is the first bit of r.

[15] Roughly speaking, $\oplus\mathsf{L}$ consists of languages L such that there is an $O(\log n)$ space nondeterministic Turing machine M, such that on every input x, $x \in L$ if and only if there is an odd number of computational paths making M accept on input x.

An error corrector in $\widetilde{\mathsf{Sum}}_\delta \circ f$. The second property of CMD stated above is *amazing*. It enables us to do the desired error correction with $\widetilde{\mathsf{Sum}}_\delta \circ f$ circuits (a linear sum of f functions composed with projections). See Section 3 for the details. It then follows that if NQP can be $(1/2 + n^{-k})$-approximated by ACC^0 circuits, we would have the desired collapse from NC^1 to $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$.

1.3.4 A Simpler Proof for a Worst-Case Lower Bound Against $\mathsf{MAJ} \circ \mathsf{ACC}^0$

With the improved collapse result, we can already prove worst-case lower bounds against $\mathsf{MAJ} \circ \mathsf{ACC}^0$. For simplicity, here we only show the following weaker version.

Theorem 1.14 (Toy Example)**.** $\mathsf{NQP} \not\subset \mathsf{MAJ} \circ \mathsf{ACC}^0$.

Proof Sketch. There are two cases.

- First, we assume DCMD (which is in NQP) cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by ACC^0. This implies that $\mathsf{NQP} \not\subset \mathsf{MAJ} \circ \mathsf{ACC}^0$, via the standard Discriminator Lemma [HMP+93].
- Second, suppose DCMD can be $(1/2 + 1/n^k)$-approximated by n^k-size ACC^0 circuits for a constant k. This implies that NC^1 collapses to $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$.

  By [CW19], $\mathsf{NQP} \not\subset \widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$. This in turn implies that $\mathsf{NQP} \not\subset \mathsf{NC}^1$, and clearly also $\mathsf{NQP} \not\subset \mathsf{MAJ} \circ \mathsf{ACC}^0$. □

1.3.5 Toward Average-Case Hardness: The Updated Three Steps Plan

Now we switch to the new average-case circuit lower bounds. With the new conditional collapse theorem, the following are our updated three steps plan for the new average-case lower bounds.

Step I′ (Conditional collapse from NC^1 to $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$.)

Assuming NQP can be $(1/2 + n^{-k})$-approximated by ACC^0, we show that NC^1 collapses to $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$, utilizing the nice properties of the problems CMD and DCMD.

Step II′ (An NE algorithm certifying low depth hardness.)

Next, making use of the non-trivial constant error CAPP algorithm for $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits [Wil18, CW19], we show that there is an NE algorithm $V(\cdot,\cdot)$ certifying n^ε-depth hardness.

Step III′ (Certifying low depth hardness implies average-case lower bounds for low depth circuits.)

Finally, we show that the above algorithm V would be sufficient to imply that NQP cannot be $(1/2 + n^{-k})$-approximated by NC^1 (and also ACC^0).

As previously discussed, Step III′ can be achieved easily by combing [Che19] and the XOR Lemma [Yao82, IJKW10, GL89]. It remains to implement Step II′, which is the most technical part of this work.

1.3.6 Review of Step II: Certifying Hardness via PCP and Nondeterministic Time Hierarchy

To implement Step II′, the natural idea is to directly modify Step II ([Che19, Section 6.1]), and follow [Wil16]. Now we briefly review the details of Step II and explain why it seems hard to adapt it directly.

Setting up the verifier V_{cert}. Let L be a unary language in $\mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n]$ [Žák83]. Fix an efficient PCP verifier V_{PCP} for L (such as [BV14]). That is, for a function $\ell := \ell(n) = n + O(\log n)$, $V_{\mathsf{PCP}}(1^n)$ takes ℓ random bits as input, runs in $\mathrm{poly}(n)$ time, is given access to an oracle $O : \{0,1\}^\ell \to \{0,1\}$, and satisfies the following conditions:

1. (Completeness) if $1^n \in L$, then there exists an oracle O such that $V_{\mathsf{PCP}}(1^n)^O$ always accepts;
2. (Soundness) if $1^n \notin L$, then for all oracles O, the probability $V_{\mathsf{PCP}}(1^n)^O$ accepts is $\leq 1/n$.

Now, we define V_{cert} as follows: $V_{\mathsf{cert}}(1^n, y)$ treats y as the truth-table of an oracle $O_y : \{0,1\}^\ell \to \{0,1\}$, and verifies whether $V_{\mathsf{PCP}}(1^n)^{O_y}$ always accepts[16]. Clearly, V_{cert} runs in $\mathrm{poly}(n + |y|)$ time.

Since any depth-d circuit is equivalent to some $2^{O(d)}$-size $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuit (recall that now NC^1 collapses to $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$), to show that V_{cert} certifies n^{ε_1}-depth hardness, it suffices to show that V_{cert} certifies hardness for 2^{n^ε}-size $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits for $\varepsilon > \varepsilon_1$.

Let us suppose the opposite that V_{cert} does not certify hardness for 2^{n^ε}-size $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits. In particular, this means for all large enough n, if $V_{\mathsf{cert}}(1^n, \cdot)$ is satisfiable, then there is a 2^{n^ε}-size $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuit C such that $V_{\mathsf{cert}}(1^n, tt(C)) = 1$, where $tt(C)$ is the truth-table of C. Translating it to the setting of PCP, for large enough n, the following hold:

1. (Succinct Completeness) if $1^n \in L$, then there exists a 2^{n^ε}-size $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuit $C : \{0,1\}^\ell \to \{0,1\}$ such that $V_{\mathsf{PCP}}(1^n)^C$ always accepts;
2. (Soundness) if $1^n \notin L$, then for all oracles O, the probability $V_{\mathsf{PCP}}(1^n)^O$ accepts is $\leq 1/n$.

The issue with the direct approach. Given the above two conditions, the natural idea for putting L in $\mathsf{NTIME}[2^n/n]$ to obtain a contradiction would be to try the following nondeterministic algorithm for L: Given an input 1^n, we (non-deterministically) guess a 2^{n^ε}-size $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuit C[17], and try to estimate

$$p_{\mathsf{acc}}(V_{\mathsf{PCP}}(1^n)^C) = \Pr_{r \in \{0,1\}^\ell}[V_{\mathsf{PCP}}(1^n)^C(r)].$$

Let $D_C := V_{\mathsf{PCP}}(1^n)^C$. We would like to accept when $p_{\mathsf{acc}}(D_C) = 1$, and reject when $p_{\mathsf{acc}}(D_C) < 1/n$, so a constant additive error (say, $1/10$) approximation to $p_{\mathsf{acc}}(D_C)$ would suffice.

The issue here is that, D_C is *not a* $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ *circuit anymore*. So we don't know how to estimate $p_{\mathsf{acc}}(D_C)$ using the constant error CAPP algorithm for $\mathsf{Sum}_\delta \circ \mathsf{ACC}^0$ in [Wil18, CW19].

We remark that by [BV14], V_{PCP} can indeed be implemented by a 3-CNF, hence if C is only an ACC^0 circuit, $V_{\mathsf{PCP}}(1^n)^C$ is still an ACC^0 circuit. This is why this argument works in the original Step II, where we have a collapse from NC^1 to ACC^0 instead of $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$.

1.3.7 Getting Around of the Issue with PCP of Proximity

To avoid the aforementioned issue, we would like to adopt the PCP of Proximity framework introduced in [CW19], which also plays a crucial part in the P^{NP} construction of rigid matrices in [AC19]. For more intuition on this framework and how it compares to and improves on the earlier works [Wil13, Wil14b], one is referred to [CW19, Section 1.6].

[16] Strictly speaking, here $|y| = 2^\ell = 2^n \cdot \mathrm{poly}(n)$ which is slightly larger than 2^n, but this slight difference does not really matter in the proof.

[17] Note that here we are waiving the very important issue of *how to test whether the guessed* $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ *is valid*. We will discuss this issue at the end of the section.

For a SAT instance F, Y a subset of its variables, and $y \in \{0,1\}^{|Y|}$, we use $F_{Y=y}$ to denote the resulting instance obtained by assigning the Y variables in F to y.[18] We also use $\mathsf{OPT}(F)$ to denote the maximum fraction of clauses that can be satisfied by any assignment.

The following transformation is the key technical part of [CW19].[19]

Theorem 1.15 (Implicit in [CW19])**.** *Let* Enc *be the encoder of some constant-rate error correcting code. There is a polynomial-time transformation that, given a circuit D on n inputs of size $m \geq n$, outputs a* 2*-SAT instance F on variable set $Y \cup Z$, where $|Y| = O(n)$, $|Z| \leq \mathrm{poly}(m)$ and F has $\mathrm{poly}(m)$ clauses, such that for two constants $c_{\mathsf{PCPP}} > s_{\mathsf{PCPP}}$, the following hold for all $x \in \{0,1\}^n$.*

- *If $D(x) = 1$, then $\mathsf{OPT}(F_{Y=\mathsf{Enc}(x)}) \geq c_{\mathsf{PCPP}}$. Furthermore, there is a $\mathrm{poly}(m)$-time algorithm computing a corresponding $z_D(x)$ given x which satisfies at least a c_{PCPP} fraction of clauses.*
- *If $D(x) = 0$, then $\mathsf{OPT}(F_{Y=\mathsf{Enc}(x)}) \leq s_{\mathsf{PCPP}}$.*

The key idea of [CW19] is to apply the above transformation on the obtained circuit D_C, and *guess* the corresponding $\mathscr{C}$ circuits for each output bit of the function $z_{D_C}(x)$. In [CW19], the focus is to prove worst-case lower bounds like $\mathsf{NQP} \not\subset \mathscr{C}$ for a circuit class $\mathscr{C}$. Therefore, we can safely assume $\mathsf{P} \subseteq \mathscr{C}$ and there exist corresponding $\mathscr{C}$ circuits for each output bit of $z_{D_C}(x)$.

However, in our case, we only have the collapse from NC^1 to $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$. So we need the following adaption where the given circuit D is a *formula*, and the proof is also computable by a *formula*.

Theorem 1.16. *Let* Enc *be the encoder of some constant-rate error correcting code. There is a polynomial-time transformation that, given a* formula *D on n inputs of size $m \geq n$, outputs a* 2*-SAT instance F on variable set $Y \cup Z$, where $|Y| = O(n)$, $|Z| \leq \mathrm{poly}(m)$ and F has $\mathrm{poly}(m)$ clauses, such that for two constants $c_{\mathsf{PCPP}} > s_{\mathsf{PCPP}}$, the following hold for all $x \in \{0,1\}^n$.*

- *If $D(x) = 1$, then $\mathsf{OPT}(F_{Y=\mathsf{Enc}(x)}) \geq c_{\mathsf{PCPP}}$. Furthermore, there is a $\mathrm{poly}(m)$-size* formula *computing a corresponding $z_D(x)$ given x which satisfies at least a c_{PCPP} fraction of clauses.*
- *If $D(x) = 0$, then $\mathsf{OPT}(F_{Y=\mathsf{Enc}(x)}) \leq s_{\mathsf{PCPP}}$.*

The algorithm. Again, suppose for the sake of contradiction that V_{cert} does not certify n^ε-depth hardness. In particular, this means for all large enough n, it follows that if $V_{\mathsf{cert}}(1^n, \cdot)$ is satisfiable, then there is an n^ε-depth circuit C such that $V_{\mathsf{cert}}(1^n, tt(C)) = 1$. Translating it to the setting of PCP, the following hold for large enough n:

1. (Low Depth Completeness) if $1^n \in L$, then there exists an n^ε-depth circuit $C : \{0,1\}^\ell \to \{0,1\}$ such that $V_{\mathsf{PCP}}(1^n)^C$ always accepts;
2. (Soundness) if $1^n \notin L$, then for all oracles O, the probability that $V_{\mathsf{PCP}}(1^n)^O$ accepts is $\leq 1/n$.

Recall that we set $D_C := V_{\mathsf{PCP}}(1^n)^C$. Our goal now is still to accept when $p_{\mathsf{acc}}(D_C) = 1$, and reject when $p_{\mathsf{acc}}(D_C) \leq 1/n$.

[18] Here we don't remove the already satisfied clauses or the clauses which cannot be satisfied after the partial assignment.

[19] This formulation is due to [VW20].

By previous discussions, V_{PCP} can be taken as a 3-CNF, so D_C is indeed a circuit of depth $n^\varepsilon + O(\log n) = O(n^\varepsilon)$, and therefore it is also a formula of size $2^{O(n^\varepsilon)}$. Now we apply Theorem 1.16 to the formula D_C to obtain a 2-SAT instance F with $n_{\mathsf{clause}} = 2^{O(n^\varepsilon)}$ clauses on variable set $Y \cup Z$.

Now we guess $|Z|$ $\mathsf{Sum}_\delta \circ \mathsf{ACC}^0$ circuits $T_1, T_2, \ldots, T_{|Z|}$ and use $\widetilde{\pi}(x)$ to denote the concatenation of $T_1(x), T_2(x), \ldots, T_{|Z|}(x)$. Then we estimate the following quantity

$$p_{\mathsf{key}} := \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} \mathop{\mathbb{E}}_{i\in[n_{\mathsf{clause}}]} F_i(\mathsf{Enc}(x), \widetilde{\pi}(x)) = \mathop{\mathbb{E}}_{i\in[n_{\mathsf{clause}}]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} F_i(\mathsf{Enc}(x), \widetilde{\pi}(x)), \tag{1}$$

where F_i is the i-th clause in the 2-SAT instance F, so it only depends on two bits in $\mathsf{Enc}(x) \circ \widetilde{\pi}(x)$. By a simple manipulation, one can show that $F_i(\mathsf{Enc}(x), \widetilde{\pi}(x))$ also has a $\mathsf{Sum}_{O(\delta)} \circ \mathsf{ACC}^0$ circuit. Therefore, setting δ to be a small enough constant, we can apply the constant error CAPP algorithm from [Wil18, CW19] to estimate p_{key} in 2^{n-n^ε} time. Now we verify the correctness of the algorithm.

1. When $p_{\mathsf{acc}}(D_C) = 1$, on the correct guess that $\widetilde{\pi}(x) = z_{D_C}(x)$ for all x, by Item (1) of Theorem 1.16, it follows $p_{\mathsf{key}} \geq c_{\mathsf{PCPP}}$.
2. When $p_{\mathsf{acc}}(D_C) \leq 1/n$, on all possible guesses, by Item (2) of Theorem 1.16, we have $p_{\mathsf{key}} \leq s_{\mathsf{PCPP}} + 1/n$.

Therefore, to distinguish the above two cases, it suffices to estimate p_{key} within an additive error of $\frac{c_{\mathsf{PCPP}} - s_{\mathsf{PCPP}}}{10}$, and accept if our estimation is $\geq \frac{c_{\mathsf{PCPP}} + s_{\mathsf{PCPP}}}{2}$. Putting everything together, this puts $L \in \mathsf{NTIME}[2^n/n]$, contradiction.

Checking the guessed $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits. Finally, as we have remarked briefly before, we waived an important issue on checking whether the guessed $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits are *valid* (that is, whether the linear sum is close to either 0 or 1 on all inputs x). This is because in the algorithm described above, when $x \notin L$, it is still possible that we guess some *invalid* $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits $T_1, T_2, \ldots, T_{|Z|}$ and conclude that $p_{\mathsf{key}} > \frac{c_{\mathsf{PCPP}} + s_{\mathsf{PCPP}}}{2}$, as the constant error CAPP algorithm for $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ may behave arbitrarily on invalid $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuits.

More formally, given a presumed $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$ circuit C, let $f(x)$ be the corresponding $\sum_i \alpha_i C_i(x)$, and

$$\mathsf{bin}_f(x) := \begin{cases} 1 & f(x) > 1/2, \\ 0 & \text{otherwise.} \end{cases}$$

To test whether C is valid, we want to check whether $\|\mathsf{bin}_f - f\|_\infty \leq \delta$. Ideally, we want a test which accepts when $\|\mathsf{bin}_f - f\|_\infty \leq \delta$ and reject when (say) $\|\mathsf{bin}_f - f\|_\infty \geq 3\delta$. But this turns out to be too hard.

Luckily, a careful examination shows that we only have to reject when $\|\mathsf{bin}_f - f\|_2 \geq 3\delta$, and this can be solved by a careful polynomial manipulation as in [CW19]. See Section 5 for the details.

Contents

2 Preliminaries

We use $\mathsf{GF}(p^r)$ to denote the finite field of size p^r, where p is a prime and r is an integer. For $n \in \mathbb{N}$, we use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$. We use boldface letters to denote random variables, *e.g.* $\mathbf{x} \sim \mathcal{U}_n$ denotes an n-bit string sampled uniformly at random.

We say a function $f : \mathbb{N} \to \mathbb{N}$ is a **good resource function** if the following hold:

- f is nondecreasing, and
- there is a polynomial time algorithm that on input 1^n, outputs the value of $f(n)$.

For a non-decreasing function $f : \mathbb{N} \to \mathbb{N}$, define its inverse function f^{-1} as

$$f^{-1}(n) = \max\{m : f(m) \leq n\}.$$

It is easy to see that if f is a good resource function and $f(n) \geq n$, then f^{-1} is also a good resource function.

2.1 Complexity Classes and Basic Definitions

We assume knowledge of basic complexity theory (see [AB09, Gol08] for excellent references on this subject).

2.1.1 Basic Circuit Families

A *circuit family* is a collection of circuits $\{C_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$. A *circuit class* is a collection of circuit families. The *size* of a circuit is the number of *wires* in the circuit, and the size of a circuit family is a function of the input length that upper-bounds the size of circuits in the family. The *depth* of a circuit is the maximum number of wires on a path from an input gate to the output gate.

We will mainly consider classes in which the size of each circuit family is bounded by some polynomial; however, for a circuit class $\mathscr{C}$, we will sometimes also abuse notation by referring to $\mathscr{C}$ circuits with various other size or depth bounds.

By default, (general) circuits have fan-in two AND and OR gates and unary NOT gates. AC^0 is the class of circuit families of constant depth and polynomial size, with AND, OR, and NOT gates, where AND and OR gates have unbounded fan-in. For an integer m, the function $\mathsf{MOD}_m : \{0,1\}^* \to \{0,1\}$ is one if and only if the number of ones in the input is not divisible by m. The class $\mathsf{AC}^0[m]$ is the class of constant-depth circuit families consisting of polynomially-many unbounded fan-in AND, OR and MOD_m gates, along with unary NOT gates. We sometimes also use $\mathsf{AC}^0[\oplus]$ to denote $\mathsf{AC}^0[2]$. We use the notation AC^0_d and $\mathsf{AC}^0_d[m]$ when we explicitly specify the depth of the circuit as d. We denote $\mathsf{ACC}^0 = \bigcup_{m\geq 2} \mathsf{AC}^0[m]$.

The function majority, denoted as $\mathsf{MAJ} : \{0,1\}^* \to \{0,1\}$, is the function that outputs 1 if the number of ones in the input is no less than the number of zeros, and outputs 0 otherwise. TC^0 is the class of circuit families of constant depth and polynomial size, with unbounded fan-in MAJ gates. We also use TC^0_d to denote the sub-class of TC^0 of depth d. NC^k for a constant k is the class of $O(\log^k n)$-depth and polynomial-size circuit families consisting of fan-in two AND and OR gates and unary NOT gates.

A linear threshold function (LTF) is a function $\Phi : \{0,1\}^n \to \{0,1\}$ of the form $\Phi(x) = \mathsf{sign}(\langle x, w\rangle - \theta)$, where $w \in \mathbb{R}^n$, $\theta \in \mathbb{R}$ and $\langle x, w\rangle = \sum_{i\in[n]} x_i \cdot w_i$ is the standard inner product over $\mathbb{R}$. We use THR to denote the class of linear threshold functions. Similarly, a polynomial threshold function (PTF) is a function $\Phi : \{0,1\}^n \to \{0,1\}$ of the form $\Phi(x) = \mathsf{sign}(P(x))$, where

$P(x)$ is a polynomial over $\mathbb{R}$. The degree of the PTF is simply the degree of the polynomial, and we use PTF_d to denote the class of PTF of degree d. Clearly, THR is equivalent to PTF_1.

We say a circuit family $\{C_n\}_{n\in\mathbb{N}}$ is uniform, if there is a deterministic algorithm A, such that $A(1^n)$ runs in time polynomial of the size of C_n, and outputs C_n.[20]

For a circuit class $\mathscr{C}$, we say a circuit C is a $\mathscr{C}$ oracle circuit, if C is also allowed to use a special oracle gate (which can occur multiple times in the circuit, but with the same fan-in), in addition to the usual gates allowed by $\mathscr{C}$ circuits. We say an oracle circuit is *non-adaptive*, if on any path from an input gate to the output gate, there is at most one oracle gate.

We say a circuit class $\mathscr{C}$ is typical, if given the description of a circuit C of size s, for indices $i, j \leq n$ and a bit b, the following functions

$$\neg C, C(x_1, \ldots, x_{i-1}, x_j \oplus b, x_{i+1}, \ldots, x_n), C(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$$

all have $\mathscr{C}$ circuits of size s, and their corresponding circuit descriptions can be constructed in $\mathrm{poly}(s)$ time. That is, $\mathscr{C}$ is typical if it is closed under both *negation* and *projection*.

2.1.2 Notation

We say a circuit $C : \{0,1\}^n \to \{0,1\}$ γ-approximates a function $f : \{0,1\}^n \to \{0,1\}$, if $C(x) = f(x)$ for a γ fraction of inputs from $\{0,1\}^n$.

For a function $f : \{0,1\}^n \to \{0,1\}$, we define $\mathsf{SIZE}(f)$ (resp. $\mathsf{DEPTH}(f)$) to be the minimum size (resp. depth) of a circuit computing f exactly. Similarly, for an error parameter $\gamma > 1/2$, we define $\mathsf{heur}_\gamma\text{-}\mathsf{SIZE}(f)$ (resp. $\mathsf{heur}_\gamma\text{-}\mathsf{DEPTH}(f)$) to be the minimum size (resp. depth) of a circuit γ-approximating f.

We say a language L can be $\gamma(n)$-approximated by $\mathscr{C}$, if there is a circuit family $\{C_n\}_{n\in\mathbb{N}} \in \mathscr{C}$ such that C_n $\gamma(n)$-approximates L_n for all sufficiently large n. We also say a class of language $\mathscr{L}$ can be $\gamma(n)$-approximated by $\mathscr{C}$, if all languages $L \in \mathscr{L}$ can be $\gamma(n)$-approximated by $\mathscr{C}$.

2.1.3 Circuit Analysis Problems

We mainly consider two circuit-analysis problems.

- #SAT: Given a circuit C on n inputs, count the number of satisfying assignments of C.
- CAPP with inverse-circuit-size error: Given a circuit C on n inputs, estimate the probability that $C(x) = 1$ for a uniformly random $x \in \{0,1\}^n$, within error $\pm 1/|C|$. Here $|C|$ denotes the size of C.

In this paper, unless otherwise stated, the problem CAPP means CAPP with inverse-circuit-size error.

We say the #SAT or CAPP problem for $\mathscr{C}$ circuits of size $S(n)$ can be solved in *nondeterministic* $T(n)$ time, if the following hold. There is a nondeterministic Turing machine M such that, when given a size-$S(n)$ $\mathscr{C}$ circuit C over n Boolean variables as input,

- M accepts at least one of its nondeterministic branches.
- On every accepted nondeterministic branch, M outputs a *correct answer*. For the #SAT problem, the only correct answer is the number of satisfying assignments of C; for the CAPP

[20] That is, we use the P-uniformity by default.

problem, any rational number q such that

$$\left| \Pr_{\mathbf{x} \sim \{0,1\}^n} [C(\mathbf{x}) = 1] - q \right| < \frac{1}{|C|}$$

is a correct answer.

We remark that the CAPP problem is widely believed to be solvable in polynomial time even for polynomial-size general circuits ($\mathsf{P}_{/\mathrm{poly}}$).

2.2 $\mathsf{MA} \cap \mathsf{coMA}$ and $\mathsf{NP} \cap \mathsf{coNP}$ Algorithms

We introduce convenient definitions of $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T(n)]$ or $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T(n)]$ algorithms, which simplify the presentation.

Definition 2.1. Let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A language L is in $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T(n)]$, if there is a deterministic algorithm $A(x, y, z)$ (which is called the predicate) such that:

- A takes three inputs x, y, z such that $|x| = n$, $|y| = |z| = O(T(n))$ (y is the witness while z is the collection of random bits), runs in $O(T(n))$ time, and outputs an element from $\{0, 1, ?\}$.
- (Completeness) For every x, there exists a y such that

$$\Pr_{\mathbf{z} \sim \mathcal{U}_{O(T(n))}} [A(x, y, \mathbf{z}) = L(x)] \geq 2/3.$$

- (Soundness) For all x and y,

$$\Pr_{\mathbf{z} \sim \mathcal{U}_{O(T(n))}} [A(x, y, \mathbf{z}) = 1 - L(x)] \leq 1/3.$$

We say the predicate has *depth* d, if for every x, y, there is a depth-d circuit $C_{x,y}$ such that $C_{x,y}(z) = A(x, y, z)$.

Remark 2.2. $(\mathsf{MA} \cap \mathsf{coMA})$ languages with advice are defined similarly, with A being an algorithm with the corresponding advice.

Remark 2.3. The definition of predicate depth here is a little bit unusual: we allow Arthur to choose different circuits (that process the randomness z) on each input x and proof y. It makes essentially no difference in this paper if we change the definition to the actual depth of the predicate A, but the current definition of depth is more convenient to work with.

Definition 2.4. Let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A language L is in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T(n)]$, if there is an algorithm $A(x, y)$ (which is called the predicate) such that:

- A takes two inputs x, y such that $|x| = n$, $|y| = O(T(n))$ (y is the witness), runs in $O(T(n))$ time, and outputs an element from $\{0, 1, ?\}$.
- (Completeness) For every x, there exists a y such that

$$A(x, y) = L(x).$$

- (Soundness) For all x and y,
$$A(x,y) \neq 1 - L(x).$$

Remark 2.5. $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$ languages with advice are defined similarly, with A being an algorithm with the corresponding advice.

Note that by the above definition, the semantic of $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$ is different from $\mathsf{MA}_{/1}\cap\mathsf{coMA}_{/1}$. A language in $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$ has both an $\mathsf{MA}_{/1}$ algorithm and a $\mathsf{coMA}_{/1}$ algorithm, and *their advice bits are the same*. In contrast, a language in $\mathsf{MA}_{/1}\cap\mathsf{coMA}_{/1}$ can have an $\mathsf{MA}_{/1}$ algorithm and a $\mathsf{coMA}_{/1}$ algorithm with different advice sequences. Similar relationship holds for $(\mathsf{NP}\cap\mathsf{coNP})_{/1}$ and $\mathsf{NP}_{/1}\cap\mathsf{coNP}_{/1}$.

2.3 Pseudorandom Generators

We are going to deal with different types of Pseudorandom Generators (PRG) throughout the paper. In the following, we recall their definitions.

Definition 2.6. Let $S,\ell : \mathbb{N}\to\mathbb{N}$ and $\varepsilon : \mathbb{N}\to(0,1)$ be functions, where S,ℓ,ε denote size, seed length and error respectively. Let $\mathscr{C}$ be a circuit class. A function $G:\{0,1\}^*\to\{0,1\}^*$ is a *pseudorandom generator* (PRG) with *seed length* $\ell(n)$ that *ε-fools* $\mathscr{C}$ circuits of size S, if for every $\mathscr{C}$ circuit $C:\{0,1\}^n\to\{0,1\}$ of size $S(n)$,

$$\left|\Pr_{\mathbf{s}\sim\mathcal{U}_{\ell(n)}}[C(G(\mathbf{s}))=1]-\Pr_{\mathbf{r}\sim\mathcal{U}_n}[C(\mathbf{r})=1]\right|<\varepsilon(n). \tag{2}$$

If (2) holds for infinitely many lengths n, then we say G is an *infinitely often PRG* (i.o. PRG).

Definition 2.7. Let $G:\{0,1\}^*\to\{0,1\}$ be a PRG. We say G is E-*computable* if there is a deterministic Turing machine M that on input $x\in\{0,1\}^*$, computes $G(x)$ in $2^{O(|x|)}$ time. We say G is $(\mathsf{NE}\cap\mathsf{coNE})$-*computable* if there is a deterministic Turing machine $M(x,y)$ that on input $x\in\{0,1\}^n$ and $y\in\{0,1\}^{2^{O(n)}}$:

- M either rejects or outputs $G(x)$;
- there is at least one input y such that M does not reject;
- M runs in $2^{O(n)}$ time.

Furthermore, if M needs a bits of advice, then we say G is $\mathsf{E}_{/a}$-computable or $(\mathsf{NE}\cap\mathsf{coNE})_{/a}$-computable respectively.

We also need the concept of "nondeterministic pseudorandom generator" (NPRG).

Definition 2.8. Let $S,\ell:\mathbb{N}\to\mathbb{N}$ and $\varepsilon:\mathbb{N}\to(0,1)$ be functions, and $\mathscr{C}$ be a circuit class. Suppose there is a deterministic Turing machine $M(x,y)$ such that:

1. On input $x\in\{0,1\}^{\ell(n)}$ and $y\in\{0,1\}^{2^{O(\ell(n))}}$, $M(x,y)$ either rejects or outputs a string, and whether $M(x,y)$ rejects only depends on y (*i.e.* not x).

2. If $M(x,y)$ does not reject, then there is a function $G_y : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ such that for every $\mathscr{C}$ circuit C of size S,

$$\left| \Pr_{\mathbf{x}\sim\mathcal{U}_{\ell(n)}}[C(G_y(\mathbf{x})) = 1] - \Pr_{\mathbf{z}\sim\mathcal{U}_n}[C(\mathbf{z}) = 1] \right| < \varepsilon(n),$$

and $M(x,y)$ outputs $G_y(x)$ in nondeterministic $2^{O(\ell(n))}$ time.

3. There is at least one input y such that $M(x,y)$ does not reject.

Then we say M is a *nondeterministic pseudo-random generator (NPRG)* with seed length $\ell(n)$ that ε-fools $\mathscr{C}$ circuits of size S.

If the above conditions hold for infinitely many n's, then we say M is an *infinitely often NPRG* (i.o. NPRG).

Although NPRG in general does not output *the same PRG* in its different nondeterministic branches, it is still useful for many tasks such as derandomizing MA. Also, we can potentially shorten the seed length by allowing different PRGs in different nondeterministic branches. The concept of NPRG is implicit in [IKW02], and also used in [Che19].

We also remark that an NPRG implies an E^{NP}-computable PRG: we can find the lexicographically smallest string y such that $M(x,y)$ accepts in E^{NP}, and output $G_y(x)$.

2.4 Approximate Linear Sum of Circuits

Approximate linear sum of circuits, first explicit studied in [CW19], will be used heavily in this paper. We recall its definition below.

Definition 2.9. Let $\mathscr{C}$ be a class of functions from $\{0,1\}^n \to \{0,1\}$ and $\varepsilon \in [0, 0.5)$. We say $f : \{0,1\}^n \to \{0,1\}$ admits a $\widetilde{\mathsf{Sum}}_\varepsilon \circ \mathscr{C}$ circuit of sparsity S, if there are S functions $C_1, C_2, \ldots, C_S$ from $\mathscr{C}$, together with S coefficients $\alpha_1, \alpha_2, \ldots, \alpha_S$ in $\mathbb{R}$, such that for all $x \in \{0,1\}^n$,

$$\left| \sum_{i=1}^{S} \alpha_i \cdot C_i(x) - f(x) \right| \leq \varepsilon.$$

In particular, if $\varepsilon = 0$, we say f admits a $\mathsf{Sum} \circ \mathscr{C}$ circuit (of sparsity S).

Note when $\mathscr{C}$ is the class of AND gates (or $\oplus$/XOR gates, respectively), we are asking for the *sparsest* ε-approximate polynomial for f, with respect to the standard (or Fourier basis, respectively). This is closely related to the ε-approximate degree[21] of f, which is already a highly-nontrivial notion; for instance, the approximate degrees of simple natural functions have only recently been determined [BT17, BKT18, She18].

We define the *size* of a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit to be the total size of all its sub $\mathscr{C}$ circuits.

Remark 2.10. By [CW19, Proposition 6.8], we may assume that the $\widetilde{\mathsf{Sum}}_\varepsilon \circ \mathscr{C}$ circuits have "reasonable" coefficients. In particular, suppose $\varepsilon \in [0, 0.5)$ be a rational number j/k where $\max\{|j|, |k|\} \leq 2^b$. For every $\widetilde{\mathsf{Sum}}_\varepsilon \circ \mathscr{C}$ circuit C of sparsity s, there is an equivalent $\widetilde{\mathsf{Sum}}_\varepsilon \circ \mathscr{C}$ circuit C' of sparsity also s, in which every weight α_i in the linear combination can be written as j/k where both j, k are integers in $[-s^{\mathrm{poly}(s,b)}, s^{\mathrm{poly}(s,b)}]$.

[21] The ε-approximate degree of f is the lowest degree of all polynomial $p : \{0,1\}^n \to \{0,1\}$ such that $\|p - f\|_\infty \leq \varepsilon$. Note that a low degree polynomial is also sparse.

2.5 The Boolean Formula Evaluation Problem (Formula-Eval)

Our results rely on PCPP for Formula-Eval. Here we discuss the precise definition of this problem.

We define Formula-Eval as a *pair language*, *i.e.* a subset of $\{0,1\}^* \times \{0,1\}^*$. Intuitively, let C be a formula and w be an input, then $(C, w) \in$ Formula-Eval if and only if $C(w) = 1$. However, we need to specify the exact input format of the problem Formula-Eval so that it is indeed in NC^1.

Let C be a formula over n variables with S leaves. We encode C as a pair (T, L), where $T \in \{0,1\}^{S-1}$ represents a complete binary tree with S leaves, and $L : [S] \to [n]$ is a map from leaves to variables. In particular, C has $2S-1$ gates, and for every $1 \le i \le 2S-1$, the i-th gate of C is defined as follows.

- If $i > S-1$, then the i-th gate is a leaf, and its value is equal to the $L(i-S+1)$-th input bit.
- If $i \le S-1$, then the i-th gate receives as two inputs from the $(2i)$-th gate and the $(2i+1)$-th gate. If $T_i = 0$, then the i-th gate is an AND gate, otherwise it is an OR gate.

Let $C = (T, L)$ be a formula over n variables, and $w \in \{0,1\}^n$. We define $(C, w) \in$ Formula-Eval if the first gate of C evaluates to 1 when given w as input.

It is easy to see that Formula-Eval is indeed in NC^1.

2.6 Error Correcting Codes

We also need standard constructions of constant-rate linear error correcting codes.

Lemma 2.11 ([Spi96])**.** *There is a constant $\delta > 0$ such that there is a constant-rate linear error correcting code* ECC *with minimum relative distance δ, an efficient encoder* Enc *and an efficient decoder* Dec *recovering error up to $c_1 \cdot \delta$, where c_1 is a universal constant. Moreover, both* Enc *and* Dec *can be implemented in* NC^1.

We will use a slight modification of the above codes by [CW19], which is convenient when we want to guess-and-verify a circuit for the encoder.

Theorem 2.12 ([CW19, Lemma 2.10])**.** *There is a constant $\delta > 0$ such that there is a constant-rate linear error correcting code* ECC *with minimum relative distance δ, an efficient encoder* Enc*, and an efficient decoder* Dec *recovering error up to $c_1 \cdot \delta$, where c_1 is a universal constant. Moreover, each bit of the codeword depends on at most $n/2$ bits of the input, and both* Enc *and* Dec *can be implemented in* NC^1.

2.7 Probabilistic Checkable Proofs of Proximity (PCPP)

The concept of probabilistically checkable proofs of proximity is useful for this paper. In the following, we introduce its definition and an instantiation for Formula-Eval.

Definition 2.13 (Probabilistic Checkable Proofs of Proximity (PCP of proximity, or PCPP))**.** For $c, s, \delta : \mathbb{N} \to [0,1]$ and $r, q : \mathbb{N} \to \mathbb{N}$, a verifier V is a PCP of proximity system for a pair language $L \subseteq \{0,1\}^* \times \{0,1\}^*$ with proximity parameter δ, completeness parameter c, soundness parameter s, number of random bits r, and query complexity q if the following hold for all x, y:

- If $(x, y) \in L$, there is a proof π such that $V(x)$ accepts oracle $y \circ \pi$ with probability at least $c(|x|)$;
- if y is $\delta(|x|)$-far from $L(x) := \{z : (x, z) \in L\}$, then for all proofs π, $V(x)$ accepts oracle $y \circ \pi$ with probability at most $s(|x|)$;

- $V(x)$ tosses $r(|x|)$ random coins, and makes at most $q(|x|)$ non-adaptive queries.

Lemma 2.14 ([CW19]). *For any constant $\delta > 0$ there are two constants $0 < s < c < 1$, such that there is a PCP of proximity system for* Formula-Eval *with proximity δ, soundness s, completeness c, random bits $r = O(\log n)$, and query complexity $q = 2$, where each query is simply an* OR *on two bits or their negations. Moreover, there is a circuit of depth $O(\log |C| + \log |w|)$ such that, given a pair $(C, w) \in$* Formula-Eval*, outputs a proof π that makes $V(C)$ accepts with probability $\geq c$.*

We provide a proof sketch of Lemma 2.14 in Appendix A. In particular, we verify that the PCPP proof can be computed in low depth.

2.8 Hardness Amplification

We define black-box hardness amplification.

Definition 2.15. A $(1/2 - \varepsilon, \delta)$*-black-box hardness amplification* from input length k to input length $n = n(k)$ is a pair of oracle Turing machines $(\mathsf{Amp}, \mathsf{Dec})$, where Amp takes an oracle of k-bit inputs and computes an n-bit function, and Dec takes an oracle of n-bit inputs and an advice string of length $a = a(k)$, and computes a k-bit function. Furthermore, for every pair of functions $f : \{0,1\}^k \to \{0,1\}$ and $h : \{0,1\}^n \to \{0,1\}$, if

$$\Pr_{\mathbf{x} \sim \mathcal{U}_n} [h(\mathbf{x}) = \mathsf{Amp}^f(\mathbf{x})] \geq 1/2 + \varepsilon,$$

then there is a string $\alpha \in \{0,1\}^{a(k)}$, such that

$$\Pr_{\mathbf{x} \sim \mathcal{U}_k} [f(\mathbf{x}) = \mathsf{Dec}^h(\mathbf{x}, \alpha)] \geq 1 - \delta.$$

The name "hardness amplification" is justified by the fact that, to prove Amp^f cannot be $(1/2 + \varepsilon)$-approximated by some computational resources $\mathcal{C}$, we only need to prove that f cannot be $(1 - \delta)$-approximated by $\mathsf{Dec}^{\mathcal{C}}$, which should not be too powerful compared with $\mathcal{C}$.

We need the following hardness amplification where Amp is very efficient.

Theorem 2.16. *Let $\varepsilon, \delta > 0$. There is a $(1/2 - \varepsilon, \delta)$-black-box hardness amplification $(\mathsf{Amp}, \mathsf{Dec})$ from input length n to input length $\ell = O(n\delta^{-1} \log(\varepsilon^{-1}))$, where* Amp *is a* $\mathrm{poly}(\ell)$*-time uniform* $\mathsf{AC}^0[\oplus]$ *oracle circuit, and* Dec *is a nonadaptive* TC *oracle circuit of size* $\mathrm{poly}(\ell, \varepsilon^{-1})$ *with* $\mathrm{poly}(\ell, \varepsilon^{-1})$*-bit advice.*

Proof Sketch. This is essentially Yao's XOR lemma [Yao82], and a good exposition can be found in Section 19 of [AB09]. The complexity of Dec is the same as the complexity of Impagliazzo's hardcore lemma ([AB09, Section 19.1.2]), which is in TC^0.

Alternatively, this theorem can be proved by combining the local decoder for Walsh-Hadamard codes [GL89] (which is in TC^0) with the local decoder for direct-product codes [IJKW10] (which is in AC^0). □

We also need a black-box hardness amplification that amplifies worst-case hardness.

Theorem 2.17 ([GR08]). *Let $\varepsilon > 2^{-c\sqrt{n}}$ for some absolute constant c. There is a $(1/2 - \varepsilon, 0)$-black-box hardness amplification $(\mathsf{Amp}, \mathsf{Dec})$ from input length n to input length $O(n)$, where* Amp *runs in exponential time, and* Dec *is a* TC^0 *oracle circuit of size* $\mathrm{poly}(n, \varepsilon^{-1})$ *with* $\mathrm{poly}(\varepsilon^{-1})$*-bit advice.*

2.9 $\oplus$L-complete Problems with Good Properties

The $\oplus$L-complete problems with good reducibility properties will be important for us. (Recall that $\oplus$L is the class of problems solvable by a nondeterministic logspace Turing machine that accepts the input if the number of accepting paths is odd.) We define the following two problems, called Connected Matrix Determinant (CMD) and Decomposed Connected Matrix Determinant (DCMD):

Definition 2.18. An instance of CMD is an $n \times n$ matrix over $\mathsf{GF}(2)$ where the main diagonal and above may contain either 0 or 1, the second diagonal (*i.e.* the one below the main diagonal) contains 1, and other entries are 0. In other words, the matrix is of the following form (where $*$ represents any element in $\mathsf{GF}(2)$):

$$\begin{pmatrix} * & * & * & \cdots & * & * \\ 1 & * & * & \cdots & * & * \\ 0 & 1 & * & \cdots & * & * \\ 0 & 0 & 1 & \cdots & * & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & * \end{pmatrix}.$$

The instance is an $(n(n+1)/2)$-bit string specifying elements on and above the main diagonal. We define $x \in \mathsf{CMD}$ if and only if the determinant (over $\mathsf{GF}(2)$) of the matrix corresponding to x is 1.

An instance of DCMD is a string of length $n^3(n+1)/2$. For an input x, $\mathsf{DCMD}(x)$ is computed as follows: we partition x into blocks of length n^2, let $y_i (1 \leq i \leq n(n+1)/2)$ be the parity of the i-th block, and define $\mathsf{DCMD}(x) := \mathsf{CMD}(y_1 \circ y_2 \circ \cdots \circ y_{n(n+1)/2})$.

The precise definitions of CMD and DCMD are not important here, as we only need the following two important facts about them.

Theorem 2.19 ([AIK06, GGH+07])**.** *There is a function $P : \{0,1\}^{n(n+1)/2} \times \{0,1\}^{O(n^4)} \to \{0,1\}^{n^3(n+1)/2}$ such that the following hold.*

- *For any input $x \in \{0,1\}^{n(n+1)/2}$, the random variable $P(x, \mathcal{U}_{O(n^4)})$ is uniformly distributed in $\{0,1\}^{n^3(n+1)/2}$.*
- *For any $x \in \{0,1\}^{n(n+1)/2}$ and $r \in \{0,1\}^{O(n^4)}$, let $P(x,r) = y$, then $\mathsf{CMD}(x) = \mathsf{DCMD}(y) \oplus r_0$, where r_0 is the first bit of r.*
- *For each fixed randomness r, $P(x,r)$ is a projection over x, computable in polynomial time given r.*

Theorem 2.20 ([IK02])**.** CMD *is $\oplus$L-complete under projections.*

We provide self-contained proofs for the above theorems in Appendix B for completeness. We also refer the interested readers to Section 4 of [Vio09b] that contains an excellent exposition of these theorems.

Remark 2.21. Theorem 2.19 is essentially a *decomposable randomized encoding* (see, e.g. [App17]) of $\oplus$L. Such a randomized encoding of NC^1 can also be obtained via Yao's garbled circuit [Yao86, BHR12], which is also enough for our application.

2.10 Elementary Properties of Norm and Inner Product

Finally, we discuss some properties of norms and inner product of functions on Boolean cubes witch will be useful for us.

For a function $f : \{0,1\}^n \to \mathbb{R}$, we define its ℓ_p-norm as

$$\|f\|_p = \left(\mathop{\mathbb{E}}_{x \sim \mathcal{U}_n} [|f(x)|^p] \right)^{1/p}.$$

In particular, the ℓ_∞-norm is defined as the maximum absolute value of f:

$$\|f\|_\infty = \max_{x \in \{0,1\}^n} |f(x)|.$$

For two functions $f, g : \{0,1\}^n \to \mathbb{R}$, we define their inner product as

$$\langle f, g \rangle := \mathop{\mathbb{E}}_{x \sim \mathcal{U}_n} [f(x) \cdot g(x)].$$

Note that the Cauchy-Schwarz inequality implies $|\langle f, g \rangle| \leq \|f\|_2 \cdot \|g\|_2$. In particular, $\|f\|_1 = \langle |f|, \mathbf{1} \rangle \leq \|f\|_2$ where $\mathbf{1}$ is the all-one function. We need the following simple lemma for this paper.

Lemma 2.22 (see, *e.g.* [CW19, Lemma 28])**.** *For functions f_1, f_2 and g_1, g_2 from $\{0,1\}^n \to \mathbb{R}$ and positive $\varepsilon, \alpha \in \mathbb{R}$, suppose for all $i \in [2]$ we have:*

- $\|f_i\|_2 \leq \alpha$ *and* $\|g_i\|_2 \leq \alpha$,
- $\|f_i - g_i\|_2 \leq \varepsilon$.

Then $|\langle f_1, f_2 \rangle - \langle g_1, g_2 \rangle| \leq 2 \cdot \alpha \cdot \varepsilon$.

Proof. We have

$$\begin{aligned} |\langle f_1, f_2 \rangle - \langle g_1, g_2 \rangle| &\leq |\langle f_1, f_2 \rangle - \langle f_1, g_2 \rangle| + |\langle f_1, g_2 \rangle - \langle g_1, g_2 \rangle| \\ &\leq |\langle f_1, f_2 - g_2 \rangle| + |\langle f_1 - g_1, g_2 \rangle| \\ &\leq 2 \cdot \alpha \cdot \varepsilon. \end{aligned}$$

□

3 A Strong Collapse Theorem for $\oplus\mathsf{L}$

By using CMD and DCMD, we are able to prove a theorem stronger than [Che19, Theorem 5.3], namely that if $\oplus\mathsf{L}$ can be $(1/2 + \varepsilon)$-approximated by $\mathscr{C}$ circuits, then $\oplus\mathsf{L}$ has small $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ circuits. In contrast, Theorem 5.3 of [Che19] only works for $(1 - \delta)$-approximation for some small constant δ. Furthermore, the *sum of absolute values of coefficients* of the $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit is also small (polynomially related to its size), which will be useful in Section 5.

Lemma 3.1. *Let $\mathscr{C}$ be a typical circuit class and $\varepsilon : \mathbb{N} \to (0, 1/2]$ be an error parameter. Suppose that on input length $n^3(n+1)/2$,* DCMD *can be $(1/2 + \varepsilon(n))$-approximated by a $\mathscr{C}$ circuit. Then for every $\delta > 0$,* CMD *on input length $n(n+1)/2$ has a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of sparsity $O(\varepsilon(n)^{-2}\delta^{-2}n^2)$. Moreover, the sum of absolute values of all coefficients of the $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit is $O(\varepsilon(n)^{-1})$.*

Proof. Let C be the $\mathscr{C}$ circuit that $(1/2+\varepsilon(n))$-approximates DCMD and suppose

$$\varepsilon'(n) = \Pr_{\mathbf{x}\sim\mathcal{U}_{n^3(n+1)/2}}[C(\mathbf{x}) = \mathsf{DCMD}(\mathbf{x})] - 1/2,$$

then $\varepsilon'(n) \geq \varepsilon(n)$.

On input length $|x| = n(n+1)/2$, let $P(x,r)$ be the randomized projection given in Theorem 2.19 such that $\mathsf{CMD}(x) = \mathsf{DCMD}(P(x,r)) \oplus r_0$ for all r, and $P(x,\mathbf{r})$ is uniformly distributed in $\{0,1\}^{n^3(n+1)/2}$ if $\mathbf{r}$ is drawn from $\mathcal{U}_{O(n^4)}$.

Let $\mathcal{D}$ be a distribution of $\mathscr{C}$-circuits such that a sample D from $\mathcal{D}$ is generated by setting $D(x) := C(P(x,\mathbf{r})) \oplus \mathbf{r}_0$ where $\mathbf{r}$ is drawn from $\mathcal{U}_{O(n^4)}$. Let $\mathbf{D} \sim \mathcal{D}$. Then for any $x \in \{0,1\}^{n(n+1)/2}$, $\Pr[\mathbf{D}(x) = \mathsf{CMD}(x)] = 1/2 + \varepsilon'(n)$. The reason is that $\mathbf{D}(x) = \mathsf{CMD}(x)$ if and only if $C(P(x,\mathbf{r})) = \mathsf{DCMD}(P(x,\mathbf{r}))$, which happens with probability exactly $1/2+\varepsilon'(n)$ since $P(x,\mathbf{r})$ is uniformly distributed.

Now, we draw $t = \Theta(\varepsilon'(n)^{-2}\delta^{-2}n^2)$ i.i.d. random samples $C_1, C_2, \ldots, C_t$ from $\mathcal{D}$, and by a Chernoff bound, for any particular $x \in \{0,1\}^{n(n+1)/2}$, we have

$$\Pr\left[\Pr_i[C_i(x) = \mathsf{CMD}(x)] \in \left(1/2 + (1-2\delta)\varepsilon'(n), 1/2 + (1+2\delta)\varepsilon'(n)\right)\right] > 1 - 2^{-2n^2}.$$

By a union bound, we can select $C_1, C_2, \ldots, C_t$ such that for every $x \in \{0,1\}^{n(n+1)/2}$,

$$\Pr_i[C_i(x) = \mathsf{CMD}(x)] \in \left(1/2 + (1-2\delta)\varepsilon'(n), 1/2 + (1+2\delta)\varepsilon'(n)\right).$$

To obtain a valid $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ for CMD, we can simply scale the coefficients accordingly.

Consider the following linear combination of $\mathscr{C}$ circuits, which has sparsity $t+1$:

$$E(x) = -\frac{1}{4\varepsilon'(n)} + \frac{1}{2} + \sum_{i=1}^{t} \frac{1}{2\varepsilon'(n)\cdot t} \cdot C_i(x). \tag{3}$$

If $\mathsf{CMD}(x) = 1$, it follows that

$$\sum_{i=1}^{t} \frac{1}{2\varepsilon'(n)\cdot t} \cdot C_i(x) \in \left(\frac{1}{4\varepsilon'(n)} + \frac{1}{2} - \delta, \frac{1}{4\varepsilon'(n)} + \frac{1}{2} + \delta\right)$$

and therefore $E(x) \in (1-\delta, 1+\delta)$.

Otherwise, $\mathsf{CMD}(x) = 0$, then

$$\sum_{i=1}^{t} \frac{1}{2\varepsilon'(n)\cdot t} \cdot C_i(x) \in \left(\frac{1}{4\varepsilon'(n)} - \frac{1}{2} - \delta, \frac{1}{4\varepsilon'(n)} - \frac{1}{2} + \delta\right)$$

and therefore $E(x) \in (-\delta, \delta)$. Hence, E is a valid $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit that computes CMD.

It can be seen from (3) that the sum of absolute values of all coefficients in E is at most $\frac{t}{2\varepsilon'(n)\cdot t} + O(1) = O(\varepsilon(n)^{-1})$. □

By the standard Discriminator Lemma [HMP+93], we have the following corollary.

Corollary 3.2. *Let $\mathscr{C}$ be a typical circuit class. If $\oplus\mathsf{L} \subseteq \mathsf{MAJ} \circ \mathscr{C}$, then $\oplus\mathsf{L} \subseteq \widetilde{\mathsf{Sum}}_{1/n^{10}} \circ \mathscr{C}$.*

4 Certifying Depth hardness Implies Strong Average-case Circuit Lower Bounds

To prove the strong average-case hardness, we need to adopt the techniques from [Che19]. We first define the notion of certifying depth hardness in NE.

Definition 4.1. Let $d : \mathbb{N} \to \mathbb{N}$ be a function. We say NE *can certify* $d(n)$*-depth hardness*, if there exists a verifier $V(x, y)$ taking inputs x, y with $|y| = 2^{|x|}$ and running in $2^{O(|x|)}$ time, such that the following hold for infinitely many n's: $V(1^n, \cdot)$ is satisfiable, and $V(1^n, y) = 1$ implies that y (interpreted as a function from $\{0,1\}^n \to \{0,1\}$) cannot be computed by $d(n)$-depth circuits.

In this section, we show that certifying hardness in NE implies average-case lower bounds for various nondeterministic time classes. The following theorem can be proved using similar ideas from [Che19]. We remark that the second and the third item below already essentially follow from [Che19, Section 10], while for the first item we need to tighten some technical building blocks in [Che19].

Theorem 4.2 (Certifying Depth hardness to Average-case Circuit Lower Bounds)**.** *The followings hold.*

*(*NP *Average-Case Lower Bound) If* NE *can certify* $\Omega(n)$*-depth hardness, then for every constant* $k \geq 1$*,* NP *cannot be* $(1/2 + 1/n^k)$*-approximated by circuits of* $k \log n$ *depth. The same holds for* $(\mathsf{NP} \cap \mathsf{coNP})_{/1}$ *in place of* NP*.*

*(*NQP *Average-Case Lower Bound) If* NE *can certify* $n^{\Omega(1)}$*-depth hardness, then for all constants* $k \geq 1$*,* NQP *cannot be* $(1/2 + 1/2^{\log^k n})$*-approximated by circuits of* $\log^k n$ *depth. The same holds for* $(\mathsf{N} \cap \mathsf{coN})\mathsf{QP}_{/1}$ *in place of* NQP*.*

*(*NEXP *Average-Case Lower Bound) If* NE *can certify* $\omega(\log n)$*-depth hardness, then* NE *cannot be* $(1/2 + 1/\mathrm{poly}(n))$*-approximated by circuits of* $O(\log n)$ *depth. The same holds for* $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$ *in place of* NE*.*

In the following, we provide a proof for the first Item (which is the hardest case).[22] In Appendix D.1, we include a proof sketch of the general tradeoff (Theorem D.2) between certifying depth hardness and average-case circuit lower bounds, which implies all three items of Theorem 4.2.

Remark 4.3 (The Easy Witness Lemma Perspective)**.** We remark that one can also view Theorem 4.2 as a certain easy witness lemma (which is similar to that of [IKW02, MW18]) for *unary languages*.[23] In particular, the contrapositive of first item of Theorem 4.2 is that if NP *can* be $(1/2 + 1/n^k)$-approximated by circuits of $k \log n$ depth for some constant k, then all unary languages in NE have $\varepsilon \cdot n$-depth witness for all $\varepsilon > 0$.

4.1 Building Blocks: Low Depth PRG and A.a.e. MA ∩ coMA Average-case Circuit Lower Bounds

We list the tightened technical building blocks as follows. Their proofs can be found in Appendix C.

[22] For more intuition on the proof, the reader is referred to [Che19, Section 1.3].

[23] See also [Che19, Lemma 4.1].

First, we need a standard construction of PRGs from functions which are *worst-case* hard for low depth circuits. This essentially follows from [STV01] together with a low depth computable extractor.

Theorem 4.4 (Implicit in [STV01]). *Let $d(\ell) = \omega(\log \ell)$. There are universal constants c and g, and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for every good resource function $d : \mathbb{N} \to \mathbb{N}$, the following hold. Let $s_{\mathsf{seed}} = g\ell^2/d(\ell)$ and $S_{\mathsf{out}} = 2^{c \cdot d(\ell)}$, if $Y : \{0,1\}^\ell \to \{0,1\}$ does not have circuits of depth $d(\ell)$, then for all circuits C with depth $\log(S_{\mathsf{out}})$,*

$$\left| \Pr_{\mathbf{t} \sim \mathcal{U}_{s_{\mathsf{seed}}}} [C(G(Y, \mathbf{t})) = 1] - \Pr_{\mathbf{x} \sim \mathcal{U}_{S_{\mathsf{out}}}} [C(\mathbf{x}) = 1] \right| < 1/S_{\mathsf{out}}.$$

That is, $G(Y, \cdot)$ $1/S_{\mathsf{out}}$-fools all $\log(S_{\mathsf{out}})$-depth circuits. Moreover, G is computable in $2^{O(s_{\mathsf{seed}})}$ time.

Next, we need an almost almost-everywhere (a.a.e.) $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ average-case lower bound against low depth circuits.[24] [Che19] only proves such an average-case lower bound for $(\mathsf{MAQP} \cap \mathsf{coMAQP})_{/1}$. The proof for the theorem below follows the same steps as in [Che19], but with some technical improvements so that it also works for $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ now.

Theorem 4.5. *There is a universal constant $d \in \mathbb{N}$ such that the following hold. For all integers $a > 0$, there are constants $b, t > 0$, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[n^b]_{/1}$, such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*

- $\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L_n) > a \cdot \log n$, *or*
- $\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L_m) > a \cdot \log m$, *for some $m \in (n^t, 2n^t) \cap \mathbb{N}$.*

Moreover, the predicate depth of L is $\leq b \log n$.

4.2 $(1 - 1/\mathrm{poly}(n))$-Inapproximability Results

We first prove $(\mathsf{NP} \cap \mathsf{coNP})_{/2}$ cannot be $(1 - 1/\mathrm{poly}(n))$-approximated by low depth circuits. Then in Section 4.3 we use standard hardness amplification (Theorem 2.16) to amplify the inapproximability to $(1/2 + 1/\mathrm{poly}(n))$. We also use the enumeration trick [COS18] to show the same lower bound for NP and $(\mathsf{NP} \cap \mathsf{coNP})_{/1}$.

We remark that the proof essentially follows the same structure of [Che19, Section 9], with technical building blocks tightened in Section 4.1.

Theorem 4.6. *Let d be the universal constant d in Theorem 4.5. If NE can certify $\Omega(n)$-depth hardness, then for every constant $k \geq 1$, there is some $b \in \mathbb{N}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[n^b]_{/2}$ cannot be $(1 - n^{-d})$-approximated by circuits of $k \log n$ depth.*

Proof. Let $V(\cdot, \cdot)$ be the NE verifier as in Definition 4.1 that can certify εn-depth hardness, for some constant $\varepsilon > 0$. By Theorem 4.4, there are two absolute constants $c, g > 0$ and an i.o. NPRG that stretches $s_{\mathsf{seed}} = g\varepsilon^{-1} \cdot \ell_{\mathsf{PRG}}$ random bits into $S_{\mathsf{out}} = S_{\mathsf{out}}(\ell_{\mathsf{PRG}}) = 2^{c\varepsilon \cdot \ell_{\mathsf{PRG}}}$ pseudorandom bits, and $1/S_{\mathsf{out}}$-fools all depth-$\log(S_{\mathsf{out}})$ circuits.

In particular, on input $t \in \{0,1\}^{s_{\mathsf{seed}}}$, the i.o. NPRG computes $\ell_{\mathsf{PRG}} = \varepsilon|t|/g$ and guesses some $y_{\mathsf{hard}} \in \{0,1\}^{2^{\ell_{\mathsf{PRG}}}}$ such that $V(1^{\ell_{\mathsf{PRG}}}, y_{\mathsf{hard}})$ accepts. It then computes $G(y_{\mathsf{hard}}, t)$ in $2^{O(\ell_{\mathsf{PRG}})}$ time,

[24] This notion is borrowed from [MW18], meaning it is "almost" an almost-everywhere circuit lower bound.

where G is the function specified in Theorem 4.4. For infinitely many ℓ_{PRG}'s. $V(1^{\ell_{\mathsf{PRG}}}, \cdot)$ is satisfiable, and for every $y_{\mathsf{hard}} \in \{0,1\}^{2^{\ell_{\mathsf{PRG}}}}$ such that $V(1^{\ell_{\mathsf{PRG}}}, y_{\mathsf{hard}})$ accepts, y_{hard} is the truth table of some function that cannot be computed by circuits of depth $\varepsilon \cdot \ell_{\mathsf{PRG}}$. Therefore, by Theorem 4.4, $G(y, \cdot)$ $1/S_{\mathsf{out}}$-fools all depth-$\log(S_{\mathsf{out}})$ circuits. We call these ℓ_{PRG}'s *good for NPRG*.

By Theorem 4.5, there are constants d, t, b' and a language $L^{\mathsf{hard}} \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[n^{b'}]_{/1}$, such that for every sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either

- $\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_n) > (k+1) \cdot \log n$, or
- $\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_m) > (k+1) \cdot \log m$ for some $m \in (n^t, 2n^t) \cap \mathbb{N}$.

Moreover, the predicate depth of L^{hard} is $\leq b' \log n$. In other words, let $\{\alpha_n\}_{n\in\mathbb{N}}$ be the sequence of advice bits for L^{hard}, and $A(x, y, z, a)$ be L^{hard}'s $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ predicate. Here x is the input, y is the proof from Merlin, z is the randomness used by Arthur and a is the advice bit. (See Definition 2.1.) Then for any possible x, y and the correct advice bit $\alpha_{|x|}$, the function $A(x, y, \cdot, \alpha_{|x|})$ (over randomness z) can be implemented by a circuit $C_{x,y}$ of depth $\leq b' \log |x|$.

In the following, we apply the nondeterministic PRG to derandomize the $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm for L^{hard} above, and put the hard language in $(\mathsf{NP} \cap \mathsf{coNP})_{/2}$.

Technical definitions. We need to introduce some technical definitions first.

Let $n \in \mathbb{N}$, we define $\ell(n) = \lceil 2b't \log n/(c\varepsilon) \rceil$. Recall that $S_{\mathsf{out}}(\ell(n)) = 2^{c\varepsilon \cdot \ell(n)}$, then $S_{\mathsf{out}}(\ell(n)) \geq (2n^t)^{b'}$. The point is that if any $\ell_{\mathsf{PRG}} \geq \ell(n)$ is good for NPRG, then we can use the i.o. NPRG mentioned above on input $1^{\ell_{\mathsf{PRG}}}$, and successfully derandomize L^{hard} on any input length $\leq 2n^t$.

We also need an injective function $\mathsf{pair}(m, n)$ that encodes two positive integers into one positive integer, and satisfies the following:

- For every $m, n \in \mathbb{N}$, $\max(m, n) \leq \mathsf{pair}(m, n) \leq O(mn^2)$.
- Given $m, n \in \mathbb{N}$ in binary, we can compute $\mathsf{pair}(m, n)$ in $O(\log m + \log n)$ time.
- Given $\mathsf{pair}(m, n)$ in binary, we can compute m and n in $O(\log m + \log n)$ time.

Such an encoding is easy to construct: see *e.g.* [Che19, Section 9.1]. [25]

The language L. Formally, we define a language $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[n^b]_{/2}$ as follows. On input length $\widetilde{n}$, we first decode $\widetilde{n} = \mathsf{pair}(n, \ell_{\mathsf{PRG}})$. If there does not exist $n, \ell_{\mathsf{PRG}} \in \mathbb{Z}$ such that $\mathsf{pair}(n, \ell_{\mathsf{PRG}}) = \widetilde{n}$, we reject every input. Otherwise we receive two advice bits $\alpha'_{\widetilde{n}}$ and $\beta_{\widetilde{n}}$, where $\alpha'_{\widetilde{n}} = \alpha_n$ is the advice bit of L^{hard} on input length n, and $\beta_{\widetilde{n}}$ is 1 if and only if ℓ_{PRG} is good for NPRG. The language is defined as follows.

If any of the following are true, we reject every input of length $\widetilde{n}$:

- $\beta_{\widetilde{n}} = 0$, *i.e.* ℓ_{PRG} is not good for NPRG,
- $\ell_{\mathsf{PRG}} > 2\ell(n)$, *i.e.* ℓ_{PRG} is too large that we cannot afford run time $2^{O(\ell_{\mathsf{PRG}})}$, or
- $c\varepsilon \cdot \ell_{\mathsf{PRG}} < b' \log n$.

[25] In [Che19], the function $\mathsf{pair}(m, n)$ is defined as follows. For a number $a \in \mathbb{N}$, let $\mathsf{bin}(a)$ be a string of length $\lfloor \log a \rfloor + 1$ that denotes the binary representation of a. Denote $\ell = |\mathsf{bin}(n)|$, we duplicate each bit of $\mathsf{bin}(\ell)$ and obtain a string z_{len}. (For example, if $\ell = 5$, then $\mathsf{bin}(\ell) = 101$ and $z_{\mathsf{len}} = 110011$.) Then we let $z = \mathsf{bin}(m) \circ \mathsf{bin}(n) \circ 01 \circ z_{\mathsf{len}}$, where $\circ$ means concatenation. We define $\mathsf{pair}(m, n)$ be the integer with binary representation z.

(The last item means that ℓ_{PRG} is too small. In particular, recall that $S_{\mathsf{out}} = S_{\mathsf{out}}(\ell_{\mathsf{PRG}}) = 2^{c\varepsilon \cdot \ell_{\mathsf{PRG}}}$, it follows $\log(S_{\mathsf{out}}) = c\varepsilon \cdot \ell_{\mathsf{PRG}}$ is less than the predicate depth $b' \log n$ of L_n^{hard}. Therefore the i.o. NPRG constructed above with seed length $g \cdot \ell_{\mathsf{PRG}}$ cannot derandomize L_n^{hard}, even if ℓ_{PRG} is good for NPRG.)

If none of the above happens, we say $\widetilde{n}$ is a *non-trivial input length*. Let $x \in \{0,1\}^{\widetilde{n}}$ where $\widetilde{n}$ is non-trivial. We define $x \in L$ if and only if $x' \in L^{\mathsf{hard}}$, where x' is the first n bits of x. That is, in this case, $L_{\widetilde{n}}$ essentially computes L_n^{hard}.

The $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[n^b]_{/2}$ algorithm for L. Let $\widetilde{n} = \mathsf{pair}(n, \ell_{\mathsf{PRG}})$ be a non-trivial length and $x \in \{0,1\}^n$ be the length-n prefix of some input, we show how to compute $L^{\mathsf{hard}}(x)$ in $(\mathsf{NP} \cap \mathsf{coNP})_{/2}$. We guess a string y_{hard} of length $2^{\ell_{\mathsf{PRG}}}$ such that $V(1^{\ell_{\mathsf{PRG}}}, y_{\mathsf{hard}})$ accepts, as well as the proof y that Merlin gives Arthur. We then use $G(y_{\mathsf{hard}}, \cdot)$ to fool the $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ predicate of L_n^{hard}. To be more precise, let $S_{\mathsf{out}} = 2^{c\varepsilon \cdot \ell_{\mathsf{PRG}}} \geq n^{b'}$. For every $\xi \in \{0,1\}$, let

$$p_{x,y}(\xi) = \Pr_{\mathbf{z} \sim \mathcal{U}_{S_{\mathsf{out}}}}[A(x, y, \mathbf{z}, \alpha_n) = \xi], \quad \text{and} \quad p'_{x,y}(\xi) = \Pr_{\mathbf{w} \sim \mathcal{U}_{g \cdot \ell_{\mathsf{PRG}}}}[A(x, y, G(y_{\mathsf{hard}}, \mathbf{w}), \alpha_n) = \xi].$$

Since ℓ_{PRG} is good for NPRG, and the predicate depth of A is $\leq \log(S_{\mathsf{out}})$, it follows that

$$|p_{x,y}(\xi) - p'_{x,y}(\xi)| < 1/S_{\mathsf{out}}.$$

We compute $p'_{x,y}(\xi)$ by enumerating the seed $w \in \{0,1\}^{g \cdot \ell_{\mathsf{PRG}}}$. If there is some $\xi \in \{0,1\}$ such that $p'_{x,y}(\xi) > 0.6$, then we output ξ. Otherwise we output ?.

Since A satisfies $\mathsf{MA} \cap \mathsf{coMA}$ promise, it is easy to see that our algorithm satisfies $\mathsf{NP} \cap \mathsf{coNP}$ promise:

- We only consider nontrivial input lengths $\widetilde{n} = \mathsf{pair}(n, \ell_{\mathsf{PRG}})$.
- For any input $x \in \{0,1\}^n$, there is an input y such that $p_{x,y}(L^{\mathsf{hard}}(x)) \geq 2/3$. Hence $p'_{x,y}(L^{\mathsf{hard}}(x)) \geq 2/3 - 1/S_{\mathsf{out}} > 0.6$ and the nondeterministic machine above indeed outputs $L^{\mathsf{hard}}(x)$.
- For any input $x \in \{0,1\}^n$ and any input y, $p_{x,y}(1 - L^{\mathsf{hard}}(x)) \leq 1/3$. Hence $p'_{x,y}(1 - L^{\mathsf{hard}}(x)) < 1/3 + 1/S_{\mathsf{out}} < 0.6$ and the nondeterministic machine above never outputs $1 - L^{\mathsf{hard}}(x)$.

It follows that L satisfies $\mathsf{NP} \cap \mathsf{coNP}$ promise. The machine uses $2^{O(\ell_{\mathsf{PRG}})}$ time to generate y_{hard}, and $O(2^{g\varepsilon^{-1} \cdot \ell_{\mathsf{PRG}}} \cdot n^{b'})$ time to compute each $p'_{x,y}(\xi)$. Recall that $\ell_{\mathsf{PRG}} \leq 2\ell(n) = O(\log n)$, and thus $\widetilde{n} = O(n \log^2 n)$. Note that the above algorithm takes two advice bits $\alpha'_{\widetilde{n}}$ and $\beta_{\widetilde{n}}$ on inputs of length n, it follows that $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[\widetilde{n}^b]_{/2}$ for some large enough constant b.

The lower bound for L. Let ℓ_{PRG} be a large enough number that is good for NPRG, $\tau = \lceil \ell_{\mathsf{PRG}} \cdot (c\varepsilon)/(3b't) \rceil$, and $n = 2^\tau$. Then either

$$\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L_n^{\mathsf{hard}}) > (k+1) \cdot \log n, \tag{4}$$

or there is some $m \in (n^t, 2n^t) \cap \mathbb{N}$ such that

$$\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L_m^{\mathsf{hard}}) > (k+1) \cdot \log m. \tag{5}$$

If (4) holds, then let $\widetilde{n} = \mathsf{pair}(n, \ell_{\mathsf{PRG}})$. Since ℓ_{PRG} is good for NPRG, $2\ell(n) = 2\lceil 2b't \log n/(c\varepsilon)\rceil \geq \ell_{\mathsf{PRG}}$, and $b' \log n \leq \ell_{\mathsf{PRG}} \cdot c\varepsilon/t < c\varepsilon \cdot \ell_{\mathsf{PRG}}$, $\widetilde{n}$ is a non-trivial input length. Therefore $L_{\widetilde{n}}$ computes L^{hard}_n. Since $\widetilde{n} = O(n \log^2 n)$,

$$\mathsf{heur}_{(1-\widetilde{n}^{-d})}\text{-}\mathsf{DEPTH}(L_{\widetilde{n}}) > k \cdot \log \widetilde{n}.$$

If (5) holds, then let $\widetilde{m} = \mathsf{pair}(m, \ell_{\mathsf{PRG}})$. Since ℓ_{PRG} is good for NPRG, $2\ell(m) = 2\lceil 2b't \log m/(c\varepsilon)\rceil \geq \ell_{\mathsf{PRG}}$, and $b' \log m \leq 1.1b't \cdot \tau \leq c\varepsilon \cdot \ell_{\mathsf{PRG}}$, $\widetilde{m}$ is a non-trivial input length. Therefore $L_{\widetilde{m}}$ computes L^{hard}_m. Since $\widetilde{m} = O(m \log^2 m)$,

$$\mathsf{heur}_{(1-\widetilde{m}^{-d})}\text{-}\mathsf{DEPTH}(L_{\widetilde{m}}) > k \cdot \log \widetilde{m}.$$

It follows that for infinitely many input lengths n, we have

$$\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L_n) > k \cdot \log n. \qquad \square$$

4.3 Proof of Theorem 4.2

Now we are ready to prove Theorem 4.2, we first use standard hardness amplification to strengthen the previous inapproximability to $(1/2 + n^{-k})$.

Theorem 4.7. *If* NE *can certify $\Omega(n)$-depth hardness, then for every constant $k \geq 1$, there is some $b \in \mathbb{N}$ such that $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[n^b]_{/2}$ cannot be $(1/2 + n^{-k})$-approximated by circuits of $k \log n$ depth.*

Proof. Let a, k' be constants to be fixed later. Let d be the universal constant in Theorem 4.6, and L' be a language in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[n^b]_{/2}$ that cannot be $(1 - n^{-d})$-approximated by circuits of $a \log n$ depth. We need a $(1/2 - n^{-k'}, n^{-d})$-black-box hardness amplification $(\mathsf{Amp}, \mathsf{Dec})$ specified in Theorem 2.16, from input length n to input length $m(n) = n^{d+2}$. (The hardness amplification is actually from length n to length $\widetilde{O}(n^{d+1})$. For convenience, we pad it to length exactly n^{d+2}.)

We define a language L that on input length m:

- if $m = m(n)$ for some $n \in \mathbb{N}$, computes the language $\mathsf{Amp}^{L'_n}$;
- otherwise reject every input.

We prove that $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[m^{O(b)}]_{/2}$. On input length $m = m(n)$, the advice we receive is the advice $\alpha'_n \circ \beta_n$ for L''s *on input length* n. For every query to the oracle L'_n made by Amp (which is a string $x_{\mathsf{query}} \in \{0,1\}^n$), we guess the proof $y_{\mathsf{query}} \in \{0,1\}^{n^b}$ used by the machine for L' (call this machine $M_{L'}$) on input x_{query}. We then simulate $M_{L'}$. If $M_{L'}$ outputs ?, we immediately output ? and halt. Otherwise we treat the output of $M_{L'}$ as the return value of this oracle call. We can thus compute $\mathsf{Amp}^{L'_n}$ by an $(\mathsf{NP} \cap \mathsf{coNP})$ algorithm of time complexity $\mathrm{poly}(m, n^b) = m^{O(b)}$ with two advice bits.

Let the depth for Dec be $Ck' \log n$ for some constant C depending on d. For infinitely many n's,

$$\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L'_n) > a \cdot \log n.$$

Therefore by Theorem 2.16,

$$\mathsf{heur}_{(1/2+n^{-k'})}\text{-}\mathsf{DEPTH}(L_m) > (a - Ck') \log n,$$

which implies

$$\mathsf{heur}_{(1/2+m^{-k'/(d+2)})}\text{-}\mathsf{DEPTH}(L_m) > \frac{a - Ck'}{d+2} \log m.$$

Let $k' = k(d+2)$, $a = k(d+2) + Ck'$. It follows that for infinitely many m's,

$$\mathsf{heur}_{(1/2+m^{-k})}\text{-}\mathsf{DEPTH}(L_m) > k \log m. \qquad \square$$

Then we use the enumeration trick to put our hard language into NP and $(\mathsf{NP} \cap \mathsf{coNP})_{/1}$ respectively.

Reminder of Item 1 of Theorem 4.2. *If* NE *can certify $\Omega(n)$-depth hardness, then for every constant $k \geq 1$,* NP *cannot be $(1/2 + n^{-k})$-approximated by circuits of $k \log n$ depth. The same holds for* $(\mathsf{NP} \cap \mathsf{coNP})_{/1}$ *in place of* NP.

Proof. Let k' be some constant that we fix later, $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[n^b]_{/2}$ be the language specified in Theorem 4.7 that cannot be $(1/2 + n^{-k'})$-approximated by circuits of $k' \log n$ depth. Let $A(x, y, a)$ be the predicate of L' (Definition 2.4), where x is the input, y is the nondeterministically guessed proof and a is the advice.

NP lower bounds. We define a language L, computed by the following nondeterministic machine. On input $x \in \{0,1\}^m$, let $m_1 = \lfloor m/4 \rfloor$ and $m_2 = m \bmod 4$. We treat m_2 as a two-bit string corresponding to the advice for L'_{m_1}. Let x' be the first m_1 bits of x, we accept x if there is some y such that $A(x', y, m_2)$ accepts.

For every length m' such that

$$\mathsf{heur}_{(1/2+(m')^{-k'})}\text{-}\mathsf{DEPTH}(L'_{m'}) > k' \log(m'),$$

let $a_{m'} \in \{0,1,2,3\}$ be the advice used by $L'_{m'}$, and $m = 4m' + a_{m'}$. Then L_m computes exactly the same function as $L'_{m'}$ (except that L_m ignores all but the first m' bits of its inputs). It follows that there are infinitely many m's such that

$$\mathsf{heur}_{(1/2+\lfloor m/4 \rfloor^{-k'})}\text{-}\mathsf{DEPTH}(L_m) > k' \log \lfloor m/4 \rfloor.$$

If we let $k' = k + 1$, then

$$\mathsf{heur}_{(1/2+m^{-k})}\text{-}\mathsf{DEPTH}(L_m) > k \log m.$$

$(\mathsf{NP} \cap \mathsf{coNP})_{/1}$ lower bounds. We use the same language L as the NP lower bound, with a minor modification. Let m be the input length, $m_1 = \lfloor m/4 \rfloor$ and $m_2 = m \bmod 4$. If m_2 is the correct advice for L'_{m_1}, then the advice bit is 1 and we compute L'_{m_1}, otherwise the advice bit is 0 and we reject every input. If m_2 is the correct advice for L'_{m_1}, the machine for L'_{m_1} satisfies $\mathsf{NP} \cap \mathsf{coNP}$ promise and decides L_m. We can thus compute this (modified) L in $(\mathsf{NP} \cap \mathsf{coNP})_{/1}$.

The same argument as above shows that this modified L cannot be $(1/2 + m^{-k})$-approximated by circuits of depth $k \log m$, on infinitely many input lengths m. □

5 Non-trivial CAPP Algorithms Imply Strong Average-case Circuit Lower Bounds

In this section we prove the main theorem of the paper, establishing the connection between non-trivial CAPP algorithms and average-case circuit lower bounds. (Recall that CAPP means estimating the approximate probability of a circuit C *within error* $\pm 1/|C|$, where $|C|$ is the size of C.)

Reminder of Theorem 1.1. *Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. The following hold.*

(NP *Average-Case Lower Bound) Suppose there is a constant* $\varepsilon > 0$ *such that the* CAPP *problem of* $\mathsf{AND}_4 \circ \mathscr{C}$ *circuits of size* $2^{\varepsilon n}$ *can be solved in* $2^{n-\varepsilon n}$ *time. Then for every constant* $k \geq 1$, NP *cannot be* $(1/2 + n^{-k})$*-approximated by* $\mathscr{C}$ *circuits of* n^k *size.*

(NQP *Average-Case Lower Bound) Suppose there is a constant* $\varepsilon > 0$ *such that the* CAPP *problem of* $\mathsf{AND}_4 \circ \mathscr{C}$ *circuits of size* 2^{n^ε} *can be solved in* 2^{n-n^ε} *time. Then for every constant* $k \geq 1$, NQP *cannot be* $(1/2 + 2^{-\log^k n})$*-approximated by* $\mathscr{C}$ *circuits of* $2^{\log^k n}$ *size.*

(NEXP *Average-Case Lower Bound) Suppose the* CAPP *problem of* $\mathsf{AND}_4 \circ \mathscr{C}$ *circuits of size* $\mathrm{poly}(n)$ *can be solved in* $2^n/n^{\omega(1)}$ *time. Then* NE *cannot be* $1/2 + 1/\mathrm{poly}(n)$*-approximated by* $\mathscr{C}$ *circuits of* $\mathrm{poly}(n)$ *size.*

We also include a more general version of Theorem 1.1 (Theorem D.4) in Appendix D.2. We remark that all bullets of Theorem 1.1 are implied by Theorem D.4. In this section, we provide a detailed proof of Item 2 of Theorem 1.1.

To prove our theorem, we need the following Average-Product Estimation problem over functions from $\mathscr{C}$. (This problem can be seen as a relaxation of the Sum-Product problem defined in [Wil18].)

Average-Product Estimation over $\mathscr{C}$ within error δ: Given k functions $f_1, \ldots, f_k : \{0,1\}^n \to \mathbb{R}$ from $\mathscr{C}$, estimate

$$\mathop{\mathbb{E}}_{x \sim \mathcal{U}_n} \left[\prod_{i=1}^{k} f_i(x) \right]. \tag{6}$$

within absolute error δ.

Given a $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ circuit $f = \sum_{i \in [S]} \alpha_i \cdot C_i$, we define the *complexity* of f to be the maximum of its size and the sum of absolute values of its coefficients, *i.e.*

$$\max \left\{ \sum_{i \in [S]} |C_i|, \sum_{i \in [S]} |\alpha_i| \right\}.$$

Although the coefficients for a general $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit could be very large, in this section we mainly focus on linear combinations of reasonable complexity (*i.e.* polynomial in the circuit size).

The following theorem is the most important technical ingredient of this section.

Theorem 5.1. *Let* $\mathscr{C}$ *be a typical circuit class. There is a universal constant* $\delta > 0$ *such that, if the following hold:*

- CMD *has* $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ *circuits of complexity* $2^{\log^{O(1)} n}$*, and*
- *there is a constant* $\varepsilon > 0$ *such that the Average-Product of* 4 $\mathscr{C}$ *circuits of size* 2^{n^ε} *can be estimated within error* 2^{-n^ε}*, in* 2^{n-n^ε} *time.*

Then NE *can certify* $n^{\Omega(1)}$ *depth hardness.*

Before proving Theorem 5.1, we show that it implies item 2 of Theorem 1.1.

Proof of Theorem 1.1, Item (2). Fix $k \geq 1$. We assume DCMD can be $(1/2 + 2^{-\log^k n})$-approximated by $\mathscr{C}$ circuits of size $2^{\log^k n}$ (otherwise the theorem is trivial). Let δ be the constant in Theorem 5.1. Then by Lemma 3.1, CMD has $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits of complexity $2^{\log^{O(k)} n}$.

The problem of estimating Average-Product for 4 $\mathscr{C}$'s (of size 2^{n^ε}, within error 2^{-n^ε}) is exactly the CAPP problem for $\mathsf{AND}_4 \circ \mathscr{C}$ (of size 2^{n^ε}), thus is solvable in 2^{n-n^ε} time. By Theorem 5.1, NE can certify $n^{\Omega(1)}$ depth hardness. By Theorem 4.2, it follows that NQP cannot be $(1/2 + 2^{-\log^{k+1} n})$-approximated by $\log^{k+1} n$-depth circuits, which contains $2^{\log^k n}$-size $\mathscr{C}$ circuits by our assumption. □

5.1 Proof of Theorem 5.1

To prove Theorem 5.1, we first define the Gap-UNSAT problem.

Gap-UNSAT **with gap** δ**:** Given a circuit C of n inputs, output YES when $C(x) = 0$ for all $x \in \{0,1\}^n$, and NO when C has at least $\delta \cdot 2^n$ satisfying assignments.

We need the following theorem, which is implicit in [Wil13, Wil16].

Theorem 5.2 ([Wil13, Wil16])**.** *Suppose there is a constant $\varepsilon > 0$ such that* Gap-UNSAT *with gap $1 - 1/n^{10}$ for n^ε-depth circuits can be solved in nondeterministic 2^{n-n^ε} time. Then* NE *can certify $n^{\Omega(1)}$ depth.*

We prove a more general version of Theorem 5.2 (Theorem D.3) in Appendix E.1. Theorem 5.2 is simply a direct corollary of Theorem D.3 in which $f(n) = n^\varepsilon$.

It will be convenient to introduce some notation. Let $d_{\mathsf{bin}}(z) = \min_{b\in\{0,1\}} |z - b|$. Intuitively, $d_{\mathsf{bin}}(z)$ measures how close z is to a bit-value. For a function $f : \{0,1\}^n \to \mathbb{R}$, define its closest binary function bin_f as follows: for all $x \in \{0,1\}$, if $f(x) \geq 1/2$, $\mathsf{bin}_f(x) := 1$, otherwise $\mathsf{bin}_f(x) := 0$.

We use the following lemma, adapted from [CW19, Lemma 33], to approximately test whether a linear combination of $\mathscr{C}$ circuits is a valid $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit. Actually, as in [CW19], it suffices to distinguish between linear combinations that are close to Boolean w.r.t. ℓ_∞ norm, and linear combinations that are far from Boolean w.r.t. ℓ_2 norm. Lemma 33 of [CW19] claims to require a Sum-Product algorithm (*i.e.* an *exact* Average-Product algorithm), but their proof also works for algorithms that only *estimates* Average-Product.

Lemma 5.3. *For $S \in \mathbb{N}$, suppose we are given S reals $\{\alpha_i\}_{i\in[S]}$, S $\mathscr{C}$ circuits $\{C_i\}_{i\in[S]}$, and a parameter $\varepsilon < 0.01$. Let $\alpha_{\mathsf{tot}} = \sum_{i\in[S]} |\alpha_i|$ and $\delta = \frac{\varepsilon^2}{2(\alpha_{\mathsf{tot}}+1)^4}$. Suppose the Average-Product of 4 $\mathscr{C}$ circuits on n bits can be estimated within error δ in $T(n)$ time. Let $f = \sum_{i=1}^S \alpha_i \cdot C_i$. There is an algorithm A running in $O(T(n) \cdot (S+1)^4)$ time such that:*

- *If $\|f - \mathsf{bin}_f\|_\infty \leq \varepsilon$, then A always accepts;*
- *if $\|f - \mathsf{bin}_f\|_2 \geq 3\varepsilon$, then A always rejects;*
- *otherwise, A can output anything.*

For completeness, we provide a proof of this lemma in Appendix E.2.

We also need the following transformation on formulas, which is proved by combing PCPP of Formula-Eval and error correcting codes. This generalizes the corresponding transformation for circuits in [CW19] (Theorem 1.15).

Recall that, for a SAT instance F, Y a subset of its variables, and $y \in \{0,1\}^{|Y|}$, we use $F_{Y=y}$ to denote the resulting instance obtained by assigning the Y variables in F to y. (Here we don't remove the already satisfied clauses or the clauses which cannot be satisfied after the partial assignment.) We also use $\mathsf{OPT}(F)$ to denote the maximum fraction of clauses that can be satisfied by any assignment.

Reminder of Theorem 1.16. *Let* $\mathsf{Enc} : \{0,1\}^n \to \{0,1\}^{cn}$ *be the encoder of the error correcting code specified in Theorem 2.12. There is a polynomial-time transformation that, given a formula* D *on* n *inputs of size* $m \geq n$*, outputs a* 2*-SAT instance* F *on variable set* $Y \cup Z$*, where* $|Y| = O(n)$*,* $|Z| \leq \mathrm{poly}(m)$ *and* F *has* $\mathrm{poly}(m)$ *clauses, such that for two constants* $c_{\mathsf{PCPP}} > s_{\mathsf{PCPP}}$*, the following hold for all* $x \in \{0,1\}^n$*.*

- *If* $D(x) = 1$*, then* $\mathsf{OPT}(F_{Y=\mathsf{Enc}(x)}) \geq c_{\mathsf{PCPP}}$*. Furthermore, there is a* $\mathrm{poly}(m)$*-size formula computing a corresponding* $z_D(x)$ *given* x *which satisfies at least* c_{PCPP} *fraction of clauses.*
- *If* $D(x) = 0$*, then* $\mathsf{OPT}(F_{Y=\mathsf{Enc}(x)}) \leq s_{\mathsf{PCPP}}$*.*

Proof. Let δ_1 be the minimum relative distance of the error correcting code specified in Theorem 2.12. Let $\mathsf{Dec} : \{0,1\}^{cn} \to \{0,1\}^n$ be the corresponding decoder.

Let $E(y) = D(\mathsf{Dec}(y))$, then

$$E(\mathsf{Enc}(x)) = 1 \iff D(x) = 1.$$

Recall that Dec can be implemented in NC^1, thus E is a formula of size $m \cdot \mathrm{poly}(n)$.

Consider the 2-query PCPP system $V(E)$ of Formula-Eval with proximity $\delta_{\mathsf{PCPP}} < c_1 \cdot \delta_1$, and let $s_{\mathsf{PCPP}}, c_{\mathsf{PCPP}}$ be its soundness and completeness parameters respectively.

The verifier $V(E)$ consists of $m_{\mathsf{cons}} = 2^{O(\log |E|)} = \mathrm{poly}(m)$ constraints $F_i(E)^{\mathsf{Enc}(x)\circ\pi}$ ($i \in [m_{\mathsf{cons}}]$), and each constraint F_i only depends on 2 bits of $\mathsf{Enc}(x) \circ \pi$. We assume $|\pi| = \ell_{\mathsf{proof}} = \mathrm{poly}(m)$ is the proof length. Lemma 2.14 translates into the following.

- If $D(x) = 1$, then $E(\mathsf{Enc}(x)) = 1$ and there is a proof $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$ (constructible by a $\mathrm{poly}(m)$-size formula with input x) that satisfies at least a c_{PCPP}-fraction of constraints, *i.e.*

$$\Pr_{i \sim [m_{\mathsf{cons}}]}[F_i(E)^{\mathsf{Enc}(x)\circ\pi} = 1] \geq c_{\mathsf{PCPP}};$$

- if $D(x) = 0$, then $E(\mathsf{Enc}(x)) = 0$ and for all proof $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$, at most an s_{PCPP}-fraction of constraints are satisfied, *i.e.*

$$\Pr_{i \sim [m_{\mathsf{cons}}]}[F_i(E)^{\mathsf{Enc}(x)\circ\pi} = 1] \leq s_{\mathsf{PCPP}}.$$

Now, we construct the 2-SAT instance where the variable set is $Y \cup Z$ where $Y = \mathsf{Enc}(x)$ and $Z = \pi$, and the constraints are all the F_i's (note that since each query computes an OR on two bits or their negations, it is indeed a 2-SAT instance). The correctness of our construction follows directly from the above two conditions. □

Recall that the complexity of a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit is the maximum over its size and the sum of absolute values of its coefficients. We now prove Theorem 5.1.

Proof of Theorem 5.1. Let $\varepsilon' = \varepsilon/K$ for a large enough constant $K > 1$, we show that the Gap-UNSAT problem with gap $1 - 1/n^{10}$ for $n^{\varepsilon'}$-depth circuits can be solved in nondeterministic $2^{n-n^{\varepsilon'}}$ time. It follows from Theorem 5.2 that NE can certify $n^{\Omega(1)}$ hardness.

Suppose we are given an $n^{\varepsilon'}$-depth circuit C, and we want to distinguish between the case that C has no satisfying assignments, and that C has $\geq (1 - 1/n^{10}) \cdot 2^n$ satisfying assignments, in nondeterministic $2^{n-n^{\varepsilon'}}$ time. For notation convenience, we first replace C by $\neg C$, and now we need to distinguish between C is a tautology, and C has at most $2^n/n^{10}$ satisfying assignments.

Now we apply Theorem 1.16 to C, which is a formula of $2^{O(n^{\varepsilon'})}$ size, to obtain a 2-SAT instance F with variables set $Y \cup Z$ and $n_{\mathsf{clause}} = 2^{O(n^{\varepsilon'})}$ clauses, where $|Y| = O(n)$ and $|Z| = 2^{O(n^{\varepsilon'})}$. Let $\ell_{\mathsf{proof}} = |Z|$.

Since each output bit of Enc is the parity of a subset of input bits of size at most $n/2$, and $\mathsf{PARITY} \in \oplus\mathsf{L}$, we can in nondeterministic $2^{n/2+o(1)}$ time guess-and-check a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity $2^{\log^{O(1)} n}$ that computes PARITY on $n/2$ bits. Then we obtain, for every output bit of Enc, a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity $2^{\log^{O(1)} n}$ that computes it.

We guess a sequence $T_1, T_2, \ldots, T_{\ell_{\mathsf{proof}}}$, where each T_i is a linear combination of $\mathscr{C}$ circuits of complexity $2^{n^{O(\varepsilon')}}$. If C is always satisfiable, then there are depth-$O(n^{\varepsilon'})$ circuits $\widetilde{T}_1, \widetilde{T}_2, \ldots, \widetilde{T}_{\ell_{\mathsf{proof}}}$ such that for every $x \in \{0,1\}^n$, $\widetilde{\pi}(x) := \widetilde{T}_1(x) \circ \widetilde{T}_2(x) \circ \cdots \circ \widetilde{T}_{\ell_{\mathsf{proof}}}(x)$ satisfies

$$\mathop{\mathbb{E}}_{i \in [n_{\mathsf{clause}}]} [F_i(\mathsf{Enc}(x), \widetilde{\pi}(x))] \geq c_{\mathsf{PCPP}},$$

where F_i is the i-th clause of F.

Since circuits of depth d can be implemented by parity branching programs of size $2^{O(d)}$ [Bar89], each $\widetilde{T}_i$ can be computed by a parity branching program of size $2^{O(n^{\varepsilon'})}$. By Theorem 2.20, for each $\widetilde{T}_i$, there is a projection $p_i : \{0,1\}^n \to \{0,1\}^{2^{O(n^{\varepsilon'})}}$ such that for every $x \in \{0,1\}^n$, $\widetilde{T}_i(x) = \mathsf{CMD}(p_i(x))$. Since CMD has $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits of complexity $2^{\log^{O(1)} n}$, each $\widetilde{T}_i$ has $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits of complexity $2^{n^{O(\varepsilon')}}$.

We are going to show an algorithm that accepts if each T_i we guessed coincides with this circuit for $\widetilde{T}_i$, and always rejects if C has at most $2^n/n^{10}$ satisfying assignments.

We first apply the test in Lemma 5.3 to each T_i. Since the complexity of each T_i is $2^{n^{O(\varepsilon')}}$, it suffices to estimate the Average-Product of 4 $\mathscr{C}$ circuits within error $2^{-n^{O(\varepsilon')}} \geq 2^{-n^{\varepsilon}}$. We reject if any T_i was rejected by the test. For any $1 \leq i \leq \ell_{\mathsf{proof}}$, if $\|T_i - \mathsf{bin}_{T_i}\|_\infty \leq \delta$, then it passes the test. On the other hand, if T_i passes the test, then we have $\|T_i - \mathsf{bin}_{T_i}\|_2 \leq 3\delta$. For each T_i, the test runs in $2^{n-n^{\varepsilon}} \cdot 2^{n^{O(\varepsilon')}}$ time.

For any $x \in \{0,1\}^n$, let $\pi(x) = \mathsf{bin}_{T_1}(x) \circ \mathsf{bin}_{T_2}(x) \circ \cdots \circ \mathsf{bin}_{T_{\ell_{\mathsf{proof}}}}(x)$ be our guessed assignment for Z. Slightly abusing notation, we write $F_i(x) = F_i(\mathsf{Enc}(x), \pi(x))$. For each clause F_i, we can write it as $F_i(x) = P_i(\mathsf{bin}_{C_1}(x), \mathsf{bin}_{C_2}(x))$ where C_1, C_2 are two linear combinations of $\mathscr{C}$ circuits and P_i is a Boolean function over two variables. Furthermore, for each $i \in \{1,2\}$, $\|C_i - \mathsf{bin}_{C_i}\|_2 \leq 3\delta$. Let

$$p_i = \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{U}_n} [F_i(\mathbf{x})], \text{ and } p = \mathop{\mathbb{E}}_{i \sim [n_{\mathsf{clause}}]} p_i.$$

If C is a tautology, then on the correct guess, $p \geq c_{\mathsf{PCPP}}$. If C has at most $2^n/n^{10}$ satisfying assignments, then $p \leq s_{\mathsf{PCPP}} + 1/n^{10}$ for all possible $\pi(x)$. Thus it suffices to estimate each p_i within error $< \gamma = (c_{\mathsf{PCPP}} - s_{\mathsf{PCPP}} - 1/n^{10})/2$.

We write $P_i : \{0,1\}^2 \to \{0,1\}$ as a multi-linear polynomial:

$$P_i(z) = \sum_{S \subseteq [2]} \alpha_S \cdot \prod_{j \in S} z_j,$$

where each $\alpha_S \in [-4,4]$. Therefore it suffices to estimate

$$\mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{U}_n} \left[\prod_{j \in S} \mathsf{bin}_{C_j}(\mathbf{x}) \right] \tag{7}$$

within error $< \gamma/16$.

First we show that

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}\left[\prod_{j\in S} C_j(\mathbf{x})\right]. \tag{8}$$

is a good estimation of (7). When $|S| = 1$, w.l.o.g. assume $S = \{1\}$, we have

$$|(7) - (8)| = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}[\mathsf{bin}_{C_1}(\mathbf{x}) - C_1(\mathbf{x})] \le \|C_1 - \mathsf{bin}_{C_1}\|_1 \le \|C_1 - \mathsf{bin}_{C_1}\|_2 \le 3\delta.$$

Now suppose $|S| = 2$ and $S = \{1,2\}$. Since bin_{C_1} is Boolean, $\|\mathsf{bin}_{C_1}\|_2 \le 1$ and by triangle inequality $\|C_1\|_2 \le 1 + 3\delta$. Similarly $\|\mathsf{bin}_{C_2}\|_2 \le 1$ and $\|C_2\|_2 \le 1 + 3\delta$. By Lemma 2.22 we have

$$|(7) - (8)| = |\langle \mathsf{bin}_{C_1}, \mathsf{bin}_{C_2}\rangle - \langle C_1, C_2\rangle| \le 2\cdot(1+3\delta)\cdot 3\delta.$$

We set δ such that $6\delta(1+3\delta) \le \gamma/32$, so (8) is a good estimate of (7).

Then we estimate (8) within error $< \gamma/32$. W.l.o.g. we assume $|S| = 2$ and $S = \{1,2\}$, *i.e.* we want to estimate

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}[C_1(\mathbf{x})\cdot C_2(\mathbf{x})]. \tag{8}$$

Suppose C_1, C_2 are $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits of sparsity $S_{\mathsf{sparsity}} = 2^{n^{O(\varepsilon')}}$, $C_1(x) = \sum_{i=1}^{S_{\mathsf{sparsity}}} \alpha_i A_i(x)$, and $C_2(x) = \sum_{i=1}^{S_{\mathsf{sparsity}}} \beta_i B_i(x)$. Here A_i, B_i are $\mathscr{C}$ circuits of size $2^{n^{O(\varepsilon')}} \le 2^{n^\varepsilon}$. Then

$$\begin{aligned}(8) &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}\left[\left(\sum_{i=1}^{S_{\mathsf{sparsity}}} \alpha_i A_i(\mathbf{x})\right)\cdot\left(\sum_{j=1}^{S_{\mathsf{sparsity}}} \beta_j B_j(\mathbf{x})\right)\right]\\ &= \sum_{i=1}^{S_{\mathsf{sparsity}}}\sum_{j=1}^{S_{\mathsf{sparsity}}} \alpha_i\beta_j \cdot \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}\left[A_i(\mathbf{x})B_j(\mathbf{x})\right].\end{aligned}$$

Since we can estimate the Average-Product of 2 $\mathscr{C}$ circuits within error 2^{-n^ε}, we can estimate (8) within error

$$2^{-n^\varepsilon}\cdot\left(\sum_{i=1}^{S_{\mathsf{sparsity}}}\sum_{j=1}^{S_{\mathsf{sparsity}}} |\alpha_i|\cdot|\beta_j|\right) \le 2^{-n^\varepsilon}\cdot 2^{n^{O(\varepsilon')}} < \gamma/32.$$

For each $i \le \ell_{\mathsf{proof}} = 2^{O(n^{\varepsilon'})}$, we run a test on T_i in $2^{n-n^\varepsilon}\cdot 2^{n^{O(\varepsilon')}}$ time. For each clause $i \le n_{\mathsf{clause}} = 2^{O(n^{\varepsilon'})}$, we estimate $O(S_{\mathsf{sparsity}}^2)$ instances of Average-Product, in $2^{n-n^\varepsilon}\cdot 2^{n^{O(\varepsilon')}}$ time. It follows that the total time complexity is $2^{n-n^\varepsilon}\cdot 2^{n^{O(\varepsilon')}} \le 2^{n-n^{\varepsilon'}}$. □

5.2 Extensions of Theorem 1.1

We also mention some extensions of Theorem 1.1.

Non-trivial #SAT algorithms imply $\mathsf{Sum}\circ\mathscr{C}$ average-case lower bounds. If we have non-trivial #SAT algorithms for $\mathscr{C}$ circuits (which is stronger than CAPP algorithms), then we obtain average-case lower bounds against $\mathsf{Sum}\circ\mathscr{C}$ circuits, as shown in the following corollary.

Corollary 5.4. *Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. Suppose there is a constant $\varepsilon > 0$ such that the #SAT problem of $\mathsf{AND}_4\circ\mathscr{C}$ circuits of size 2^{n^ε} can be solved in 2^{n-n^ε} time. Then for every constant $k \ge 1$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{Sum}\circ\mathscr{C}$ circuits of $2^{\log^k n}$ size.*

Proof Sketch. Fix $k \geq 1$. We assume DCMD can be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{Sum} \circ \mathscr{C}$ circuits of size $2^{\log^k n}$ (otherwise the corollary is trivial). Then CMD has $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{Sum} \circ \mathscr{C}$ circuits of size $2^{O(\log^k n)}$, which is equivalent to $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits of the same size. (Note that we do *not* have bounds on the sum of absolute values of coefficients of this circuit. Hence CAPP algorithms do *not* work here. But #SAT algorithms will be fine.)

Inspecting the proof of Lemma 5.3, we see that we can test whether a linear combination of $\mathscr{C}$ circuits is close to Boolean with a #SAT algorithm for $\mathsf{AND}_4 \circ \mathscr{C}$, even if there is no constraint on the coefficients of the input linear combination. Inspecting the proof of Theorem 5.1, we can also compute (8) *exactly* with a #SAT algorithm for $\mathsf{AND}_2 \circ \mathscr{C}$, even if there is no constraint on the coefficients. Therefore NE can certify $n^{\Omega(1)}$ depth hardness.

By Theorem 4.2, it follows that NQP cannot be $(1/2 + 2^{-\log^{k+1} n})$-approximated by circuits of depth $\log^{k+1} n$, which contains $\mathsf{Sum} \circ \mathscr{C}$ circuits of $2^{\log^k n}$ size. □

Circuit-analysis algorithms for $\mathsf{OR}_4 \circ \mathscr{C}$ or $\oplus_4 \circ \mathscr{C}$ also imply lower bounds. Let $\mathscr{C}$ be a circuit class. We show that CAPP (or #SAT resp.) algorithms for $\mathsf{OR}_4 \circ \mathscr{C}$ or $\oplus_4 \circ \mathscr{C}$ imply CAPP (or #SAT resp.) algorithms for $\mathsf{AND}_4 \circ \mathscr{C}$. Hence by Theorem 1.1, these algorithms also imply lower bounds.

Lemma 5.5. *Let $\mathscr{C}$ be a circuit class. Suppose there is a* CAPP *(or* #SAT *resp.) algorithm for $s(n)$ size $\mathsf{OR}_4 \circ \mathscr{C}$ circuits running in time $T(n)$. Then there is a* CAPP *(or* #SAT *resp.) algorithm for $(s(n)/16)$ size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits running in time $O(T(n))$. The same holds for $\oplus_4 \circ \mathscr{C}$ in place of $\mathsf{OR}_4 \circ \mathscr{C}$.*

Proof. We write the AND_4 function in the basis of OR functions:

$$\mathsf{AND}_4(z_1, z_2, z_3, z_4) = \sum_{S \subseteq [4]} \alpha_S \cdot \bigvee_{i \in S} z_i, \tag{9}$$

where each α_S is a constant in $[-1, 1]$.

Let $C(x) = \bigwedge_{i=1}^4 C_i(x)$ be an $\mathsf{AND}_4 \circ \mathscr{C}$ circuit of size $s(n)/16$, where each $C_i(x)$ is a $\mathscr{C}$ circuit. Then

$$\mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{U}_n}[C(\mathbf{x})] = \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{U}_n}\left[\bigwedge_{i=1}^4 C_i(\mathbf{x})\right] = \sum_{S \subseteq [4]} \alpha_S \cdot \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{U}_n}\left[\bigvee_{i \in S} C_i(\mathbf{x})\right].$$

We enumerate $S \subseteq [4]$. For #SAT algorithms, it suffices to compute

$$\mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{U}_n}\left[\bigvee_{i \in S} C_i(\mathbf{x})\right] \tag{10}$$

exactly, which can be done by the #SAT algorithm for $\mathsf{OR}_4 \circ \mathscr{C}$. For CAPP algorithms, it suffices to estimate (10) within error $(16/s(n))/16 = 1/s(n)$, which can be done by the CAPP algorithm for $\mathsf{OR}_4 \circ \mathscr{C}$.

The case that we have circuit-analysis algorithms for $\oplus_4 \circ \mathscr{C}$ is essentially the same, except that we write AND_4 in the basis of XOR functions instead of (9):

$$\mathsf{AND}_4(z_1, z_2, z_3, z_4) = \sum_{S \subseteq [4]} \alpha'_S \cdot \bigoplus_{i \in S} z_i,$$

and use the CAPP or #SAT algorithm for $\oplus_4 \circ \mathscr{C}$ instead. □

Nondeterministic algorithms also imply lower bounds. It is easy to see from the proofs of Lemma 5.3 and Theorem 5.1 that, Theorem 5.1 holds even when the corresponding CAPP or #SAT algorithms are *nondeterministic*. We state without proof the following corollary:

Corollary 5.6. *Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. Then:*

- *Suppose there is a constant $\varepsilon > 0$ such that the* CAPP *problem of* $\mathsf{AND}_4 \circ \mathscr{C}$ *circuits of size* 2^{n^ε} *can be solved in* nondeterministic 2^{n-n^ε} *time. Then for every constant $k \geq 1$,* NQP *cannot be* $(1/2 + 2^{-\log^k n})$*-approximated by $\mathscr{C}$ circuits of* $2^{\log^k n}$ *size.*
- *Suppose there is a constant $\varepsilon > 0$ such that the* #SAT *problem of* $\mathsf{AND}_4 \circ \mathscr{C}$ *circuits of size* 2^{n^ε} *can be solved in* nondeterministic 2^{n-n^ε} *time. Then for every constant $k \geq 1$,* NQP *cannot be* $(1/2 + 2^{-\log^k n})$*-approximated by* $\mathsf{Sum} \circ \mathscr{C}$ *circuits of* $2^{\log^k n}$ *size.*

6 Non-trivial CAPP Algorithms Imply Nondeterministic PRGs

In this section, we show that to construct nondeterministic infinitely often PRGs fooling a circuit class $\mathscr{C}$, it suffices to have nontrivial CAPP algorithms for a related class $\mathscr{C}'$. Recall that Junta_k is the family of k-juntas, *i.e.* functions that only depend on k input bits. If there is a nontrivial CAPP algorithm for $\mathscr{C}' = \mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_{\log n}$, then we can construct nondeterministic infinitely often PRGs fooling polynomial-size $\mathscr{C}$ circuits.

Theorem 6.1. *Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. Suppose there is a constant $\varepsilon > 0$ such that the* CAPP *problem of* $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_{\log n}$ *circuits of size* 2^{n^ε} *can be solved in* 2^{n-n^ε} *time. For every $k \geq 1$ and $\delta > 0$:*

1. *there is an i.o. NPRG with seed length* $2^{\log^\delta n}$ *that* $(1/n^k)$*-fools $\mathscr{C}$ circuits of n^k size, and*
2. *there is an* $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$*-computable PRG with seed length* $2^{\log^\delta n}$ *that* $(1/n^k)$*-fools $\mathscr{C}$ circuits of n^k size.*

In Section 6.2 and Section 6.3, we prove generalized versions of the i.o. NPRG and the $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG provided in Theorem 6.1. We discuss these generalizations below.

Seed length. The seed lengths in Theorem 6.1 can be shorten to other less "natural" functions. We choose $2^{\log^\delta n}$ in Theorem 6.1 only for ease of presentation.

- Let $f : \mathbb{N} \to \mathbb{N}$ be a good resource function. If for every constant $k \geq 1$, $f(f(n^k)^k) = 2^{n^{o(1)}}$, then we say f is **sub-half-exponential**. Actually, the seed length of the i.o. NPRG in Item 1 of Theorem 6.1 can be shortened to be polynomially related to *the inverse of a sub-half-exponential function*. It is easy to see that $f(n) = 2^{\log^k n}$ is sub-half-exponential for any constant $k \geq 1$.
- Let $f : \mathbb{N} \to \mathbb{N}$ be a good resource function. If for every constant $k \geq 1$, $f(f(f(f(n^k)^k)^k)^k) = 2^{n^{o(1)}}$, then we say f is **sub-fourth-exponential**. The seed length of the $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG in Item 2 of Theorem 6.1 can be shortened to be polynomially related to *the inverse of a sub-fourth-exponential function*. It is also easy to see that $f(n) = 2^{\log^k n}$ is sub-fourth-exponential for any constant $k \geq 1$.

One can also compare our i.o. NPRG with our $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG. Ignoring the advice bit, the requirements of $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG is stronger than that of i.o. NPRG. (The former construction computes *a single* PRG over every nondeterministic branch, but the latter does not necessarily compute the same PRG over its nondeterministic branches. See Definition 2.7 and Definition 2.8). However, our i.o. NPRG has a smaller seed length (inverse-sub-half-exponential) than that of $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG (inverse-sub-fourth-exponential). We believe that the seed length of our i.o. NPRG can be shortened to polylog(n), which we leave as an open problem.

Juntas. Note that we need CAPP algorithms for $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_{\log n}$ in order to construct PRGs fooling $\mathscr{C}$. The reason is that we use Nisan-Wigderson PRG [NW94]: Given an oracle $\mathcal{O}$ that breaks the PRG, [NW94] shows that there is a circuit of the form $\mathcal{O} \circ \mathsf{Junta}_{\log n}$ that weakly approximates the underlying hard function.

For some circuit classes $\mathscr{C}$, we only know how to do circuit-analysis for $\mathscr{C} \circ \mathsf{Junta}_a$, where $a = o(\log n)$ is a parameter. In this case we can also construct i.o. NPRGs and $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRGs, but with a larger seed length. We refer to Theorem 6.5 and Theorem 6.7 for more details.

6.1 Preliminaries

We need the following generalization of Theorem 5.1. A proof can be found in Appendix D.2.

Lemma 6.2. *Let $\mathscr{C}$ be a typical circuit class. Let $S_{\mathsf{CMD}}, S_{\mathsf{cert}} : \mathbb{N} \to \mathbb{N}$ be good resource functions such that $S_{\mathsf{CMD}}(n) \leq 2^{0.1n}$, and $S_{\mathsf{cert}}(n) = n^{\omega(1)}$. There are universal constants $\delta > 0$ and $K > 4$ such that, if the following hold:*

- CMD *has* $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ *circuits of complexity* $S_{\mathsf{CMD}}(n)$*, and*
- *the Average-Product for* 4 $\mathscr{C}$ *circuits of size* $S(n) = S_{\mathsf{CMD}}(S_{\mathsf{cert}}(n)^K)^K$ *can be estimated in* $T(n) = 2^n/S(n)$ *time, within error* $1/S(n)$*.*

Then NE *can certify* $\log S_{\mathsf{cert}}(n)$ *depth hardness.*

The following lemma concerns the best circuit lower bounds for NE provable by our approaches. To prove NE cannot be $(1/2 + 1/S(n))$-approximated by $\mathscr{C}$ circuits of $S(n)$ size, we need $S(n)$ to satisfy some technical properties. We say a function $S(n)$ that satisfies those properties is *nice*. As the definition of nice functions is very technical, we don't bother to formally define it here. We refer the reader to Definition D.5.

We remark that all "natural" functions $S(n)$ that is super-polynomial and sub-fourth-exponential are nice, *e.g.* $S(n) = 2^{\log^k n}$ and $S(n) = 2^{\log^{k \log\log n} n}$.

Lemma 6.3. *Let $S : \mathbb{N} \to \mathbb{N}$ be a nice function. If for every constant $d \geq 1$ and $S_{\mathsf{CAPP}}(n) = S(S(S(S(n^2)^d)^d)^d)^d$, the* CAPP *problem for $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $S_{\mathsf{CAPP}}(n)$ can be solved in $2^n/S_{\mathsf{CAPP}}(n)$ time, then $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$ cannot be $(1/2 + 1/S(n))$-approximated by $\mathscr{C}$ circuits of size $S(n)$.*

We use the standard construction of Nisan-Wigderson PRG. As discussed above, it is important that given a $\mathscr{C}$ circuit that breaks the NW PRG based on some hard function, the complexity of approximating that hard function is $\mathscr{C} \circ \mathsf{Junta}_a$, where a is some parameter. Therefore we stress that the hard function required by the NW PRG needs to be hard to approximate by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits.

Lemma 6.4 ([NW94]). *Let m, ℓ, a be integers such that $a \leq \ell$, and $t = O(\ell^2 \cdot m^{1/a}/a)$. Let $\mathscr{C}$ be a circuit class closed under negation. There is a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that the following hold. For any function $Y : \{0,1\}^\ell \to \{0,1\}$ represented as a length-2^ℓ truth table, if Y cannot be $(1/2 + \varepsilon/m)$-approximated by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits (where the top $\mathscr{C}$ circuit has size S), then $G(Y, \mathcal{U}_t)$ ε-fools every $\mathscr{C}$ circuit (of size S). That is, for any $\mathscr{C}$ circuit C (of size S),*

$$\left| \Pr_{\mathbf{s} \sim \mathcal{U}_t} [C(G(Y, \mathbf{s})) = 1] - \Pr_{\mathbf{x} \sim \mathcal{U}_m} [C(\mathbf{x}) = 1] \right| \leq \varepsilon.$$

Moreover, the function G is computable in $\mathrm{poly}(m, 2^t)$ time.

We include a proof of Lemma 6.4 in Appendix E.3, showing that the construction only puts juntas at the bottom.

6.2 An i.o. NPRG

In this section, we construct an i.o. NPRG with inverse-sub-half-exponential seed length.

Theorem 6.5. *There is an absolute constant $d = 8$ such that the following hold. Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. Let $a = a(n) \leq \log n$ be a parameter, and $f : \mathbb{N} \to \mathbb{N}$ be a good resource function.*

Suppose that for every constant $k \geq 1$, the CAPP problem of $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_a$ circuits of size $f(f(n)^k)^k$ can be solved in $2^n / f(f(n)^k)^k$ time. Then at least one of the following holds.

- *For every $k \geq 1$, there is an E-computable i.o. PRG with seed length $O\big(f^{-1}(n^k)^d \cdot n^{1/a}/a\big)$, that $(1/n^k)$-fools $\mathscr{C}$ circuits of n^k size.*
- *For every $k \geq 1$, there is an i.o. NPRG with seed length $O\big(f^{-1}(n^k)^d\big)$, that $(1/n^k)$-fools (general) circuits of $k \log n$ depth.*

Proof. We divide the argument into two cases.

Case I. Suppose that for infinitely many n's, DCMD on input length $n^3(n+1)/2$ cannot be $(1/2 + 1/f(n)^2)$-approximated by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits, where the size of the top $\mathscr{C}$ circuit is $f(n)$. Let $m = n$, $s = f^{-1}(n^k) + 1$, $\ell_{\mathsf{PRG}} = s^3(s+1)/2$, $\varepsilon_{\mathsf{adv}} = m/f(s)^2$ (*i.e.* $\varepsilon_{\mathsf{adv}}/m = 1/f(s)^2$) and Y be the function $\mathsf{DCMD}_{\ell_{\mathsf{PRG}}}$. Since $a(n) \leq \log n$ and $f(n) \leq 2^n$, we have $f(a) \leq n^k$, therefore $\ell_{\mathsf{PRG}} > f^{-1}(n^k) \geq a$. By Lemma 6.4, there is an i.o. PRG with seed length

$$O(\ell_{\mathsf{PRG}}^2 \cdot n^{1/a}/a) = O\Big((f^{-1}(n^k))^8 \cdot n^{1/a}/a\Big),$$

that $\varepsilon_{\mathsf{adv}}$-fools $\mathscr{C}$ circuits of $f(s) \geq n^k$ size, where

$$\varepsilon_{\mathsf{adv}} = n/f(s)^2 \leq 1/n^k.$$

Since $\mathsf{DCMD} \in \mathsf{P}$, this PRG is E-computable.

Case II. Suppose that for all large enough n, DCMD on length $n^3(n+1)/2$ can be $(1/2+1/f(n)^2)$-approximated by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits, where the size of the top $\mathscr{C}$ circuit is $\leq f(n)$. By Lemma 3.1, CMD has $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C} \circ \mathsf{Junta}_a$ circuits of complexity $S_{\mathsf{CMD}}(n) = \mathrm{poly}(f(n))$.

Let δ and K be the universal constants in Lemma 6.2, and c be the universal constant c in Theorem 4.4. Let $S_{\mathsf{cert}}(n) = f(n)^{1/c}$ and $S(n) = S_{\mathsf{CMD}}(S_{\mathsf{cert}}(n)^K)^K$. Then $S(n) \leq f(f(n)^C)^C$ for some constant C. The problem of estimating Average-Product for 4 $\mathscr{C} \circ \mathsf{Junta}_a$ circuits (of size $S(n)$, within error $1/S(n)$) is equivalent to the CAPP problem for $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_a$ (of size $S(n)$), thus is solvable in $2^n/S(n)$ time. By Lemma 6.2, NE can certify $\log S_{\mathsf{cert}}(n)$ depth hardness.

Let $V(\cdot,\cdot)$ be the NE verifier in Definition 4.1 that certifies $\log S_{\mathsf{cert}}(n)$ depth hardness. Let $\ell_{\mathsf{PRG}} = f^{-1}(n^k)+1$ and $S_{\mathsf{out}} = S_{\mathsf{cert}}(\ell_{\mathsf{PRG}})^c \geq n^k$. By Theorem 4.4, there is an i.o. NPRG with seed length

$$s_{\mathsf{seed}} \leq O(\ell_{\mathsf{PRG}}^2) = O\Big(f^{-1}(n^k)^2\Big),$$

that $\varepsilon'_{\mathsf{adv}}$-fools circuits of depth $\log(S_{\mathsf{out}}) \geq k \log n$, where

$$\varepsilon'_{\mathsf{adv}} = 1/S_{\mathsf{out}} \leq 1/n^k. \qquad \square$$

The two PRGs in Theorem 6.5 imply an i.o. NPRG fooling $\mathscr{C}$ circuits:

Corollary 6.6. *Let circuit class $\mathscr{C}$, functions a, f and constant d be as in Theorem 6.5. For every $k \geq 1$, there is an i.o. NPRG with seed length $O\big((f^{-1}(n^k))^d \cdot n^{1/a}/a\big)$ that $(1/n^k)$-fools $\mathscr{C}$ circuits of n^k size.*

It is easy to see that the following extensions hold:

- The conclusion of Theorem 6.5 still holds even if the CAPP algorithm is *nondeterministic*.
- If there is a (possibly nondeterministic) #SAT algorithm for $\mathscr{C}$ with the same time bounds, then the i.o. NPRG constructed in Theorem 6.5 also fools $\mathsf{Sum} \circ \mathscr{C}$ circuits of the same size with the same advantage.[26]

6.3 An $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG

If one insists that each nondeterministic branch should output the *same* PRG (*i.e.* as in Definition 2.7), then we can still achieve a seed length of inverse-sub-fourth-exponential.

Theorem 6.7. *Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by circuits of depth $O(\log S)$. Let $a = a(n) \leq \log n$ be a parameter, and $f : \mathbb{N} \to \mathbb{N}$ be a nice function.*

Suppose that for every constant $k \geq 1$, the CAPP problem of $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_a$ circuits of size $f(f(f(f(n^2)^k)^k)^k)^k$ can be solved in $2^n/f(f(f(f(n^2)^k)^k)^k)^k$ time. Then for every constant $k \geq 1$, there is an $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable i.o. PRG with seed length $O\big(f^{-1}(n^k)^2 \cdot n^{1/a}/a\big)$ that $(1/n^k)$-fools $\mathscr{C}$ circuits of n^k size.

Proof. Let $S(n) = f(n)^2$. By Definition D.5, $S(n)$ is also a nice function. For every $d \geq 1$, let $S_{\mathsf{CAPP}}(n) = S(S(S(S(n^2)^d)^d)^d)^d$, then the CAPP problem for $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_a$ circuits of size $S_{\mathsf{CAPP}}(n)$ can be solved in $2^n/S_{\mathsf{CAPP}}(n)$ time. By Lemma 6.3, there is a language $L \in (\mathsf{NE} \cap \mathsf{coNE})_{/1}$ that cannot be $(1/2+1/S(n))$-approximated by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits of $S(n)$ size.

[26] If DCMD is not well-approximated by $\mathsf{Sum} \circ \mathscr{C}$ circuits, we directly use Nisan-Wigderson PRG. Otherwise, we can still certify depth hardness, construct an i.o. NPRG fooling low depth circuits, which contains $\mathsf{Sum} \circ \mathscr{C}$ circuits.

Let $\ell = \ell_{\mathsf{PRG}} = f^{-1}(n^k) + 1$, $m = n$ and the function Y be $L_{\ell_{\mathsf{PRG}}}$. By Lemma 6.4, there is a PRG $G(Y, \cdot)$ with seed length

$$s_{\mathsf{seed}} = O\Big(\ell_{\mathsf{PRG}}^2 \cdot n^{1/a}/a\Big) = O\Big(f^{-1}(n^k)^2 \cdot n^{1/a}/a\Big),$$

that $\varepsilon_{\mathsf{adv}}$-fools $\mathscr{C}$ circuits of size $S(\ell_{\mathsf{PRG}}) \geq n^{2k}$, where

$$\varepsilon_{\mathsf{adv}} = n/S(\ell_{\mathsf{PRG}}) \leq 1/n^k.$$

It remains to argue that $G(Y, \cdot)$ is $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable. Let $t \in \{0,1\}^{s_{\mathsf{seed}}}$ be the random seed that the PRG receives as an input. We first compute the value of ℓ_{PRG} from the value of $|t|$. Then we receive the advice bit for $L_{\ell_{\mathsf{PRG}}}$, and guess the truth table of $L_{\ell_{\mathsf{PRG}}}$. In particular, for every $x \in \{0,1\}^{\ell_{\mathsf{PRG}}}$ we guess the value of $L(x)$. Then for each $x \in \{0,1\}^{\ell_{\mathsf{PRG}}}$, we use the $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$ algorithm for L to verify that the x-th entry of the truth table is correct. After that, we use Lemma 6.4 to output $G(L_{\ell_{\mathsf{PRG}}}, t)$ in $\mathrm{poly}(n, 2^{s_{\mathsf{seed}}})$ time. It is easy to verify that the whole algorithm runs in (nondeterministic) $2^{O(s_{\mathsf{seed}})}$ time. □

It is easy to see that the following extensions hold:

- The conclusion of Theorem 6.7 still holds even if the CAPP algorithm is *nondeterministic*.
- If there is a (possibly nondeterministic) #SAT algorithm for $\mathscr{C}$ with the same time bounds, then the PRG constructed in Theorem 6.7 also fools $\mathsf{Sum} \circ \mathscr{C}$ circuits of the same size, and with the same advantage.

7 Applications

In this section we discuss various applications of our results in Section 5 and Section 6.

7.1 Strong Average-case Lower Bounds for $\mathsf{ACC}^0 \circ \mathsf{THR}$

We first apply the non-trivial #SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$ of [Wil14a] to prove our average-case lower bound against ACC^0. In particular, [Wil14a] showed that for every constants $d_\star \geq 1, m_\star \geq 2$, there is a constant $\varepsilon > 0$ such that the number of satisfying assignments for an $\mathsf{AC}^0_{d_\star}[m_\star] \circ \mathsf{THR}$ circuit of size 2^{n^ε} can be computed in 2^{n-n^ε} time.

Reminder of Theorem 1.5. *For every constant $k > 0$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$. Consequently, NQP cannot be computed by $\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$ (in the worst-case), for all $k \geq 1$.*

The same holds for $(\mathsf{N}\cap\mathsf{coN})\mathsf{QP}_{/1}$ in place of NQP.

Proof of Theorem 1.5. We assume DCMD can be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$ (otherwise the theorem is trivial). In particular, this means there are constants $d_\star, m_\star$ such that DCMD can be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{AC}^0_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$. By Lemma 3.1, CMD has $\widetilde{\mathsf{Sum}}_\delta \circ \mathsf{AC}^0_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits of size $2^{\log^{O(k)} n}$, where δ is the constant in Theorem 5.1.

By Theorem 5.1, and applying the #SAT algorithm for $\mathsf{AC}^0_{d_\star}[m_\star] \circ \mathsf{THR}$, it follows that NE can certify $n^{\Omega(1)}$-depth hardness. By Theorem 4.2, NQP $((\mathsf{N}\cap\mathsf{coN})\mathsf{QP}_{/1})$ cannot be $(1/2 + 2^{-\log^{k+1} n})$-approximated by $\log^{k+1} n$-depth circuits, which contains $2^{\log^k n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. □

As the algorithm is indeed for #SAT, the lower bounds also hold for $\mathsf{Sum} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

Reminder of Corollary 1.9. *For every constant $k \geq 1$, NQP cannot be $(1/2 + 2^{-\log^k n})$-approximated by $\mathsf{Sum} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$. Consequently, NQP cannot be computed by $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of size $2^{\log^k n}$ (in the worst-case), for all $k \geq 1$.*

The same holds for $(\mathsf{N} \cap \mathsf{coN})\mathsf{QP}_{/1}$ in place of NQP.

7.2 Nondeterministic Infinitely Often PRG for $\mathsf{AC}^0_{d_\star}[m_\star]$ Circuits

Now we construct PRGs for polynomial-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits for every constant $d_\star \geq 1, m_\star \geq 2$. Recall that a function $f : \mathbb{N} \to \mathbb{N}$ is sub-half-exponential if for every constant $k \geq 1$, $f(f(n^k)^k) = 2^{n^{o(1)}}$, and f is sub-fourth-exponential if for every constant $k \geq 1$, $f(f(f(f(n^k)^k)^k)^k) = 2^{n^{o(1)}}$. We also call a function satisfying some technical conditions *nice*, see Definition D.5 for details.

Fix some $d_\star \geq 1, m_\star \geq 2$. Then there is a constant $d'_\star = d_\star + O(1)$ such that

$$\mathsf{AND}_4 \circ \mathsf{AC}^0_{d_\star}[m_\star] \circ \mathsf{Junta}_{\log n} \subseteq \mathsf{AC}^0_{d'_\star}[m_\star].$$

Recall that there is a constant $\varepsilon > 0$ such that the number of satisfying assignments for an $\mathsf{AC}^0_{d'_\star}[m_\star]$ circuit of size 2^{n^ε} can be computed in 2^{n-n^ε} time [Wil14a]. We have the following corollaries of Theorem 6.5 and Theorem 6.7:

Theorem 7.1 (An i.o. NPRG for $\mathsf{AC}^0_{d_\star}[m_\star]$ Circuits)**.** *Let $d_\star \geq 1, m_\star \geq 2$ be constants, $f : \mathbb{N} \to \mathbb{N}$ be a good resource function that is sub-half-exponential, and d be the absolute constant in Theorem 6.5. Then at least one of the following holds.*

- *For every $k \geq 1$, there is an E-computable i.o. PRG with seed length $O\big((f^{-1}(n^k))^d\big)$ that $(1/n^k)$-fools $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits of n^k size.*
- *For every $k \geq 1$, there is an i.o. NPRG with seed length $O\big((f^{-1}(n^k))^d\big)$ that $(1/n^k)$-fools circuits of $k \log n$ depth.*

As a consequence, for every $k \geq 1$, there is an i.o. NPRG with seed length $O\big((f^{-1}(n^k))^d\big)$ that $(1/n^k)$-fools $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits of n^k size.

Theorem 7.2 (An $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable PRG for $\mathsf{AC}^0_{d_\star}[m_\star]$ Circuits)**.** *Let $d_\star \geq 1, m_\star \geq 2$ be constants, $f : \mathbb{N} \to \mathbb{N}$ be any nice function that is sub-fourth-exponential. For every constant $k \geq 1$, there is an $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable i.o. PRG with seed length $O\big((f^{-1}(n^k))^2\big)$ that $(1/n^k)$-fools $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits of n^k size.*

For comparison, the previous best $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable i.o. PRG for polynomial-size $\mathsf{AC}^0[6]$ circuits needs $n(1 - o(1))$ seed length [COS18].

Remark 7.3. Theorem 7.1 and Theorem 7.2 only show that for every *fixed* $d_\star \geq 1, m_\star \geq 2$, we can construct PRGs fooling $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits. However, looking inside the proofs, we can actually construct *a single* PRG that fools $\mathsf{AC}^0_d[m]$ circuits for every $d \geq 1, m \geq 2$ *simultaneously*, with the same seed length and complexity.

- For Theorem 7.1, we divide the argument into two cases. If DCMD cannot be $(1/2 + 1/f(n)^2)$-approximated by $\mathsf{ACC}^0 = \bigcup_{d,m} \mathsf{AC}^0_d[m]$ circuits of size $f(n)$, then we construct an E-computable

i.o. PRG fooling ACC^0 circuits as in Case I of Theorem 6.5. Otherwise, DCMD can be approximated by $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits *for some fixed* $d_\star, m_\star$, we use the #SAT algorithm for $\mathsf{AC}^0_{d_\star}[m_\star]$ and proceed as in Case II of Theorem 6.5. In this case, the i.o. NPRG we constructed actually fools NC^1 circuits, which contains ACC^0 circuits.

- For Theorem 7.2, we prove an average-case lower bound for ACC^0 using the same argument as the proof of Theorem 1.5, and plug it into the Nisan-Wigderson construction.

7.3 Lower Bounds and PRGs for $\mathsf{Sum} \circ \mathsf{PTF}$ Circuits

Recall that a *polynomial threshold function* (PTF) of degree k is a Boolean function $f : \{0,1\}^n \to \{0,1\}$, such that there is a degree-k polynomial $p(x_1, \ldots, x_n) \in \mathbb{R}[x_1, \ldots, x_n]$, such that for every $x \in \{0,1\}^n$,

$$f(x) = 0 \iff p(x_1, \ldots, x_n) \geq 0.$$

We denote PTF_k as the class of functions computed by degree-k PTFs. Clearly, THR is just PTF_1.

Note that every PTF of degree k can be sign-represented by an integer polynomial with coefficients $2^{\widetilde{O}(n^k)}$ [Mur71].

In this section, we prove lower bounds and construct (nondeterministic) PRGs for $\mathsf{Sum} \circ \mathsf{PTF}$ circuits, *i.e. linear combination* of PTF gates. We remark that [KKL17] proved lower bounds for PTF *circuits* (of constant depth and slightly superlinear wire complexity), and [Kan12, MZ13, KL18] constructed PRG fooling PTF *gates*. However, our results for $\mathsf{Sum} \circ \mathsf{PTF}$ circuits are novel. In particular, even for fooling $\mathsf{Sum} \circ \mathsf{THR}$ circuits, no nontrivial PRG was known before our results.

We first discuss the special case $\mathsf{Sum} \circ \mathsf{THR}$, as we can prove the stronger NP lower bounds for them. Since the #SAT problem for $\mathsf{AND}_4 \circ \mathsf{THR}$ circuits can be computed in $2^{n/2}\mathrm{poly}(n)$ time [Wil18, HS74], by Theorem 1.1 (its first extension in Section 5.2) we have:

Reminder of Theorem 1.10. *For all constants k, NP cannot be $(1/2 + 1/n^k)$-approximated by n^k-size $\mathsf{Sum} \circ \mathsf{THR}$ circuits. Consequently, NP cannot be computed by n^k-size $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{THR}$ circuits for all constants k.*

To prove our results for $\mathsf{Sum} \circ \mathsf{PTF}$ circuits, we use the following #SAT algorithm for $\mathsf{AND}_4 \circ \mathsf{PTF}$.

Theorem 7.4 ([BKK+19])**.** *For every parameter $k = k(n)$, there is a ZPP algorithm that counts the number of satisfying assignments of any $\mathsf{AND}_4 \circ \mathsf{PTF}_k$ circuit over n variables in $2^{n-m} \cdot \mathrm{poly}(n)$ time, where $m = n^{1/(k+1)}/\log n$.*

Remark 7.5. [BKK+19, Section 3] only presented a #SAT algorithm for *a single* PTF *gate*, but it is easy to extend their algorithm to *the* AND *of four* PTF *gates*. For completeness, we include a proof sketch of Theorem 7.4 in Appendix E.4.

Therefore, the following hold.

- For every constant $d \geq 1$, there is a constant $\varepsilon > 0$ such that the #SAT problem for $\mathsf{AND}_4 \circ \mathsf{PTF}_d$ can be solved in 2^{n-n^ε} time by a ZPP algorithm.
- Let $d(n) = 0.49\frac{\log n}{\log\log n}$, then $n^{1/(d(n)+1)}/\log n = \omega(\log n)$. Thus the #SAT problem for $\mathsf{AND}_4 \circ \mathsf{PTF}_{d(n)}$ can be solved in $2^n/n^{\omega(1)}$ time by a ZPP algorithm.

As ZPP algorithms are also *nondeterministic* algorithms, the following lower bounds follow from Theorem 1.1:

Reminder of Theorem 1.11. *The following hold.*

- *For every constant $d, k \geq 1$,* NQP *cannot be* $(1/2 + 2^{-\log^k n})$*-approximated by* $\mathsf{Sum} \circ \mathsf{PTF}_d$ *circuits of sparsity* $2^{\log^k n}$*. Consequently,* NQP *does not have* $2^{\log^k n}$*-size* $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{PTF}_d$ *circuits.*
- *Let* $d(n) = 0.49\frac{\log n}{\log\log n}$*, then* NE *cannot be* $(1/2 + 1/\mathrm{poly}(n))$*-approximated by* $\mathsf{Sum} \circ \mathsf{PTF}_{d(n)}$ *circuits of sparsity* $\mathrm{poly}(n)$*. Consequently,* $\mathsf{NE} \not\subset \mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{PTF}_{d(n)}$*.*

Since $\mathsf{PTF}_k \circ \mathsf{Junta}_a \subseteq \mathsf{PTF}_{k \cdot a}$, we can also use Theorem 6.7 to construct PRGs fooling $\mathsf{Sum} \circ \mathsf{PTF}$ circuits. We prove that there is an $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable i.o. PRG fooling $\mathsf{Sum} \circ \mathsf{PTF}_{O(1)}$, that only requires $2^{\log^\varepsilon n}$ seed length, for any constant $\varepsilon > 0$.

Reminder of Theorem 1.12. *For every constants $d, k \geq 1$ and $\varepsilon > 0$, there is an $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable i.o. PRG with seed length $O(2^{\log^\varepsilon n})$ that $(1/n^k)$-fools $\mathsf{Sum} \circ \mathsf{PTF}_d$ circuits of sparsity n^k.*

Proof. Let $a = \Theta(\log n / \log^{\varepsilon/2} n)$, then $\mathsf{PTF}_d \circ \mathsf{Junta}_a \subseteq \mathsf{PTF}_{a \cdot d}$. Let

$$m = n^{1/(ad+1)} / \log n \geq \Omega(2^{\log^{\varepsilon/3} n}).$$

By Theorem 7.4, the #SAT algorithm for $\mathsf{AND}_4 \circ \mathsf{PTF}_{a \cdot d}$ can be solved in nondeterministic 2^{n-m} time.

Let $f(n) = 2^{\log^{4/\varepsilon} n}$, then $f(n)$ is a nice function. For every constant $k \geq 1$, $f(f(f(f(n^2)^k)^k)^k)^k = 2^{\log^{O(1)} n} \leq 2^m$. Therefore, by Theorem 6.7, there is an $(\mathsf{NE} \cap \mathsf{coNE})_{/1}$-computable i.o. PRG with seed length

$$O\Big((2^{\log^{\varepsilon/4} n})^2 \cdot n^{1/a}/a\Big) = O(2^{\log^\varepsilon n}),$$

that $(1/n^k)$-fools $\mathsf{Sum} \circ \mathsf{PTF}_k$ circuits of sparsity n^k. □

7.4 Towards TC^0_3 Lower Bounds

Finally, we show that non-trivial derandomization of $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits implies lower bounds for depth-3 TC circuits.

Reminder of Theorem 1.13. *If there is a $2^n/n^{\omega(1)}$ time* CAPP *algorithm for* $\mathrm{poly}(n)$*-size* $\mathsf{MAJ} \circ \mathsf{MAJ}$ *circuits. Then* $\mathsf{NEXP} \not\subset \mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$*.*

Proof. Suppose there is a $2^n/n^{\omega(1)}$ time CAPP algorithm for $\mathrm{poly}(n)$ size $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits. Since $\oplus_4 \circ \mathsf{MAJ} \circ \mathsf{MAJ} \subseteq \mathsf{MAJ} \circ \mathsf{MAJ}$,[27] there is a $2^n/n^{\omega(1)}$ time CAPP algorithm for $\mathrm{poly}(n)$ size $\oplus_4 \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ circuits. The theorem follows from Lemma 5.5 and Theorem 1.1. □

[27] [CW19, Lemma 50] proved $\oplus_4 \circ \mathsf{THR} \circ \mathsf{THR} \subseteq \mathsf{THR} \circ \mathsf{THR}$, and the proof can be adapted to $\mathsf{MAJ} \circ \mathsf{MAJ}$.

8 Open Problems

We conclude with several interesting open problems stemming from our work.

1. The most exciting open question would be to apply Theorem 1.13 to prove super-polynomial lower bounds for TC^0_3.

2. Are there P-complete problems with similar random-reducibility properties of CMD and DCMD? Besides being an interesting problem in its own right, the existence of such a problem would greatly simplify our framework for strong average-case lower bounds. In particular, we will no longer need hard MA problems with *low depth* predicates, and PCPP with *low depth* computable proofs.

3. The seed length of our i.o. NPRG fooling ACC^0 circuits is only inverse sub-half-exponential. Can we obtain an i.o. NPRG with $\mathrm{polylog}(n)$ seed length? As a related question, can we show that there is a constant $\varepsilon > 0$ such that E^{NP} cannot be $(1/2 + 1/2^{n^\varepsilon})$-approximated by ACC^0 circuits of 2^{n^ε} size? (This paper only implicitly proves that E^{NP} cannot be $(1/2 + 1/f(n))$-approximated by ACC^0 circuits of $f(n)$ size for sub-half-exponential $f(n)$.)

4. Since we have proved lower bounds for $\mathsf{MAJ} \circ \mathsf{ACC}^0$, the natural next step would be to prove lower bounds for $\mathsf{THR} \circ \mathsf{ACC}^0$. Can we formulate any *algorithmic approach* to prove such a lower bound? That is, are there certain non-trivial circuit-analysis algorithms for $\mathscr{C}$ which would imply $\mathsf{THR} \circ \mathscr{C}$ lower bounds?

   It seems plausible to us that non-trivial #SAT algorithms would suffice (note that that we already proved non-trivial #SAT algorithms for $\mathscr{C}$ imply $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathscr{C}$ lower bounds, which is a non-trivial sub-class of $\mathsf{THR} \circ \mathscr{C}$). Such a connection would also imply lower bounds for $\mathsf{THR} \circ \mathsf{ACC}^0 \circ \mathsf{THR}$, which is (much) stronger than the already notorious circuit class $\mathsf{THR} \circ \mathsf{THR}$.

5. Is THR contained in $\mathsf{MAJ} \circ \mathsf{ACC}^0$? (Or even $\mathsf{MAJ} \circ \mathsf{Sum} \circ \mathsf{ACC}^0$?) We don't have an inclination on the answer. But if it is contained in $\mathsf{MAJ} \circ \mathsf{ACC}^0$, it would immediately imply super-polynomial lower bounds for $\mathsf{THR} \circ \mathsf{THR}$.

6. Vyas and Williams [VW20] conjectured that $\mathsf{SYM} \circ \mathscr{C}$ lower bounds should follow from #SAT algorithms for $\mathscr{C}$, where SYM denotes arbitrary symmetric functions. Can the new techniques in this paper help to prove this conjecture?

Acknowledgment

The first author wants to thank his advisor Ryan Williams for basically everything: introducing the problem of proving lower bounds for $\mathsf{MAJ} \circ \mathsf{ACC}^0$ to him and being very optimistic about its resolution, countless encouraging and valuable discussions during the project, and also suggestions and comments on an early draft of this paper. The second author also wants to thank Ryan Williams for hosting him during the summer of 2019 and many inspiring discussions during the period.

We are also grateful to Shuichi Hirahara, Ce Jin, Guy Kindler, Dana Moshkovitz, Igor Carboni Oliveira, Guy Rothblum, Ron Rothblum, Rahul Santhanam, Ronen Shaltiel, Roei Tell for helpful discussions. We would like to thank Nikhil Vyas for helpful discussions and sharing the manuscript of [VW20].

A PCPP for Formula-Eval with NC^1-computable Proofs

In this section, we verify that the PCPP construction in [Har04] applied to Formula-Eval has NC^1 computable proofs. Formally, we prove Lemma 2.14 restated below.

Reminder of Lemma 2.14. *For any constant $\delta > 0$ there are two constants $0 < s < c < 1$, such that there is a PCP of proximity system for* Formula-Eval *with proximity δ, soundness s, completeness c, random bits $r = O(\log n)$, and query complexity $q = 2$, where each query is simply an* OR *on two bits or their negations. Moreover, there is a circuit of depth $O(\log |C| + \log |w|)$ such that, given a pair $(C, w) \in$ Formula-Eval, outputs a proof π that makes $V(C)$ accepts with probability $\geq c$.*

The rest of this section contains a proof sketch for Lemma 2.14. In particular, we show that a PCPP proof for a YES instance of Formula-Eval can be constructed in NC^1. We choose to follow the PCPP protocol described in Section 5 of [Har04]. Then we use methods in [CW19, Section A] to reduce the query complexity of PCPP to 2 bits.

In the following, we assume the reader is familiar with the Section 3 and 5 of [Har04] (which contains a very accessible self-contained proof of the PCP theorem [AS98, ALM+98]).[28]

Notation. We begin with some notation. Given a circuit of depth d, let $n = 2^{O(d)}$ be an integer greater than the circuit size, $m = \log n / \log\log n$, $|H| = n^{1/m} = \log n$, $d = m|H|$, and $\mathbb{F}$ be a field of characteristic 2 and size in $[Cd^3, 2Cd^3]$ for some large constant C. Let H be a specific subset of $\mathbb{F}$ of size $|H|$. W.l.o.g. assume m and $n^{1/m}$ are integers. We can see that $|\mathbb{F}|^m$ is bounded by a polynomial of $|H|^m$, so the output lengths of problems below (*e.g.* Lemma A.2) are polynomially bounded by their input lengths.

A.1 Low Depth Circuits for Basic Algebraic Tasks

To argue the proof is computable in NC^1, we first review some basic algebraic tasks which are computable in low depth.

In fact, all of them are computable in TC^0, but we choose to use NC^1 circuits for convenience. Throughout the section, we use TC^0 or NC^1 to denote P-uniform TC^0 or NC^1 circuits.

Iterated Addition / Multiplication in low depth. In the *iterated addition* problem, we are given a list $a_1, a_2, \ldots, a_t$ of $\mathsf{GF}(2^n)$ elements, and we want to compute $\sum_{i=1}^{t} a_i$. In the *iterated multiplication* problem, we want to compute $\prod_{i=1}^{t} a_i$ instead. Both problems are solvable in depth $O(\log n + \log t)$ [HAB02].

Two representations of multivariate polynomials. When we deal with multivariate polynomials, there are two types of representations:

- *by coefficients*: a degree-d m-variate polynomial p is represented as a list $\vec{\alpha}$, where

$$p(\vec{x}) = \sum_{e_1+\cdots+e_m \leq d} \alpha_{e_1,\ldots,e_m} \prod_{i=1}^{m} x_i^{e_i} \quad \forall \vec{x} \in \mathbb{F}^m;$$

- *by values*: an m-variate polynomial is represented as the values of $p(\vec{x})$ for every $\vec{x} \in \mathbb{F}^m$.

[28] We wish this section could be self-contained, but that would make it unnecessarily long (we would have to basically provide a proof for PCP theorem). It is highly recommended reading Section 5 of [Har04] before reading this section.

A straightforward corollary of iterated addition being in NC^1 is:

Lemma A.1 (Representation by Coefficients to Representation by Values)**.** *Given an m-variate degree-d polynomial represented by coefficients, we can output its representation by values in depth $O(m \log |\mathbb{F}|)$.*

Extension in low depth. We need to extend a partial function $f : H^m \to \mathbb{F}$ to a low-degree polynomial defined on the entire $\mathbb{F}^m$. We show this is doable in low depth:

Lemma A.2. *Given a function $f : H^m \to \mathbb{F}$ represented by values, there is a depth-$O(m \log |\mathbb{F}|)$ circuit that computes a polynomial $\hat{f} : \mathbb{F}^m \to \mathbb{F}$ with degree at most $|H| - 1$ in each variable, such that for every $\vec{h} \in H^m$, $\hat{f}(\vec{h}) = f(\vec{h})$. The circuit outputs $\hat{f}$ both by coefficients and by values.*

Proof. For $\vec{h} \in H^m$, $\vec{x} \in \mathbb{F}^m$, let

$$p_{\vec{h}}(\vec{x}) = \prod_{i=1}^{m} \prod_{h' \in H \setminus \{h_i\}} (x_i - h'),$$
$$q_{\vec{h}}(\vec{x}) = p_{\vec{h}}(\vec{x}) \cdot p_{\vec{h}}(\vec{h})^{-1}.$$

Then $q_{\vec{h}}$ is a polynomial that takes value 1 on $\vec{h}$ and 0 on all other points in H^m. Since H is predetermined, we can precompute every $q_{\vec{h}}$ both by coefficients and by values, and hardcode them into the circuit. We define

$$\hat{f}(\vec{x}) = \sum_{\vec{h} \in H^m} q_{\vec{h}}(\vec{x}) \cdot f(\vec{h}).$$

It is easy to see that $\hat{f}$ agrees with f in H^m, and is a polynomial with degree at most $|H| - 1$ in each variable. Since iterated addition is in NC^1, $\hat{f}$ can be computed in depth $O(m \log |\mathbb{F}|)$ both by coefficients and by values. □

Single variate low degree testing in low depth. We will frequently use the following algorithm:

Lemma A.3. *Given a function $f : \mathbb{F} \to \mathbb{F}$ and an integer d, there is a depth-$O(\log |\mathbb{F}|)$ circuit that decides whether f is a (univariate) polynomial of degree at most d.*

Proof. Let $\vec{v}$ be a column vector such that $v_x = f(x)$ for $x \in \mathbb{F}$. Let $\vec{\alpha}$ be a column vector such that $f(x) = \sum_{i=0}^{|\mathbb{F}|-1} \alpha_i x^i$. Let A be the Vandermonde matrix such that for every $i \in \mathbb{F}$ and $0 \le j \le |\mathbb{F}| - 1$, $A_{ij} = i^j$. Then A is invertible over $\mathbb{F}$, and we can precompute A^{-1} and hardcode it into the circuit. We have $\vec{\alpha} = A^{-1} \cdot \vec{v}$, thus $\vec{\alpha}$ can be computed in $O(\log |\mathbb{F}|)$ depth. We finish by checking $\alpha_i = 0$ for every $i > d$. □

A.2 Low Depth Verifiers and Composition of PCPPs

The first subtlety arises from the composition theorem: it turns out that we not only need to show that the proofs can be constructed in low depth, but we also need to check that the verifier itself can be implemented by a low depth circuit.

Recall that the verifier is given input x and randomness R, outputs $q(n)$ indices as query bits, and a circuit D with $q(n)$ inputs that decides the output of the verifier, given the query outcomes. The proof required by the composed verifier (refer to the algorithm in [Har04, p.49]) consists of two parts:

- The proof π for the outer verifier.
- For each possible randomness R, a proof that $D(\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_q})$ accepts, where $(i_1, i_2, \ldots, i_q)$ and D are indices and the decision circuit generated by the outer verifier respectively, on randomness R.

Therefore, to compute the second part of the proof, we need to simulate the outer verifier, and compute a proof for $(D, (\pi_{i_1}, \ldots, \pi_{i_q})) \in$ Formula-Eval. That means we require both the outer verifier and D to have small depths. Equivalently, the outer verifier has to be a low depth (non-adaptive) oracle circuit with the proof oracle as the oracle gate. We define the *depth* of a verifier (of Formula-Eval) to be the maximum depth of the following two circuits:

- The verifier itself as an oracle circuit.
- The circuit that computes the proof from a YES input of Formula-Eval.

Let $d_{\mathsf{out}}(n)$ be the depth of the outer verifier, $d_{\mathsf{in}}(n)$ be the depth of the inner verifier, and $|D|$ be the size of the decision circuit outputted by the outer verifier. Inspecting [Har04, p.49], we can see that the depth of the composed verifier (*i.e.* both of its implementation and of proof construction) is $O(d_{\mathsf{out}}(n) + d_{\mathsf{in}}(|D|))$. Therefore if both verifiers are low depth, then the composed verifier is also low depth.

We also care about the size of D (since it is the input length of the inner verifier), so we say the *decision complexity* for a verifier is the size of D. The decision circuit of the composed verifier is the same as the decision circuit of the inner verifier (on input length $|D|$).

Composition of PCPPs. In Appendix A.3, we will verify that the main PCPP described in [Har04, Section 5] has depth $O(m \log |\mathbb{F}|) = O(\log n)$ and decision complexity $\mathrm{poly}(m, |\mathbb{F}|) = \mathrm{polylog}(n)$. We compose the final PCPP as follows:

(a) We compose the PCPP in Appendix A.3 with itself, and obtain a PCPP for Formula-Eval with depth $O(\log n)$ and decision complexity $\mathrm{polylog}(\mathrm{polylog}(n)) = \mathrm{poly}(\log \log n)$.

(b) We fix an arbitrary 3-query PCPP for Formula-Eval (say [CW19, Lemma 25]) whose depth is trivially bounded by $\mathrm{poly}(n)$. The input length for this PCPP is only $\mathrm{poly}(\log \log n)$, so we compose the (previously composed) PCPP in (a) with this PCPP, and obtain a PCPP for Formula-Eval with depth $O(\log n)$ that only needs to query 3 bits of the proof.

(c) We use the same method as [CW19, Appendix A] to transform the PCPP verifier in (b) into a 2-query PCPP with imperfect completeness.[29] The proof of the 2-query PCPP can in fact be computed in NC^0 given the proof of the PCPP in (b). Lemma 2.14 then follows.

A.3 The Main Construct

It remains to check that the verifiers in [Har04, Section 5] have depth $O(m \log |\mathbb{F}|)$ and decision complexity $\mathrm{poly}(m, |\mathbb{F}|)$.

PCPP-LDT. (See [Har04, p.66].) It queries a line $\mathcal{L} : \mathbb{F} \to \mathbb{F}$ represented by values, then checks if $\mathcal{L}$ is a polynomial of degree $\leq d$. By Lemma A.3, this verifier has depth $O(\log(m|\mathbb{F}|))$. Its decision complexity is thus $2^{O(\log(m|\mathbb{F}|))} = \mathrm{poly}(m, |\mathbb{F}|)$.

This verifier receives no proof.

[29] As a technical detail, the transformation in [CW19, Section A] requires the decision circuit of the 3-query PCPP to be an OR_3 gate, which is true for (b). The decision circuit of the final PCPP is an OR_2 gate.

ROBUST-PCPP-ZERO-ON-SUBCUBE. (See [Har04, p.70].) It invokes Lemma A.3 $O(m)$ times, each time checking whether a function f, P_i or $Q_i : \mathbb{F} \to \mathbb{F}$ is a low-degree polynomial. It then performs Division Check [Har04, Eq (5.4)] and Identity Check [Har04, Eq (5.5)] which are $O(m)$ arithmetic operations over $\mathbb{F}$, and computes an AND of fanin $O(m)$. It follows that the verifier has depth-$O(\log(m|\mathbb{F}|))$ circuits. Therefore its decision complexity is $\text{poly}(m, |\mathbb{F}|)$.

The proofs are computed by [Har04, Eq (5.1)], as follows. We are given an m-variate polynomial $p_0(\cdot)$ by coefficients, and for every $1 \le i \le m$, we compute polynomials $p_i(\cdot)$ and $q_i(\cdot)$ as follows:

$$p_{i-1}(\vec{x}) = g_H(x_i) \cdot q_i(\vec{x}) + p_i(\vec{x}),$$

where $g_H(x) = \prod_{h \in H}(x - h)$ is a predetermined univariate polynomial, and $q_i(\vec{x})$, $p_i(\vec{x})$ are the quotient and reminder of $p_{i-1}(\vec{x})$ divided by $g_H(x_i)$.

Consider the task of computing $p(\vec{x}) = g_H(x_i) \cdot q(\vec{x}) + r(\vec{x})$, where we are given the polynomial $p(\cdot)$ represented by coefficients, and want to compute $q(\cdot)$ and $r(\cdot)$ also by coefficients. Let

$$p(\vec{x}) = \sum_{e_1 + \cdots + e_m \le d} \alpha_{e_1,\ldots,e_m} \prod_{j=1}^{m} x_j^{e_j}.$$

For every $i \ge 1$, let $\hat{q}_i(x), \hat{r}_i(x)$ be the quotient and reminder of x^i divided by $g_H(x)$, *i.e.*

$$x^i = g_H(x) \cdot \hat{q}_i(x) + \hat{r}_i(x).$$

Then we have

$$q(x_1, \ldots, x_m) = \sum_{e_1 + \cdots + e_m \le d} \alpha_{e_1,\ldots,e_m} \cdot \hat{q}_{e_i}(x_i) \cdot \prod_{j \ne i} x_j^{e_j},$$
$$r(x_1, \ldots, x_m) = \sum_{e_1 + \cdots + e_m \le d} \alpha_{e_1,\ldots,e_m} \cdot \hat{r}_{e_i}(x_i) \cdot \prod_{j \ne i} x_j^{e_j}.$$

Let $\vec{\alpha}^{(p)}$ as the column vector consisting of the representation of p by coefficients, and define column vectors $\vec{\alpha}^{(q)}$, $\vec{\alpha}^{(r)}$ similarly. Then there are matrices $M^q_{i,H}$ and $M^r_{i,H}$ (superscripts q, r are mnemonics) such that $\vec{\alpha}^{(q)} = M^q_{i,H} \cdot \vec{\alpha}^{(p)}$, and $\vec{\alpha}^{(r)} = M^r_{i,H} \cdot \vec{\alpha}^{(p)}$. Moreover, both $M^q_{i,H}$ and $M^r_{i,H}$ only depends on i and H. Therefore we can precompute the following matrices:

$$P^q_{i,H} = M^q_{i,H} \cdot M^r_{i-1,H} \cdot M^r_{i-2,H} \cdot \cdots \cdot M^r_{1,H}, \quad \text{and} \quad P^r_{i,H} = M^r_{i,H} \cdot M^r_{i-1,H} \cdot \cdots \cdot M^r_{1,H}.$$

It follows that both $\vec{\alpha}^{(p_i)} = P^r_{i,H} \cdot \vec{\alpha}^{(p_0)}$ and $\vec{\alpha}^{(q_i)} = P^q_{i,H} \cdot \vec{\alpha}^{(p_0)}$ can be computed in depth $O(m \log |\mathbb{F}|)$. We then use Lemma A.1 to compute the representations by values of each p_i and q_i, also in depth $O(m \log |\mathbb{F}|)$.

ROBUST-PCP-CIRCUIT-SAT. (See [Har04, p.75].) The verifier computes a function $C' : H^{3m+3} \to \mathbb{F}$ from the input circuit C ([Har04, Eq (5.6)]), and uses Lemma A.2 to extend it to a $(3m+3)$-variate polynomial $\hat{C}$ over $\mathbb{F}$. It then invokes PCPP-LDT and ROBUST-PCPP-ZERO-ON-SUBCUBE, and does $O(m)$ additional arithmetic operations over $\mathbb{F}$. It follows that the verifier has depth-$O(m \log |\mathbb{F}|)$ circuits. Its decision complexity is the sum of the decision complexity of PCPP-LDT, ROBUST-PCPP-ZERO-ON-SUBCUBE and $O(m)$ arithmetics over $\mathbb{F}$, thus bounded by $\text{poly}(m, |\mathbb{F}|)$.

In our application, we are given a depth-d circuit C and a satisfying assignment w of C, and we want to compute a valid proof for this verifier that $(C, w) \in \text{Formula-Eval}$. We first compute an "extended assignment" A in depth $O(d)$ which contains the output value of every gate in C. The proof consists of the following.

- An extension $\hat{A}$ of A computable in depth $O(m \log |\mathbb{F}|)$ by Lemma A.2.
- A polynomial $p_{(\hat{A})}$ computed from A by [Har04, Eq (5.7)], which can be done in depth $O(\log(m|\mathbb{F}|))$.
- The proof oracle for ROBUST-PCPP-ZERO-ON-SUBCUBE. Recall that we need a representation of $p_{(\hat{A})}$ by coefficients to compute a proof for this verifier. It follows from [Har04, Eq (5.7)] that it suffices to compute both $\hat{A}$ and $\hat{C}$ by coefficients. Since both $\hat{A}$ and $\hat{C}$ are computed by Lemma A.2, our circuit can easily obtain these representations by coefficients.

It follows that this verifier has depth $O(d + m \log |\mathbb{F}|)$, which is $O(\log n)$ when $d = O(\log n)$.

ROBUST-PCPP-CIRCUIT-SAT. (See [Har04, p.78].) The verifier invokes ROBUST-PCP-CIRCUIT-SAT, tests whether a line $\mathcal{L} : \mathbb{F} \to \mathbb{F}$ is a low-degree polynomial, and performs $O(1)$ extra arithmetic operations over $\mathbb{F}$. It follows that the verifier has depth $O(d + m \log |\mathbb{F}|)$ and decision complexity $\text{poly}(m, |\mathbb{F}|)$.

The verifier needs no extra proof except those of ROBUST-PCP-CIRCUIT-SAT.

Translating into binary alphabet. The proof oracle for ROBUST-PCPP-CIRCUIT-SAT consists of entries in the alphabet $\Sigma = \mathbb{F}^{6m+6}$. Fix an error correcting code $(\mathsf{Enc}, \mathsf{Dec})$ as in Lemma 2.11, where Enc has NC^1 circuits. Let the original proof oracle be Γ, we augment it with an auxiliary proof oracle Υ. For each entry $\Gamma[x]$ in Γ, we interpret $\Gamma[x]$ as a $(\log |\Sigma|)$-bit string and let $\Upsilon[x] = \mathsf{Enc}(\Gamma[x])$. For every entry $\Gamma[x]$ the old verifier probes, the new verifier also probes $\Upsilon[x]$ and tests whether $\Upsilon[x] = \mathsf{Enc}(\Gamma[x])$. This transformation adds at most $O(m \log |\mathbb{F}|)$ to the verifier depth, and multiplies at most $\text{poly}(m, |\mathbb{F}|)$ to the decision complexity.

B A Self-contained Exposition of Properties of CMD and DCMD

The nice random reducibility properties of the problem Connected Matrix Determinant (CMD) and Decomposed Connected Matrix Determinant (DCMD) are the key ingredients of our new average-case circuit lower bounds. We believe these two problems could be useful for other questions in average-case complexity, or complexity theory in general.

Therefore, in this section, we provide a self-contained exposition of the properties of these problems here. We also refer the readers to [IK02, GGH+07, GGH+08, Vio09b, FSUV13, App14] for more applications and motivations of these two problems.

In this section, all matrices and vectors are over $\mathsf{GF}(2)$. We may use $*$ to denote an arbitrary element in $\mathsf{GF}(2)$.

We first recall the definitions of CMD and DCMD.

Reminder of Definition 2.18. An instance of CMD is an $n \times n$ matrix over $\mathsf{GF}(2)$ where the main diagonal and above may contain any entry, the second diagonal (*i.e.* the one below the main diagonal) contains 1 and other entries are 0. In other words, the matrix is of the following form

(where $*$ represents any element in $\mathsf{GF}(2)$):

$$\begin{pmatrix} * & * & * & \cdots & * & * \\ 1 & * & * & \cdots & * & * \\ 0 & 1 & * & \cdots & * & * \\ 0 & 0 & 1 & \cdots & * & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & * \end{pmatrix}. \tag{11}$$

The instance is an $(n(n+1)/2)$-bit string specifying elements on and above the main diagonal. We define $x \in \mathsf{CMD}$ if and only if the determinant (over $\mathsf{GF}(2)$) of the matrix corresponding to x is 1.

An instance of DCMD is a string of length $n^3(n+1)/2$. For an input x, $\mathsf{DCMD}(x)$ is computed as follows: we partition x into blocks of length n^2, let $y_i (1 \le i \le n(n+1)/2)$ be the parity of the i-th block, and define $\mathsf{DCMD}(x) := \mathsf{CMD}(y_1 \circ y_2 \circ \cdots \circ y_{n(n+1)/2})$.

We say an $n \times n$ matrix is a *valid* CMD *matrix* if it is of the form of (11).

B.1 $\oplus$L-completeness

We first show that CMD is $\oplus$L-complete (it is then easy to see that DCMD is also $\oplus$L-complete by definition). The following theorem is proved in [IK97].

Reminder of Theorem 2.20. CMD *is* $\oplus$L*-complete under projections.*

Proof. We first prove $\oplus$L-hardness. Consider any problem $L \in \oplus\mathsf{L}$. Suppose that on input length n, L is solved by a parity branching program $P = (G, \sigma, f)$, where:

- $G = (V, E)$ is a directed acyclic graph with $m = \mathrm{poly}(n)$ nodes.
- $\sigma_1, \sigma_2, \ldots, \sigma_m$ is a permutation of V that forms a topological order of G.
- $f : E \to [n] \times \{0,1\}$ gives each edge in G a label. Intuitively, let $e \in E$ and $f(e) = (i, b)$, then e "reads" the i-th bit of the input and "works" if this bit is equal to b.
- For $x \in \{0,1\}^n$, we define G_x as a subgraph of G, such that an edge e is in G_x if and only if $f(e) = (i, b)$ and $x_i = b$. Then $x \in L$ if and only if G_x contains an odd number of paths that starts at σ_1 and ends at σ_m.

Since G is fixed in advance, for any $x \in \{0,1\}^n$, the adjacency matrix A_x of G_x can be computed by a projection over x. In particular, $(A_x)_{ij} = 1$ if and only if there is an edge $e \in G_x$ from σ_i to σ_j. Since σ is a topological order of G, A_x is always an upper triangular matrix (with zeros on the diagonal). Let I_m be the $m \times m$ identity matrix over $\mathsf{GF}(2)$, then $I_m - A_x$ has full rank, and

$$(I_m - A_x)^{-1} = \sum_{i=0}^{m} (A_x)^i.$$

(To see this, multiply both sides by $(I_m - A_x)$, and notice that since G_x is an m-vertex DAG, $(A_x)^{m+1}$ is the all-zero matrix.)

Let $B = (I_m - A_x)^{-1}$, then for any $s, t \in V(G_x)$, B_{st} is the parity of the number of s-t paths in G_x. Since $\det(I_m - A_x) = 1$, we have $B_{st} = \mathrm{adj}(I_m - A_x)_{st}$, where $\mathrm{adj}(A)$ is the adjugate matrix of

A. Recall that the (s,t)-th element of $\mathrm{adj}(A)$ is the (t,s)-th cofactor of A, which (over $\mathsf{GF}(2)$) is the determinant of the sub-matrix of A formed by deleting the t-th row and s-th column. Let $C_{m,1}$ be the sub-matrix formed by removing the m-th row and first column of $I_m - A_x$. It follows that

$$B_{1,m} = \det(C_{m,1}).$$

It's easy to verify that $C_{m,1}$ is a valid CMD matrix and is computable by a projection over x. Therefore L reduces to CMD by a projection, and CMD is $\oplus\mathsf{L}$-hard.

By the same reasoning as above, we can see that $\mathsf{CMD} \in \oplus\mathsf{L}$: Given a valid $(n-1)\times(n-1)$ CMD matrix C, we create a DAG G whose vertex set is $[n] = \{1,2,\ldots,n\}$. For $1 \le i \le j \le n-1$, if $C_{ij} = 1$, we add to G an edge from vertex i to vertex $(j+1)$. Let A be the adjacency matrix of G, $B = (I_n - A)^{-1}$, then C is the sub-matrix of B formed by deleting the n-th row and the first column. As argued above, $\det(C)$ over $\mathsf{GF}(2)$ is the parity of the number of paths from vertex 1 to vertex n, thus can be easily computed in $\oplus\mathsf{L}$. □

B.2 A Randomized Reduction from CMD to DCMD

We prove Theorem 2.19 in this section. We follow the exposition in [Vio09b] and [GGH+07].

Let $\mathcal{R}_1$ be the set of upper-triangular matrices over $\mathsf{GF}(2)$ with 1's in the diagonal:

$$\begin{pmatrix} 1 & * & \cdots & * & * \\ 0 & 1 & \cdots & * & * \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & * \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Let $\mathcal{R}_2$ be the subset of $\mathcal{R}_1$ where every 1's are either in the diagonal or in the last column:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & * \\ 0 & 1 & \cdots & 0 & * \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & * \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

It is easy to see that both $\mathcal{R}_1$ and $\mathcal{R}_2$ form groups w.r.t. matrix multiplication.

We need the following lemma.

Lemma B.1 ([Vio09b]). *For every valid* CMD *matrix A, there is some $R_1 \in \mathcal{R}_1$ and $R_2 \in \mathcal{R}_2$ such that*

$$R_1 \cdot A \cdot R_2 = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & \det(A) \\ 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}. \tag{12}$$

Proof. Since A is a valid CMD matrix, we can write A as

$$A = \begin{pmatrix} \vec{u}^\top & w \\ A' & \vec{v} \end{pmatrix},$$

where $\vec{u}, \vec{v} \in \mathsf{GF}(2)^{n-1}, w \in \mathsf{GF}(2)$, and A' is a $(n-1) \times (n-1)$ upper triangular matrix over $\mathsf{GF}(2)$, with 1s along the diagonals. Therefore, A' is invertible. We let

$$R_1 = \begin{pmatrix} 1 & \vec{u}^\top (A')^{-1} \\ \mathbf{0} & (A')^{-1} \end{pmatrix},$$

where $\mathbf{0}$ is the all-zero column vector in $\mathsf{GF}(2)^{n-1}$. Since A' is upper-triangular with 1's along the diagonal, it is easy to see that $R_1 \in \mathcal{R}_1$. We have

$$R_1 \cdot A = \begin{pmatrix} \mathbf{0}^\top & w + \vec{u}^\top (A')^{-1} \vec{v} \\ I_{n-1} & (A')^{-1} \vec{v} \end{pmatrix},$$

where I_{n-1} is the $(n-1) \times (n-1)$ identity matrix over $\mathsf{GF}(2)$. We then define

$$R_2 = \begin{pmatrix} I_{n-1} & (A')^{-1} \vec{v} \\ \mathbf{0}^\top & 1 \end{pmatrix},$$

then it is easy to see $R_2 \in \mathcal{R}_2$. We have

$$R_1 \cdot A \cdot R_2 = \begin{pmatrix} \mathbf{0}^\top & w + \vec{u}^\top (A')^{-1} \vec{v} \\ I_{n-1} & 0 \end{pmatrix},$$

which is of the form (12). Since $\det(R_1) = \det(R_2) = 1$, we have

$$\det(A) = \det(R_1 \cdot A \cdot R_2) = w + \vec{u}^\top (A')^{-1} \vec{v}. \qquad \square$$

Inspecting the above proof, we also have:

Corollary B.2. *Let A be a valid* CMD *matrix and A' be the same matrix as A, except that we flip the $(1,n)$-th entry in A'. Then* $\det(A) \oplus \det(A') = 1$.

The following lemma and proof comes from [GGH+07]:

Reminder of Theorem 2.19. *There is a function $P : \{0,1\}^{n(n+1)/2} \times \{0,1\}^{O(n^4)} \to \{0,1\}^{n^3(n+1)/2}$ such that the following hold.*

- *For any input $x \in \{0,1\}^{n(n+1)/2}$, the random variable $P(x, \mathcal{U}_{O(n^4)})$ is uniformly distributed in $\{0,1\}^{n^3(n+1)/2}$.*
- *For any $x \in \{0,1\}^{n(n+1)/2}$ and $r \in \{0,1\}^{O(n^4)}$, let $P(x,r) = y$, then $\mathsf{CMD}(x) = \mathsf{DCMD}(y) \oplus r_0$, where r_0 is the first bit of r.*
- *For each fixed randomness r, $P(x,r)$ is a projection over x, computable in polynomial time given r.*

Proof. We first describe a random self-reduction for CMD. Given a valid CMD matrix A, we choose a random bit $\mathbf{b} \sim \{0,1\}$, and two random matrices $\mathbf{R}_1 \sim \mathcal{R}_1, \mathbf{R}_2 \sim \mathcal{R}_2$. We then compute $\mathbf{A}' = \mathbf{R}_1 \cdot A \cdot \mathbf{R}_2$, and flip the entry $\mathbf{A}'_{1n}$ if $\mathbf{b} = 1$. We output $P(A, \mathbf{r}) = \mathbf{A}'$ where $\mathbf{r}$ is the random string encoding $(\mathbf{b}, \mathbf{R}_1, \mathbf{R}_2)$.

For $b \in \mathsf{GF}(2)$, let M_b be the matrix that has 1 on the second diagonal, b in the $(1,n)$-th entry, and 0 in all other entries. (For instance, the RHS of (12) is equal to $M_{\det(A)}$.) Let $A^\star$ be any valid

CMD matrix such that $\det(A) = \det(A^\star)$, then there are matrices $R_1, R_1^\star \in \mathcal{R}_1$ and $R_2, R_2^\star \in \mathcal{R}_2$, such that

$$R_1 \cdot A \cdot R_2 = M_{\det(A)} = R_1^\star \cdot A^\star \cdot R_2^\star.$$

It follows that

$$\left((R_1^\star)^{-1} R_1\right) \cdot A \cdot \left(R_2 (R_2^\star)^{-1})\right) = A^\star.$$

Therefore, for every valid CMD matrix $A^\star$ such that $\det(A) = \det(A^\star)$, there are $R_1^\circ \in \mathcal{R}_1$ and $R_2^\circ \in \mathcal{R}_2$ such that $R_1^\circ \cdot A \cdot R_2^\circ = A^\star$. Since $|\mathcal{R}_1| \cdot |\mathcal{R}_2| = 2^{n(n-1)/2} \cdot 2^{n-1}$, and there are $2^{n(n+1)/2-1} = |\mathcal{R}_1| \cdot |\mathcal{R}_2|$ valid CMD matrices with determinant equal to $\det(A)$, we conclude that for each $A, A^\star$ with the same determinant, there is *exactly* one pair of (R_1, R_2) that transforms A into $A^\star$. Now it is easy to see that $\mathbf{A}'$ is uniformly randomly distributed among valid CMD instances.

Unfortunately, this reduction cannot be implemented by projections. Still, the same method can be used to construct a randomized projection that reduces CMD to DCMD. In particular, for every $1 \le i, j \le n$, we have

$$\mathbf{A}'_{ij} = \bigoplus_{k=1}^{n} \bigoplus_{l=1}^{n} \left((\mathbf{R}_1)_{ik} \cdot A_{kl} \cdot (\mathbf{R}_2)_{kj}\right).$$

(The only exception is that we may flip $\mathbf{A}'_{1n}$.) Recall that the instance for DCMD is a length-$n^3(n+1)/2$ string partitioned into chunks of length n^2. For each $1 \le i \le j \le n$ we sample a random string $\mathbf{z}^{(ij)}$ of length n^2. For each $1 \le i \le j \le n$ and each $1 \le k, l \le n$, the $((k-1)n+l)$-th bit of the chunk corresponding to $\mathbf{A}'_{ij}$ is

$$\mathbf{z}^{(ij)}_{(k-1)n+l-1} \oplus \left((\mathbf{R}_1)_{ik} \cdot A_{kl} \cdot (\mathbf{R}_2)_{kj}\right) \oplus \mathbf{z}^{(ij)}_{(k-1)n+l}, \tag{13}$$

where we define $\mathbf{z}^{(ij)}_0 = \mathbf{z}^{(ij)}_{n^2}$. We then flip the last bit of the chunk corresponding to $\mathbf{A}'_{1n}$ if $\mathbf{b} = 1$.

For every string x of length n^2 (think of x as the concatenation of $\left((\mathbf{R}_1)_{ik} \cdot A_{kl} \cdot (\mathbf{R}_2)_{kj}\right)$ over all $1 \le k, l \le n$), and every string y of length n^2 and same parity with x (think of y as the outcome of (13)), there are exactly two strings z of length n^2 such that the concatenation of

$$(z_{i-1} \oplus x_i \oplus z_i)_{i=1}^{n^2}$$

is y. Therefore, the DCMD instance we generated is a uniformly random one. Furthermore, it is easy to verify that for each fixed randomness, (13) is a projection over the input matrix A. □

C Technical Building Blocks for Section 4

In this section, we prove the technical building blocks used for Section 4. In particular, we prove Theorem 4.4 in Appendix C.1, and prove Theorem 4.5 in Appendix C.3. In Appendix C.2 we improve the result in Section 8 of [Che19], which plays a crucial part of the proof of Theorem 4.5.

C.1 A PRG Construction for Low Depth Circuits

We first prove Theorem 4.4, which shows one can construct a PRG with nearly optimal seed length fooling low depth circuits, with a truth-table which is only worst-case hard for low depth circuits.

Reminder of Theorem 4.4. *Let $d(\ell) = \omega(\log \ell)$. There are universal constants c and g, and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for every good resource function $d : \mathbb{N} \to \mathbb{N}$, the following*

hold. Let $s_{\mathsf{seed}} = g\ell^2/d(\ell)$ *and* $S_{\mathsf{out}} = 2^{c\cdot d(\ell)}$*, if* $Y : \{0,1\}^\ell \to \{0,1\}$ *does not have circuits of depth* $d(\ell)$*, then for all circuits* C *with depth* $\log(S_{\mathsf{out}})$*,*

$$\left| \Pr_{\mathbf{t}\sim\mathcal{U}_{s_{\mathsf{seed}}}} [C(G(Y,\mathbf{t})) = 1] - \Pr_{\mathbf{x}\sim\mathcal{U}_{S_{\mathsf{out}}}} [C(\mathbf{x}) = 1] \right| < 1/S_{\mathsf{out}}. \tag{14}$$

That is, $G(Y,\cdot)$ $1/S_{\mathsf{out}}$*-fools all* $\log(S_{\mathsf{out}})$*-depth circuits. Moreover,* G *is computable in* $2^{O(s_{\mathsf{seed}})}$ *time.*

Before presenting the proof, let us discuss the previous approach by [Che19] and point out why we need a different one.

Review of [Che19]. In [Che19], we are given a hard truth table Y which is worst-case hard for low depth circuits. We first amplify it by Theorem 2.17 to a truth table that cannot be $(1/2+2^{-\ell^\delta})$-approximated by low depth circuits, and then plug it into the Nisan-Wigderson [NW94] PRG.

In our case when $d(\ell) = 2^{\Omega(\ell)}$, we need to stretch $s_{\mathsf{seed}} = O(\ell)$ random bits to $S_{\mathsf{out}} = 2^{\Omega(\ell)}$ pseudorandom bits. The hybrid argument of the Nisan-Wigderson PRG would require a $(1/2 + 2^{-\Omega(\ell)})$-inapproximable truth table. However, Theorem 2.17 does not produce $(1/2+\varepsilon)$-inapproximable truth tables when $\varepsilon < 2^{-\omega(\sqrt{\ell})}$. Therefore we have to take a different approach.

It turns out that [STV01][30] only requires a mildly hard (*e.g.* 0.99-inapproximable) truth table. Therefore, to prove Theorem 4.4, we can first use Theorem 2.17 to transform the (worst-case) hard truth table Y into a 0.99-inapproximable one, and then plug it into the PRG constructed in [STV01].

C.1.1 A Brief Review of [STV01]

We give a very brief introduction to the PRG construction of [STV01]. Recall that the reason we cannot use Nisan-Wigderson PRG directly is that we cannot amplify worst-case hardness to $2^{-\Omega(n)}$ hardness directly by Theorem 2.17. Fortunately, the PRG in [STV01] only needs a hard truth table that cannot be 0.99-approximated. It then uses a variant of the Nisan-Wigderson generator that stretches a short random seed into a distribution over longer strings. Since the truth table is *not* $(1/2+\varepsilon)$-hard, the distribution of output string is *not* indistinguishable from the *uniform* distribution. Instead, [STV01] managed to show that this distribution is indistinguishable from some distribution with high *min-entropy*. One can then apply an *extractor* to convert this distribution into one which is indistinguishable from the uniform one. We refer the interested readers to the original paper of [STV01] for more details of this approach.

The actual definition of min-entropy is not very important here, but we include it for completeness.

Definition C.1. Let X be a random variable, we define the *min-entropy* of X, denoted as $H_\infty(X)$, as the largest real number k such that for any x in the support of X,

$$\Pr[X = x] \le 2^{-k}.$$

Definition C.2. A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)*-extractor* if, for any distribution $\mathcal{D}$ over $\{0,1\}^n$ that has min-entropy at least k, the statistical distance between the distribution $\mathsf{Ext}(\mathcal{D},\mathcal{U}_d)$ and $\mathcal{U}_m$ is at most ε.

We also need the following construction of extractors, which can be computed in $\mathsf{AC}^0[\oplus]$.

[30] Indeed, [STV01] provides two different PRG construction, one is based on pseudo-entropy generator and extractor, another one is based on local-list decodable codes. We are going to apply the first one.

Lemma C.3 ([SU05]). *There is a (k,ε)-extractor $\mathsf{Ext}:\{0,1\}^n\times\{0,1\}^{O(\log n)}\to\{0,1\}^m$ with $k=n^{2/3}$, $\varepsilon=n^{-2/3}$ and $m=n^{1/3}$. Moreover, Ext can be implemented in $\mathsf{AC}^0[\oplus]$.*[31]

In our final PRG, we will combine the following theorem with the above extractor construction.

Lemma C.4 ([STV01, Theorem 11]). *Suppose we are given a truth table Y of length 2^ℓ that cannot be $(1-\delta)$-approximated by depth-d circuits. For any $m\le 2^\ell$, there is a generator $\mathrm{PE}^Y:\{0,1\}^{O(\ell^2/\log m)}\to\{0,1\}^m$ computable in $\mathrm{poly}(m,2^{\ell^2/\log m})$ time such that the following hold. There is a distribution $\mathcal{D}$ of min-entropy at least $\delta m/2$ such that, any adversary circuit of depth $d-C\log m$ does not distinguish between $\mathrm{PE}^Y(\mathcal{U}_{O(\ell^2/\log m)})$ and $\mathcal{D}$ with advantage $\Omega(1/\delta m)$, where C is a universal constant.*

Proof Sketch. Everything except the adversary depth follows directly from [STV01]. It is easy to verify that the construction has depth $O(\log m)$. □

C.1.2 Proof of Theorem 4.4

Now we are ready to prove Theorem 4.4.

Proof of Theorem 4.4. Let $(\mathsf{Amp},\mathsf{Dec})$ be a $(0.01,0)$-fully-black-box hardness amplification in Theorem 2.17. We denote the truth table of Amp^Y (of length $2^{O(\ell)}$) as Y'. If Y cannot be computed by depth-$d(\ell)$ circuits, then Y' cannot be 0.99-approximated by $c_{\mathsf{GR}}\cdot d(\ell)$-depth circuits for a universal constant $c_{\mathsf{GR}}>0$.

Fix a small constant $\gamma>0$ and let $m=2^{\gamma\cdot d(\ell)}$. From Lemma C.4, we obtain a generator $\mathrm{PE}^{Y'}$ stretching from $O(\ell^2/\log m)=O(\ell^2/d(\ell))$ bits to m bits. This generator $\mathrm{PE}^{Y'}(\mathcal{U}_{O(\ell^2/d(\ell))})$ is indistinguishable from some distribution $\mathcal{D}$ of min-entropy at least $\Omega(m)$ with advantage $\Omega(1/m)$, by adversary circuits of depth $(c_{\mathsf{GR}}\cdot d(\ell)-O(\log m))\ge c_{\mathsf{GR}}/2\cdot d(\ell)$.

Let $\mathsf{Ext}:\{0,1\}^m\times\{0,1\}^{O(\log m)}\to\{0,1\}^{m^{1/3}}$ be the (k,ε)-extractor specified in Lemma C.3, with $k=m^{2/3}$ and $\varepsilon=m^{-2/3}$. Let $s_{\mathsf{seed}}=O(\ell^2/d(\ell))+O(\log m)=O(\ell^2/d(\ell))$ be the sum of the seed lengths of $\mathrm{PE}^{Y'}$ and Ext.

The generator G treats the seed $t\in\{0,1\}^{s_{\mathsf{seed}}}$ as the concatenation of two strings $t_1\circ t_2$, where $|t_1|=O(\ell^2/d(\ell))$ and $|t_2|=O(\log m)=O(d(\ell))$, and outputs $\mathsf{Ext}\left(\mathrm{PE}^{Y'}(t_1),t_2\right)$. Note that the length of the output is $S_{\mathsf{out}}=\Omega(m)^{1/3}=2^{\Omega(d(\ell))}$.

Now, let C be a circuit with depth at most $c_{\mathsf{GR}}/3\cdot d(\ell)$. First, since $\mathcal{D}$ has min-entropy at least $\Omega(m)$, it follows

$$\left|\mathbb{E}[C(\mathsf{Ext}(\mathcal{D},\mathcal{U}_{O(\log n)}))]-\mathbb{E}[C(\mathcal{U}_{S_{\mathsf{out}}})]\right|\le\varepsilon.$$

Now, note that for each fixed $u\in\{0,1\}^{O(\log n)}$, $C(\mathsf{Ext}(\cdot,u))$ is a circuit of depth at most $c_{\mathsf{GR}}/2\cdot d(\ell)$. Hence, let $\mathcal{E}=\mathrm{PE}^{Y'}(\mathcal{U}_{O(\ell^2/d(\ell))})$, it follows

$$|\mathbb{E}[C(\mathsf{Ext}(\mathcal{D},u))]-\mathbb{E}[C(\mathsf{Ext}(\mathcal{E},u))]|\le O(1/m),$$

and hence

$$\left|\mathbb{E}[C(\mathsf{Ext}(\mathcal{D},\mathcal{U}_{O(\log n)}))]-\mathbb{E}[C(\mathsf{Ext}(\mathcal{E},\mathcal{U}_{O(\log n)}))]\right|\le O(1/m).$$

Putting them together, we have

$$\left|\mathbb{E}[C(\mathsf{Ext}(\mathcal{E},\mathcal{U}_{O(\log n)}))]-\mathbb{E}[C(\mathcal{U}_{S_{\mathsf{out}}})]\right|\le\varepsilon+O(1/m),$$

[31] Indeed, it is a *linear* extractor in the sense that for every $u\in\{0,1\}^{O(\log n)}$, the function $\mathsf{Ext}(\cdot,u)$ is a linear function over $\mathsf{GF}(2)$.

which establishes the correctness of the PRG.

Finally, by Theorem 2.17, we can compute Y' in $2^{O(\ell)}$ time. By [STV01], given Y', the above PRG can be computed in $2^{O(\ell^2/d(\ell))}$ time, which completes the proof. □

C.2 A TC^0 Error Correctable PSPACE-complete Language

In this section we review the PSPACE-complete language L^{PSPACE} constructed in Section 8 of [Che19] with nice reducibility properties (see Definition C.7). This language will be used in Appendix C.3 to prove lower bounds for $\mathsf{MA} \cap \mathsf{coMA}$.

We show that L^{PSPACE} is indeed TC^0 error correctable. Previously [Che19] only proved that L^{PSPACE} is NC^3 error correctable, so it did not imply circuit lower bounds for NP.

Theorem C.5 ([Che19]). *There is a constant $d > 0$ and a PSPACE-complete language $L = L^{\mathsf{PSPACE}}$ that is paddable, TC^0 downward self-reducible, TC^0 same-length checkable, and $(1 - 1/n^d, \mathsf{TC}^0)$ error-correctable.*

Remark C.6. [Che19] established the error correctability with a constant error, which requires to apply a more involved error corrector for polynomials, where the computational bottleneck is solving linear equations (which we don't know how to implement even in NC^1). Our observation here is that it indeed suffices to have an error corrector with inverse-polynomial error, so a much simpler error corrector can be used, which can be implemented in TC^0.

C.2.1 Preliminaries

Desired properties. The problem L^{PSPACE} is paddable, downward self-reducible, same-length checkable and error correctable. We will focus on error correctability in this section, but for completeness (and for the sake of Appendix C.3), we define all these properties as follows.

Definition C.7. Let $L : \{0,1\}^* \to \{0,1\}$ be a language and $\mathscr{C}$ be a circuit class. We say L is

- *paddable*, if there is a polynomial-time computable projection Pad such that the following hold. For all integers $1 \le n < m$ and input $x \in \{0,1\}^n$, $x \in L$ iff $\mathsf{Pad}(x, 1^m) \in L$, where $\mathsf{Pad}(x, 1^m)$ is always an m-bit string.
- *$\mathscr{C}$ downward self-reducible*, if there is a polynomial-size uniform oracle $\mathscr{C}$ circuit A with non-adaptive oracles, such that for all $x \in \{0,1\}^n$, $A^{L_{n-1}}(x) = L_n(x)$.
- *$\mathscr{C}$ same-length checkable*, if there is a probablistic polynomial-size uniform oracle $\mathscr{C}$ circuit M with nonadaptive oracles, which receives the input x and randomness r, and outputs $\{0, 1, ?\}$, such that for any x,
  - M only has oracle gates of fanin $|x|$;
  - if M is given L as the oracle, then $\Pr_r[M^L(x) = L(x)] = 1$;
  - for any oracle O, $\Pr_r[M^O(x) = 1 - L(x)] \le 1/3$.

  (We call M the *instance checker* for L.)
- *$(\delta, \mathscr{C})$ error-correctable*, if for every oracle $O : \{0,1\}^n \to \{0,1\}$ that δ-approximates L_n, there is a polynomial-size oracle $\mathscr{C}$ circuit D such that D^O computes L_n exactly.

A TC^0 decoder for Reed-Muller code. We need a TC^0 decoder for Reed-Muller code that has few errors. In particular, given an oracle that is close to a (hidden) low-degree m-variate polynomial P over a field $\mathbb{K}$, it computes the polynomial P correctly on all inputs.

We also observe that the decoder works correctly when the domain of P is restricted to $\mathbb{F}^m$, where $\mathbb{F}$ is a sub-field of $\mathbb{K}$. In other words, now we can only query the oracle about inputs in $\mathbb{F}^m$, and we also only want to compute P on this domain. (This observation will be useful for handling the field-transferring polynomials $F^{\mathsf{trans}}_{k,i}$ in Appendix C.2.2.)

Lemma C.8 (Low Depth Decoder for Reed-Muller Code, [AB09, Section 19.3, 19.4])**.** *Let $\mathbb{K} = \mathsf{GF}(2^n)$, $\mathbb{F}$ be a sub-field of $\mathbb{K}$, and m be an integer. Suppose there is a (hidden) degree-d m-variate polynomial P over $\mathbb{K}$, and $\delta < \frac{1}{3(d+1)}$. For any oracle $O : \mathbb{F}^m \to \mathbb{K}$ such that*

$$\Pr_{\vec{\mathbf{x}} \sim \mathbb{F}^m}[O(\vec{\mathbf{x}}) = P(\vec{\mathbf{x}})] > 1 - \delta,$$

there is a TC^0 circuit C of size $\mathrm{poly}(m, n)$ with nonadaptive O oracle gates, such that for every $\vec{x} \in \mathbb{F}^m$, $C^O(\vec{x}) = P(\vec{x})$.

Proof. Let $\vec{x} \in \mathbb{F}^m$ be the input, recall that we want to compute $P(\vec{x})$. We will show a randomized TC^0 oracle circuit $\mathbf{C}^O$ (with O oracle gates) of size $\mathrm{poly}(m, n)$ that computes $P(\vec{x})$ w.p. $\geq 2/3$. Then by Adleman's argument, we can obtain a deterministic TC^0 oracle circuit of size $\mathrm{poly}(m, n)$ that correctly computes P, by drawing $\mathrm{poly}(m, n)$ samples from $\mathbf{C}^O$ and taking their majority votes.

We choose a random vector $\vec{\mathbf{v}} \sim \mathbb{F}^m$, and for every $t \in \mathbb{F}$ we define $Q(t) = P(\vec{x} + t \cdot \vec{\mathbf{v}})$. We randomly select $d + 1$ values $t_0, t_1, \ldots, t_d \in \mathbb{F}$ and compute $O(\vec{x} + t_i \cdot \vec{\mathbf{v}})$. Let $\mathbf{z}$ denote the number of i's such that $O(\vec{x} + t_i \cdot \vec{\mathbf{v}}) \neq Q(t_i)$, then $\mathbb{E}[\mathbf{z}] \leq \delta(d + 1)$, and by Markov bound $\Pr[\mathbf{z} = 0] \geq 2/3$. If $\mathbf{z} = 0$, we use Lagrange polynomial interpolation to compute $Q(0) = P(\vec{x})$, which is in TC^0 [HAB02]. □

C.2.2 Review of the Constructions in [Che19]

The PSPACE-complete problems with nice properties have found many applications to complexity theory [FS04, TV07, San09, Che19]. The high-level idea of constructing such languages is to engineer the proof of IP = PSPACE [LFKN92, Sha92].

In the following we first give a self-contained but concise exposition of the construction in [Che19], the interested readers are referred to [Che19] for a more detailed exposition.

We start with a family of multivariate polynomials constructed in [TV07], which are arithmetizations of TQBF and thus PSPACE-hard to compute. In particular, for every integer n, we have a list of multivariate polynomials $f_{n,0}, f_{n,1}, \ldots, f_{n,m(n)}$, where $m(n) = \mathrm{poly}(n)$, such that the following hold.

(a) Each $f_{n,i}$ is a degree-$\mathrm{poly}(n)$ polynomial over a field $\mathsf{GF}(2^n)$.

(b) Each $f_{n,m(n)}$ is computable in polynomial time.

(c) Each $f_{n,i}$ (for $i < m(n)$) is computed by one of the following rules from $f_{n,i+1}$:

1. $f_{n,i}(x_1, \ldots, x_\ell) = f_{n,i+1}(x_1, \ldots, x_\ell, 0) \cdot f_{n,i+1}(x_1, \ldots, x_\ell, 1)$;
2. $f_{n,i}(x_1, \ldots, x_\ell) = 1 - (1 - f_{n,i+1}(x_1, \ldots, x_\ell, 0)) \cdot (1 - f_{n,i+1}(x_1, \ldots, x_\ell, 1))$;
3. $f_{n,i}(x_1, \ldots, x_k, \ldots, x_\ell) = x_k \cdot f_{n,i+1}(x_1, \ldots, 1, \ldots, x_\ell) + (1 - x_k) \cdot f_{n,i+1}(x_1, \ldots, 0, \ldots, x_\ell)$.

(The particular rule used to define $f_{n,i}$ is chosen in some way easily computable from n and i.)

In [Che19], to ensure paddability, each polynomial $f_{n,i}$ is defined to be over the field $\mathsf{GF}(2^{\mathsf{pow}(n)})$, where $\mathsf{pow}(n)$ is the smallest power of 2 no less than n. (The reason is that $\mathbb{F} = \mathsf{GF}(2^{2^i})$ is always a sub-field of $\mathbb{K} = \mathsf{GF}(2^{2^{i+1}})$, while the same does not necessarily hold for $\mathbb{F} = \mathsf{GF}(2^i)$ and $\mathbb{K} = \mathsf{GF}(2^{i+1})$.) Then we fix a bijection ϕ_n from $\mathsf{GF}(2^{\mathsf{pow}(n)})$ to $\{0,1\}^{\mathsf{pow}(n)}$, such that for any element $a \in \mathsf{GF}(2^{\mathsf{pow}(m)})$ where $\mathsf{pow}(m) < \mathsf{pow}(n)$, the first $\mathsf{pow}(m)$ bits of $\phi_n(a)$ is equal to the representation $\phi_m(a)$.

The particular method of wrapping these polynomials into one language used in [Che19] is as follows. We list the polynomials in the following order:

$$f_{1,m(1)}, f_{1,m(1)-1}, \ldots, f_{1,0}, f_{2,m(2)}, \ldots, f_{2,0}, \ldots, f_{n,m(n)}, \ldots, f_{n,0}, \ldots,$$

and let $g_k = f_{n,j}$ be the k-th polynomial in the list. Suppose g_k is a d-variate polynomial over $\mathbb{F} = \mathsf{GF}(2^\ell)$ where $\ell = \mathsf{pow}(n)$.

Now we want to wrap $g_1, g_2, \ldots, g_k$ into a single polynomial G_k for a fixed k. We treat all polynomials $g_i (i \le k)$ as d-variate polynomials over $\mathbb{F}$. If g_i has less than d variables, we add some dummy variables at the end. If g_i is not over $\mathbb{F}$, then it is over some sub-field of $\mathbb{F}$, and we can extend g_i as a polynomial over $\mathbb{F}$. Then we interpolate $g_1, g_2, \ldots, g_k$ into a polynomial G_k as follows. For variables $\vec{x} \in \mathbb{F}^d$, $\vec{y} \in \mathbb{F}^k$,

$$G_k(\vec{x}, \vec{y}) = \sum_{i=1}^{k} g_i(\vec{x}) y_i.$$

It seems that we can now wrap these polynomials G_k into one language that satisfies Theorem C.5. However, if G_k and G_{k+1} are over different fields, it is unknown how to compute G_{k+1} using an oracle of G_k. To circumvent this issue, [Che19] introduces the *field-transferring polynomials* $H^{\mathsf{int}}_{k,i}$ as follows. Let the underlying fields of G_k and G_{k+1} be $\mathbb{F} = \mathsf{GF}(2^\ell)$ and $\mathbb{K} = \mathsf{GF}(2^{2\ell})$ respectively, where $\ell = \mathsf{pow}(n)$. Let G_k be n_k-variate ($n_k = d + k$), and $\widetilde{G}_k : \mathbb{K}^{n_k} \to \mathbb{K}$ be the extension of polynomial G_k over $\mathbb{K}$. We construct $n_k + 1$ oracles $H^{\mathsf{int}}_{k,0}, H^{\mathsf{int}}_{k,1}, \ldots, H^{\mathsf{int}}_{k,n_k}$. The i-th oracle $H^{\mathsf{int}}_{k,i}$ receives an input $\vec{x} \in \mathbb{K}^i \times \mathbb{F}^{n_k - i}$, and outputs $\widetilde{G}_k(\vec{x})$. Therefore $H^{\mathsf{int}}_{k,0}$ is actually the same function as G_k, and H^{int}_{k,n_k} is the polynomial $\widetilde{G}_k$ over $\mathbb{K}$. Downward self-reducibility still holds:

- We can use polynomial interpolation to compute each $H^{\mathsf{int}}_{k,i}$ given an oracle of $H^{\mathsf{int}}_{k,i-1}$, which is in TC^0 by Lagrange interpolation formula and the iterated multiplication algorithm [HAB02].
- For every $i \le k$, we can compute the polynomial g_i over the field $\mathbb{K}$ given an oracle of H^{int}_{k,n_k}. We compute g_{k+1} by one of the three rules in Item (c). Then it is easy to compute the polynomial G_{k+1}.

Now we can wrap the polynomials G_k and $H^{\mathsf{int}}_{k,i}$ into a single language as follows.

- For every integer $k \ge 1$, we define a function F_k that encodes the polynomial G_k. In particular, for $\vec{z} \in \mathbb{F}^{n_k}$ and $r \in \{0,1\}^\ell$, define

  $$F_k(\vec{z}, r) = \langle \phi_n(G_k(\vec{z})), r \rangle,$$

  where the inner product is over $\mathsf{GF}(2)$.

  We note that F_k can be seen as a Boolean function on $e(k) = \ell \cdot (n_k + 1)$ input bits.

- For every integer $k \geq 1$, if G_k and G_{k+1} are over different fields $\mathbb{F} = \mathsf{GF}(2^\ell)$ and $\mathbb{K} = \mathsf{GF}(2^{2\ell})$, then for every $0 \leq i \leq n_k$, we similarly define a function $F^{\mathsf{trans}}_{k,i}$ that encodes $H^{\mathsf{int}}_{k,i}$. In particular, for $\vec{z} \in \mathbb{K}^i \times \mathbb{F}^{n_k - i}$ and $r \in \{0,1\}^{2\ell}$, define

$$F^{\mathsf{trans}}_{k,i}(\vec{z}, r) = \langle \phi_{n+1}(H^{\mathsf{int}}_{k,i}(\vec{z})), r \rangle,$$

where the inner product is over $\mathsf{GF}(2)$.

We also note that $F^{\mathsf{trans}}_{k,i}$ can be seen as a Boolean function on $e(k,i) = \ell \cdot (i + n_k + 2)$ inputs. Furthermore, $e(k) < e(k,0) < e(k,1) < \cdots < e(k, n_k - 1) < e(k, n_k) < e(k+1)$.

- For an input $x \in \{0,1\}^m$, let k be the largest k such that $e(k) \leq m$, and i be the largest i such that $e(k,i) \leq m$. If G_k and G_{k+1} are over different fields, we define $x \in L^{\mathsf{PSPACE}}$ if and only if $F^{\mathsf{trans}}_{k,i}(x') = 1$, where x' is the first $e(k,i)$ bits of x. Otherwise, we define $x \in L^{\mathsf{PSPACE}}$ if and only if $F_k(x') = 1$, where x' is the first $e(k)$ bits of x.

C.2.3 L^{PSPACE} is TC^0 Error Correctable

Now we are ready to prove Theorem C.5.

Proof of Theorem C.5. We only need to prove that L^{PSPACE} is $(1 - 1/n^d, \mathsf{TC}^0)$ error correctable for some constant $d > 0$. Other properties are already proved in Section 8 of [Che19].

Let $O : \{0,1\}^m \to \{0,1\}$ be an oracle that $(1 - 1/m^d)$-approximates L^{PSPACE}_m. There are two cases:

1. Suppose L^{PSPACE}_m computes F_k for some k. Let n_k and ℓ be defined as in Appendix C.2.2, and $\mathbb{F} = \mathsf{GF}(2^\ell)$. By Markov bound, for $1 - 1/m^{d-1}$ fraction of inputs $\vec{z} \in \mathbb{F}^{n_k}$, we have

$$\Pr_{\mathbf{r}}[O(\vec{z}, \mathbf{r}) = \langle \phi(G_k(\vec{z})), \mathbf{r} \rangle] \geq 1 - 1/m. \tag{15}$$

We say an input $\vec{z}$ is *good* if (15) holds. If some $\vec{z} \in \mathbb{F}^{n_k}$ is good, then for every $1 \leq i \leq \ell$, we can compute the i-th bit of $\phi(G_k(\vec{z}))$ by the following algorithm:

> We pick $\mathbf{r} \sim \mathcal{U}_\ell$ uniformly at random, and compute $\phi(G_k(\vec{z})) = O(\vec{z}, \mathbf{r}) \oplus O(\vec{z}, \mathbf{r} \oplus e^i)$, where e^i is the ℓ-bit string with 1 on the i-th bit and 0 everywhere else. We repeat this procedure $\mathrm{poly}(n_k, \ell)$ times and take the majority of the results.

By Adleman's argument, we can fix the randomness used by the above algorithm, and obtain a TC^0 oracle circuit C with O oracle gates that correctly computes $\phi(G_k(\vec{z}))$ for every good $\vec{z}$. In other words, C^O computes $\phi(G_k(\vec{z}))$ on a $(1 - 1/m^{d-1})$ fraction of inputs $\vec{z}$. Since $G_k : \mathbb{F}^{n_k} \to \mathbb{F}$ is a degree-$\mathrm{poly}(m)$ polynomial, if d is large enough, we can use Lemma C.8 to compute G_k in the worst-case by a TC^0 oracle circuit with O oracle gates. It is easy to use this circuit to compute F_k.

2. Suppose L^{PSPACE} computes $F^{\mathsf{trans}}_{k,i}$ for some k, i. Let n_k, ℓ be defined as in Appendix C.2.2, $\mathbb{F} = \mathsf{GF}(2^\ell)$ and $\mathbb{K} = \mathsf{GF}(2^{2\ell})$. Similarly we have a TC^0 oracle circuit with O oracle gates that computes $H^{\mathsf{int}}_{k,i}$ on a $(1 - 1/m^{d-1})$ fraction of inputs. We notice that $H^{\mathsf{int}}_{k,i} : \mathbb{K}^i \times \mathbb{F}^{n_k - i} \to \mathbb{K}$ is a polynomial over $\mathbb{K}$. Moreover, since $\mathbb{K}$ is a quadratic extension of $\mathbb{F}$, there is an element $\alpha \in \mathbb{K}$ such that for any $z \in \mathbb{K}$, there are elements $x, y \in \mathbb{F}$ such that $z = x \cdot \alpha + y$. We write $H^{\mathsf{int}}_{k,i}$, in an equivalent form, as a polynomial from $\mathbb{F}^{n_k + i}$ to $\mathbb{K}$: for $\vec{x}, \vec{y} \in \mathbb{F}^i$ and $\vec{z} \in \mathbb{F}^{n_k - i}$,

$$H^{\mathsf{int}}_{k,i}(\vec{x}, \vec{y}, \vec{z}) = H^{\mathsf{int}}_{k,i}(\vec{x} \cdot \alpha + \vec{y}, \vec{z}),$$

where $\cdot$ denotes element-wise scalar multiplication. Then we can still use Lemma C.8 to compute $H^{\mathsf{int}}_{k,i}$ in the worst-case. It is then easy to compute $F^{\mathsf{trans}}_{k,i}$ in the worst-case. □

C.3 A.a.e. Average-Case Lower Bounds for MA ∩ coMA Against Low Depth Circuits

Finally, we prove Theorem 4.5. For that purpose, we need a hard problem in PSPACE constructible by direct diagonalization.

Lemma C.9 ([Che19], folklore)**.** *There is a universal constant c such that for all good resource function $S(n) \le 2^{o(n)}$, there is a language $L^{\mathsf{diag}} \in \mathsf{SPACE}[S(n)^c]$ such that for all sufficiently large n, $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{diag}}_n) > \log S(n)$.*

Let L^{PSPACE} be the PSPACE-complete problem in Theorem C.5. We define a promise problem L', which is essentially a padded version of L^{PSPACE}. In particular, $L' = (L'_{\mathsf{YES}}, L'_{\mathsf{NO}})$ where

$$L'_{\mathsf{YES}} = \{\langle x, 1^s\rangle : x \in L^{\mathsf{PSPACE}} \text{ and } \mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{|x|}) \le \log s\},$$
$$L'_{\mathsf{NO}} = \{\langle x, 1^s\rangle : x \notin L^{\mathsf{PSPACE}} \text{ and } \mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{|x|}) \le \log s\}.$$

The following theorem follows from the same-length checkability of L^{PSPACE}.

Lemma C.10. *L' is in $\mathsf{PromiseMA} \cap \mathsf{coPromiseMA}$. Moreover, on inputs of length n, the predicate depth of L' is $O(\log n)$.*

Proof. Let the input be $\langle x, 1^s\rangle \in L'_{\mathsf{YES}} \cup L'_{\mathsf{NO}}$. Suppose Arthur wants to decide whether $x \in L^{\mathsf{PSPACE}}$. Merlin sends Arthur a circuit C of depth $\le \log s$ and size $\mathrm{poly}(s)$ which is supposed to decide $L^{\mathsf{PSPACE}}_{|x|}$ (such a circuit exists by the promise of L'). Arthur then runs the instance checker M^C for L^{PSPACE} on input x and outputs the result of the instance checker. By the definition of instance checkers:

- if $x \in L^{\mathsf{PSPACE}}$ and the circuit C indeed decides $L^{\mathsf{PSPACE}}_{|x|}$, then Arthur outputs $L^{\mathsf{PSPACE}}(x)$ w.p. 1;
- for any circuit C, Arthur outputs $1 - L^{\mathsf{PSPACE}}(x)$ w.p. at most $1/3$.

Therefore this protocol satisfies MA ∩ coMA promise.

This protocol runs in $\mathrm{poly}(|x|, s)$ time. Since M is a TC^0 nonadaptive oracle circuit, the predicate has depth $O(\log(|x| + s))$. □

Now we are ready to prove Theorem 4.5. The proof follows [Che19, Section 7].

Reminder of Theorem 4.5. *There is a universal constant $d \in \mathbb{N}$ such that the following hold. For all integers $a > 0$, there are constants $b, t > 0$, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[n^b]_{/1}$, such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*

- $\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L_n) > a \cdot \log n$, *or*
- $\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L_m) > a \cdot \log m$, *for some $m \in (n^t, 2n^t) \cap \mathbb{N}$.*

Moreover, the predicate depth of L is $\le b \log n$.

Proof. By Lemma C.9, there is a language $L^{\mathsf{diag}} \in \mathsf{PSPACE}$ such that $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{diag}}) > a \cdot \log n$. Since L^{PSPACE} is PSPACE-complete and paddable, there is a constant t such that length-n instances of L^{diag} reduces to length-n^t instances of L^{PSPACE}.

Let L' be the promise problem in Lemma C.10.

The language L. We first define the language L. On input x, let $n = |x|$ and $m_0 = n^t$. We are going to define some input lengths n as *good*. Our advice bit α_n is set to 1 if n is good, and to 0 otherwise. Let D be a large enough constant. The particular definition is as follows.

- If n is a power of 2, and $\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{m_0}) \leq (a+D)\log m_0$, then n is good. We directly define $L_n = L^{\mathsf{diag}}_n$. To see L^{diag} on input length n can be computed by an $(\mathsf{MA} \cap \mathsf{coMA})$ protocol, we run the reduction from L^{diag} to L^{PSPACE} on x, and obtain a length-n^t instance x' of L^{PSPACE}. We then use Lemma C.10 to decide whether $x \in L^{\mathsf{diag}}$ by checking whether $\langle x', 1^{m_0^{a+D}} \rangle$ is in L'_{YES} or L'_{NO}.
- If there is some integer k such that
  - $n = 2^k + n_1$ for some $1 \leq n_1 < 2^k$,
  - $\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{2^k}) > (a+D)k$, and
  - $n_1 = \max\{n' : n' < 2^k, \text{ and } \mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{n'}) \leq (a+D)k\}$,

  then n is good. Let x' be the first n_1 bits of x, we treat x' as an input to L^{PSPACE}, and check whether $\langle x', 1^{2^{(a+D)k}} \rangle$ is in L'_{YES} or L'_{NO}. We define $x \in L$ if and only if $x' \in L^{\mathsf{PSPACE}}$.
- Other input lengths are not good, and we reject every input of such lengths.

The $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ protocol has running time $O(n^b)$ and predicate depth $b \log n$, for some large enough constant b. (In particular, the reduction from L^{diag} to L^{PSPACE} does *not* count into the depth of the predicate, see Definition 2.1.)

The lower bound. Now we establish the desired lower bound for L. Consider some large enough $\tau \in \mathbb{N}$, let $n = 2^\tau$ and $m_0 = n^t$. There are two cases:

- If $\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{m_0}) \leq (a+D)\log m_0$, then input length n is good, and $L_n = L^{\mathsf{diag}}_n$. By Lemma C.9, if n is large enough, then $\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{diag}}_n) > a \log n$.
- If $\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{m_0}) > (a+D)\log m_0$, let
$$n_1 = \max\{n' : n' < m_0, \text{ and } \mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{n'}) \leq (a+D)\log m_0\}.$$
Then input length $m = m_0 + n_1$ is good, and $m \in (n^t, 2n^t)$. By the definition of L_m, it suffices to show
$$\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{n_1}) > (a+1)\log m_0 > a \log m.$$
By definition of n_1, we have
$$\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{n_1+1}) > (a+D)\log m_0.$$
Since L^{PSPACE} is TC^0 downward self-reducible, there is a constant d_1 such that
$$\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{n_1}) > (a+D-d_1)\log m_0.$$
Since L^{PSPACE} is $(1-1/n^d, \mathsf{TC}^0)$ error-correctable, there is a constant d_2 such that
$$\mathsf{heur}_{(1-n_1^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{PSPACE}}_{n_1}) > (a+D-d_1-d_2)\log m_0.$$
Since $1 - n_1^{-d} < 1 - m^{-d}$, if we set $D > d_1 + d_2 + 1$, then
$$\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L_m) > a \log m.$$

In conclusion, for every large enough $\tau \in \mathbb{N}$, let $n = 2^\tau$, then either $\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L_n) > a \log n$, or there is some $m \in (n^t, 2n^t) \in \mathbb{N}$ such that $\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L_m) > a \log m$. □

D The General Connection Between Strong Average-Case Lower Bounds and CAPP Algorithms

In this section we prove the general versions of Theorem 1.1, Theorem 4.2, and Theorem 5.1. They are the most general versions of our results and are also useful for the PRG constructions in Section 6. As you will see, these general versions involve *very* technical statements. We choose not to present them in the main body of the paper because we want to present the cleanest version first.

Throughout the section, we assume all functions are good resource functions. Recall that $f : \mathbb{N} \to \mathbb{N}$ is a good resource function if the following hold:

- f is nondecreasing, and
- there is a polynomial time algorithm that on input 1^n, outputs the value of $f(n)$.

Additionally, we say a function f is *smooth*, if there is a constant c such that $f(2n) \leq cf(n)$ for every $n \geq 1$.

D.1 Certifying Depth hardness Implies Strong Average-case Circuit Lower Bounds

We first state without proof the following generalization of Theorem 4.5.

Theorem D.1. *Let $S(n) = 2^{o(n)}$ be a good resource function such that for each $n \in \mathbb{N}$, $S(n)$ is a power of 2, and $\log S(n)$ is smooth. There are universal constants $d, t \in \mathbb{N}$, $T(n) = S(S(n)^{O(1)})^{O(1)}$, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T(n)]_{/1}$, such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*

- $\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L_n) > \log S(n)$, *or*
- $\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L_m) > \log S(m)$, *for some $m \in (S(n)^t, 2S(n)^t) \cap \mathbb{N}$.*

Moreover, the predicate depth of L is $\leq \log T(n)$.

We first prove the most general version of Theorem 4.2, which is stated below.

Theorem D.2 (Certifying Depth Hardness to Average-case Circuit Lower Bounds)**.** *Let $d_{\mathsf{cert}}(n) \leq n$ be a depth parameter, $S(n)$ be a size parameter, and $\ell(n) \leq n$ be a function. Suppose*

1. *$\ell(n)$, $d_{\mathsf{cert}}(n)$ and $\log S(n)$ are smooth functions,*
2. *the function $\ell(n)^2 / d_{\mathsf{cert}}(\ell(n))$ is nondecreasing, and*
3. *for every $n \geq 1$, $S(S(S(n)^K)^K)^K \leq 2^{d_{\mathsf{cert}}(\ell(n))}$, where K is a large constant depending on the functions ℓ, S and d_{cert}.*[32]

Let

$$T(n) = \exp\left(K \cdot \frac{\ell(n)^2}{d_{\mathsf{cert}}(\ell(n))}\right).$$

Suppose NE *can certify $d_{\mathsf{cert}}(n)$-depth hardness, then $\mathsf{NTIME}[T(n)]$ cannot be $(1/2 + 1/S(n))$-approximated by circuits of depth $\log S(n)$. The same holds for $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T(n)]_{/1}$ in place of $\mathsf{NTIME}[T(n)]$.*

Proof Sketch. Let C, K' be large enough constants. Let $S'(n)$ be a function that we define later. We assume that $\log S'(n)$ is smooth, and for every $n \in \mathbb{N}$, $S'(n)$ is a power of 2. We also assume that $S'(S'(S'(n)^{K'})^{K'})^{K'} \leq 2^{d_{\mathsf{cert}}(\ell(n))}$. The following proof follows closely with the proof of the first Item of Theorem 4.2 in Section 4.

[32] Actually, the constant K only depends on the "smoothness" of these functions, *i.e.* the constant c such that $\ell(2n) \leq c\ell(n)$, $d_{\mathsf{cert}}(2n) \leq c d_{\mathsf{cert}}(n)$ and $S(2n) \leq S(n)^c$.

Step I: Construction of a mildly-average-case hard language L^{mild}. In the first step, we construct a problem L^{mild} that cannot be $(1-1/\mathrm{poly}(n))$-approximated by circuits of depth $C \cdot \log S'(n)$. This corresponds to Theorem 4.6.

Suppose NE can certify $d_{\mathsf{cert}}(n)$-depth hardness. Let $\ell \in \mathbb{N}$, $s_{\mathsf{seed}} = O(\ell^2/d_{\mathsf{cert}}(\ell))$ and $S_{\mathsf{out}} = 2^{c \cdot d_{\mathsf{cert}}(\ell)}$ for some small constant c. By Theorem 4.4, there is an i.o. NPRG G_ℓ that stretches s_{seed} random bits into S_{out} pseudorandom bits. For infinitely many ℓ's (which we call *good for NPRG*), G_ℓ $1/S_{\mathsf{out}}$-fools circuits of depth $\log(S_{\mathsf{out}})$.

Let $d, t > 0$ be absolute constants, C_1 be a large enough constant. Let $T_{\mathsf{MA}}(n) = S'(S'(n)^{C_1})^{C_1}$, and $L^{\mathsf{hard}} \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T_{\mathsf{MA}}(n)]_{/1}$ be a hard problem (guaranteed by Theorem D.1) with predicate depth $\log T_{\mathsf{MA}}(n)$ such that, for every large enough $\tau \in \mathbb{N}$ and $n = 2^\tau$,

- either $\mathsf{heur}_{(1-n^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_n) > C \log S'(n)$,
- or there is some $m \in (S'(n)^t, 2S'(n)^t)$ such that $\mathsf{heur}_{(1-m^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_m) > C \log S'(m)$.

The language L^{mild} is defined as follows. On input length $\widetilde{n} = \mathsf{pair}(n, \ell_{\mathsf{PRG}})$ (recall the pair function was defined in Section 4.2), we reject every input if any of the following hold:

- ℓ_{PRG} is not good for NPRG (this criterion is encoded by an advice bit);
- $\ell_{\mathsf{PRG}} > \ell(n)$, *i.e.* ℓ_{PRG} is too large and we cannot afford $2^{O(\ell^2_{\mathsf{PRG}}/d_{\mathsf{cert}}(\ell_{\mathsf{PRG}}))}$ running time; or
- $\log(S_{\mathsf{out}}) = c \cdot d_{\mathsf{cert}}(\ell_{\mathsf{PRG}}) < \log T_{\mathsf{MA}}(n)$, *i.e.* ℓ_{PRG} is too small to derandomize L^{hard}_n.

Otherwise we say the input length $\widetilde{n}$ is *non-trivial*. Let x be the first n input bits. We treat x as an input to L^{hard}_n, and accept x if and only if $x \in L^{\mathsf{hard}}$. In particular, we use $G_{\ell_{\mathsf{PRG}}}$ to derandomize the predicate of L^{hard}_n. Let $s_{\mathsf{seed}} = O(\ell(n)^2/d_{\mathsf{cert}}(\ell(n)))$, then $L^{\mathsf{mild}} \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{O(s_{\mathsf{seed}})}]_{/2}$.

It remains to prove the circuit lower bound. For every ℓ_{PRG} good for NPRG, let n be the smallest power of 2 such that $\ell_{\mathsf{PRG}} \le \ell(n)$. Then there is some $m \in \{n\} \cup ((S'(n)^t, 2S'(n)^t) \cap \mathbb{N})$ such that L^{hard}_m cannot be $(1-m^{-d})$-approximated by circuits of $C \log S'(m)$ depth. We check that the input length $\widetilde{m} = \mathsf{pair}(m, \ell_{\mathsf{PRG}})$ is non-trivial:

- By monotonicity of $\ell(\cdot)$, we have $\ell_{\mathsf{PRG}} \le \ell(m)$.
- Since $\ell(n)$ is a smooth function, we have $\ell_{\mathsf{PRG}} > \ell(n/2) \ge c_1 \cdot \ell(n)$ for some constant $c_1 > 0$. Since $d_{\mathsf{cert}}(n)$ is also a smooth function, we have $c \cdot d_{\mathsf{cert}}(\ell_{\mathsf{PRG}}) \ge c_2 \cdot d_{\mathsf{cert}}(\ell(n))$ for some constant $c_2 > 0$. Recall that for some large enough constant K', $T_{\mathsf{MA}}(m) \le S'(S'(S'(n)^{K'})^{K'})^{c_2 K'} \le 2^{c_2 \cdot d_{\mathsf{cert}}(\ell(n))} \le 2^{c \cdot d_{\mathsf{cert}}(\ell_{\mathsf{PRG}})}$. Therefore $\log(S_{\mathsf{out}}) \ge \log T_{\mathsf{MA}}(m)$.

Thus $L^{\mathsf{mild}}_{\widetilde{m}}$ essentially computes L^{hard}_m. Since $\widetilde{m} = \mathsf{pair}(m, \ell_{\mathsf{PRG}}) \le O(m\ell^2_{\mathsf{PRG}})$, and $\ell_{\mathsf{PRG}} \le \ell(n) \le m$, we have $m = \Omega(\widetilde{m}^{1/3})$. It follows that

$$\mathsf{heur}_{(1-\widetilde{m}^{-d})}\text{-}\mathsf{DEPTH}(L^{\mathsf{mild}}_{\widetilde{m}}) \ge C \log S'(\widetilde{m}^{1/3}).$$

Step II: Mild-to-strong hardness amplification. We amplify L^{mild} to a strongly inapproximable language L^{strong}. This corresponds to Theorem 4.7.

Let $m = n^{d+2}$. We use a $(1/2 - 1/S(m), n^{-d})$-black-box hardness amplification $(\mathsf{Amp}, \mathsf{Dec})$ as in Theorem 2.16, from input length n to input length $O(n \cdot n^d \cdot \log(S(m))) \le m$. Let the new language L^{strong} be $\mathsf{Amp}^{L^{\mathsf{mild}}}$. By padding, we assume the language L^{strong} receives exactly $m = n^{d+2}$ inputs. It follows by Theorem 2.16 that L^{strong} cannot be $(1/2 + 1/S(m))$-approximated by circuits of depth $C \log S'(n^{1/3}) - O(\log S(m))$. We also have that $L^{\mathsf{strong}} \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\mathrm{poly}(m, 2^{s_{\mathsf{seed}}})]_{/2}$, where $s_{\mathsf{seed}} = O(\ell(n)^2/d_{\mathsf{cert}}(\ell(n)))$.

Finale. Let $S'(n)$ be the smallest power of 2 no less than $S(n^{3(d+2)})$. Since $\log S(n)$ is smooth, it follows that $\log S'(n)$ is also smooth. Let K be large enough, then $S'(S'(S'(n)^{K'})^{K'})^{K'} \leq S(S(S(n)^K)^K)^K \leq 2^{d_{\mathsf{cert}}(\ell(n))}$. It follows that for infinitely many $m \in \mathbb{N}$, L_m^{strong} cannot be $(1/2+1/S(m))$-approximated by circuits of depth

$$C \log S'((m^{1/(d+2)})^{1/3}) - O(\log S(m)) \geq \log S(m).$$

Since the function $\ell(n)^2/d_{\mathsf{cert}}(\ell(n))$ is monotone, we have

$$T(m) = 2^{O(\ell(n)^2/d_{\mathsf{cert}}(\ell(n)))} \leq 2^{O(\ell(m)^2/d_{\mathsf{cert}}(\ell(m)))},$$

thus $L^{\mathsf{strong}} \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]_{/2}$.

We can use enumeration tricks to put L^{strong} into $\mathsf{NTIME}[T(n)]$ and $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]_{/1}$ respectively. □

D.2 Non-trivial CAPP Algorithms Imply Strong Average-case Circuit Lower Bounds

We need the following theorem implicit in [Wil13, Wil16]. We provide a proof in Appendix E.1.

Theorem D.3. *Let $f(n) = \omega(\log n)$ be a good resource function. Suppose* Gap-UNSAT *with gap $1 - 1/n^{10}$ for depth-$f(n)$ circuits can be solved in nondeterministic $2^n/n^{\omega(1)}$ time. Then* NE *can certify $0.99f(n)$ depth.*

Recall that the complexity of a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit is the maximum of its size and the sum of absolute values of its coefficients. Now we are ready to provide a proof sketch of Lemma 6.2, which is the most general version of Theorem 5.1.

Reminder of Lemma 6.2. *Let $\mathscr{C}$ be a typical circuit class. Let $S_{\mathsf{CMD}}, S_{\mathsf{cert}} : \mathbb{N} \to \mathbb{N}$ be good resource functions such that $S_{\mathsf{CMD}}(n) \leq 2^{0.1n}$, and $S_{\mathsf{cert}}(n) = n^{\omega(1)}$. There are universal constants $\delta > 0$ and $K > 4$ such that, if the following hold:*

- CMD *has $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits of complexity $S_{\mathsf{CMD}}(n)$, and*
- *the Average-Product for 4 $\mathscr{C}$ circuits of size $S(n) = S_{\mathsf{CMD}}(S_{\mathsf{cert}}(n)^K)^K$ can be estimated in $T(n) = 2^n/S(n)$ time, within error $1/S(n)$.*

Then NE *can certify $\log S_{\mathsf{cert}}(n)$ depth hardness.*

Proof Sketch. We show that Gap-UNSAT problem with gap $1 - 1/n^{10}$ for circuits of depth $2\log S_{\mathsf{cert}}(n)$ can be solved in $2^n/n^{\omega(1)}$ time. By Theorem D.3, NE can certify $\log S_{\mathsf{cert}(n)}$ depth hardness.

Given a circuit C of depth $2\log S_{\mathsf{cert}}(n)$, where we want to distinguish between C is a tautology, and C has at most $2^n/n^{10}$ satisfying assignments. We apply Theorem 1.16 to C, and obtain a 2-SAT instance F on variable set $Y \cup Z$, where $|Y| = O(n)$ and $|Z| = \ell_{\mathsf{proof}} = \mathrm{poly}(S_{\mathsf{cert}}(n))$. The instance F has $n_{\mathsf{clause}} = \mathrm{poly}(S_{\mathsf{cert}}(n))$ clauses, and there are absolute constants $c_{\mathsf{PCPP}} > s_{\mathsf{PCPP}}$ such that the following hold.

- If C is a tautology, there are depth-$O(\log S_{\mathsf{cert}}(n))$ circuits $\widetilde{T}_1, \widetilde{T}_2, \ldots, \widetilde{T}_{\ell_{\mathsf{proof}}}$ such that for every $x \in \{0,1\}^n$, the assignment $Y = \mathsf{Enc}(x)$ and $Z = \widetilde{\pi}(x) = \widetilde{T}_1(x) \circ \widetilde{T}_2(x) \circ \cdots \circ \widetilde{T}_{\ell_{\mathsf{proof}}}(x)$ satisfies at least a c_{PCPP} fraction of clauses.
- If $C(x) = 0$ for some $x \in \{0,1\}^n$, then $\mathsf{OPT}(F_{Y=\mathsf{Enc}(x)}) \leq s_{\mathsf{PCPP}}$.

Since $\mathsf{NC}^1 \subseteq \oplus\mathsf{L}$ and CMD has $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits of complexity $S_{\mathsf{CMD}}(n)$, if C is a tautology, then each $\widetilde{T}_i$ can also be implemented by a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity $S_{\mathsf{compl}}(n) = S_{\mathsf{CMD}}(S_{\mathsf{cert}}(n)^{K'})$, where K' is a large enough constant.

For each $i \in [\ell_{\mathsf{proof}}]$, we guess a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit T_i of complexity $S_{\mathsf{compl}}(n)$, with the hope that $T_i = \widetilde{T}_i$. Then we perform the following operations.

- First, we verify that each T_i is a valid $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit. Since the complexity of T_i is at most $S_{\mathsf{compl}}(n)$ and we can estimate the Average-Product of 4 $\mathscr{C}$ circuits within error $1/S_{\mathsf{compl}}(n)^5$, we can use Lemma 5.3. For each T_i, the time complexity for this step is $O(T(n) \cdot S_{\mathsf{compl}}(n)^4)$.
- Then we estimate the accept probability of each clause, within error $S_{\mathsf{compl}}(n)^2 \cdot (1/S_{\mathsf{compl}}(n)^5) = o(1)$. For each clause, the time complexity for this step is $O(T(n) \cdot S_{\mathsf{compl}}(n)^2)$.
- After that, we can distinguish between C is a tautology and C has at most $2^n/n^{10}$ satisfying assignments, accordingly.

If C is a tautology and the $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits $\{T_i\}$ we guessed are correct, this algorithm accepts C. If C has at most $2^n/n^{10}$ satisfying assignments, this algorithm rejects C regardless of the circuits $T_1, \ldots, T_{\ell_{\mathsf{proof}}}$ we guessed.

Let K be a large enough constant, $T(n) = 2^n/S_{\mathsf{compl}}(n)^K$. The time complexity of our nondeterministic Gap-UNSAT algorithm is

$$O(n_{\mathsf{clause}} + \ell_{\mathsf{proof}}) \cdot T(n) \cdot S_{\mathsf{compl}}(n)^4 \leq 2^n/n^{\omega(1)}. \qquad \square$$

Finally, we are ready to prove Theorem D.4, which is most general version of Theorem 1.1.

Theorem D.4 (Non-trivial CAPP Algorithms Imply Strong Average-case Circuit Lower Bounds)**.** *Let $\mathscr{C}$ be a typical circuit class such that $\mathscr{C}$ circuits of size S can be implemented by (general) circuits of depth $O(\log S)$. Let $\ell, S, S_{\mathsf{cert}} : \mathbb{N} \to \mathbb{N}$ be good resource functions, and K be a large enough constant depending on ℓ, S.*[33] *Suppose that*

- $S(n) = n^{\omega(1)}$;
- $\ell(n)$, $\log S(n)$ *and* $\log S_{\mathsf{cert}}(n)$ *are smooth;*
- *the function $\ell(n)^2/\log S_{\mathsf{cert}}(\ell(n))$ is nondecreasing; and*
- $S_{\mathsf{cert}}(\ell(n)) \geq S(S(S(n)^K)^K)^K$.

Let

$$S_{\mathsf{CAPP}}(n) = S(S_{\mathsf{cert}}(n)^K)^K, \quad \textit{and} \quad T(n) = \exp\big(O(\ell(n)^2/\log S_{\mathsf{cert}}(\ell(n)))\big).$$

Suppose the CAPP *problem for $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of $S_{\mathsf{CAPP}}(n)$ size can be solved in $2^n/S_{\mathsf{CAPP}}(n)$ time, then $\mathsf{NTIME}[T(n)]$ and $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]_{/1}$ cannot be $(1/2 + 1/S(n))$-approximated by $\mathscr{C}$ circuits of $S(n)$ size.*

Proof. We simply combine Lemma 6.2 and Theorem D.2. Let K' be some large enough constant depending on ℓ, S.

We assume DCMD can be $(1/2 + 1/S(n))$-approximated by $\mathscr{C}$ circuits of $S(n)$ size (otherwise the theorem is trivial). Let δ be the constant in Lemma 6.2, then by Lemma 3.1, CMD has $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$

[33] Similar as in Theorem D.2, this constant K also only depends on the "smoothness" of $\ell(n)$ and $\log S(n)$.

circuits of complexity $S_{\mathsf{CMD}}(n) = S(n)^{K'}$. We also assume that $\mathscr{C}$ circuits of size $S(n)$ can be implemented by formulas of size $S_{\mathsf{NC}}(n) = S(n)^{K'}$.

The problem of estimating the Average-Product for 4 $\mathscr{C}$ circuits of size $S'_{\mathsf{CAPP}}(n) = S_{\mathsf{CMD}}(S_{\mathsf{cert}}(n)^{K'})^{K'}$ within error $1/S'_{\mathsf{CAPP}}(n)$ is equivalent to the CAPP problem for $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $S'_{\mathsf{CAPP}}(n)$. When K is large enough, we have $S'_{\mathsf{CAPP}}(n) \leq S_{\mathsf{CAPP}}(n)$, thus the problem is solvable in $2^n/S'_{\mathsf{CAPP}}(n)$ time. Then by Lemma 6.2, NE can certify $\log S_{\mathsf{cert}}(n)$-depth hardness.

Let $d_{\mathsf{cert}}(n) = \log S_{\mathsf{cert}}(n)$, then $d_{\mathsf{cert}}(n)$ is a smooth function, and the function $\ell(n)^2/d_{\mathsf{cert}}(\ell(n))$ is nondecreasing. For every $n \geq 1$, $S_{\mathsf{NC}}(S_{\mathsf{NC}}(S_{\mathsf{NC}}(n)^{K'})^{K'})^{K'} \leq S_{\mathsf{cert}}(\ell(n))$. By Theorem D.2, $\mathsf{NTIME}[T(n)]$ and $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]_{/1}$ cannot be $(1/2 + 1/S_{\mathsf{NC}}(n))$-approximated by circuits of depth $\log S_{\mathsf{NC}}(n)$, which contains $\mathscr{C}$ circuits of size $S(n)$. □

To conclude this section, we formulate the best NE lower bounds provable by our techniques.

We define a family of *nice* functions f. If f is nice and there exist corresponding CAPP algorithms for $\mathsf{AND}_4 \circ \mathscr{C}$ circuits, then we can use Theorem D.4 to show that NE cannot be $(1/2 + 1/f(n))$-approximated by $\mathscr{C}$ circuits of $f(n)$ size.

Definition D.5. A good resource function $f : \mathbb{N} \to \mathbb{N}$ is *nice* if $f(n) = n^{\omega(1)}$, $\log f(n)$ is smooth, and for every integer $d \geq 1$, the following are true:

- $n/\log f(f(f(n)^d)^d)^d$ is a nondecreasing function, and
- $\log f(f(f(n^2)^d)^d)^d$ is a smooth function.

Reminder of Lemma 6.3. *Let $S : \mathbb{N} \to \mathbb{N}$ be a nice function. If for every constant $d \geq 1$ and $S_{\mathsf{CAPP}}(n) = S(S(S(S(n^2)^d)^d)^d)^d$, the* CAPP *problem for $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $S_{\mathsf{CAPP}}(n)$ can be solved in $2^n/S_{\mathsf{CAPP}}(n)$ time, then $(\mathsf{NE}\cap\mathsf{coNE})_{/1}$ cannot be $(1/2 + 1/S(n))$-approximated by $\mathscr{C}$ circuits of size $S(n)$.*

Proof Sketch. This is immediate from Theorem D.4 by setting $\ell(n) = \lceil\sqrt{n}\rceil$ and $S_{\mathsf{cert}}(n) = S(S(S(n^2)^K)^K)^K$. □

E Missing Proofs

In this section, we provide the missing proofs in the paper.

E.1 Proof of Theorem D.3

We first prove Theorem D.3 (restated below). Note that the proof below essentially follows the same idea of the ACC^0 witness lower bounds of [Wil16].

Reminder of Theorem D.3. *Let $f(n) = \omega(\log n)$ be a good resource function. Suppose* Gap-UNSAT *with gap $1 - 1/n^{10}$ for depth-$f(n)$ circuits can be solved in nondeterministic $2^n/n^{\omega(1)}$ time. Then* NE *can certify $0.99f(n)$ depth.*

Proof. By the nondeterministic time hierarchy theorem [Žák83], there is a unary language $L \in \mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n^{10}]$. We apply the highly efficient PCP system of [BV14] to L, such that the following are true for input 1^n.

- The PCP verifier V_{PCP} uses $\ell(n) = n + O(\log n)$ random bits and receives a proof oracle $\pi : \{0,1\}^{\ell(n)} \to \{0,1\}$.

- V_{PCP} first computes a function $\mathsf{Query} : \{0,1\}^{\ell(n)} \to (\{0,1\}^{\ell(n)})^t$ to map the random bits to the locations of π that we query. Here $t = \mathrm{poly}(n)$ is the number of queries. Moreover, Query is a projection.
- V_{PCP} next computes a function $\mathsf{Dec} : \{0,1\}^t \to \{0,1\}$ to decide whether it accepts the input and the proof. Let $r \in \{0,1\}^{\ell(n)}$ be the randomness and $(q_1, q_2, \ldots, q_t) = \mathsf{Query}(r)$, if $\mathsf{Dec}(\pi(q_1), \pi(q_2), \ldots, \pi(q_t)) = 1$ then V_{PCP} accepts, otherwise it rejects. Moreover, Dec is a 3-CNF (a.k.a. $\mathsf{AND} \circ \mathsf{OR}_3$).
- If $1^n \in L$, then there is some proof $\pi : \{0,1\}^{\ell(n)} \to \{0,1\}$ such that V_{PCP} accepts with probability 1. Otherwise, regardless of the proof, V_{PCP} accepts with probability at most $1/\ell(n)^{10}$ over the randomness.

It follows that L has an (ordinary) NE verifier $V(x, y)$ that works as follows. It treats y (y is assumed to be a string of length $2^{\ell(n)}$) as the proof π for V_{PCP}, enumerates every possible randomness and computes the accept probability of V_{PCP}. It then accepts if this probability is 1 and rejects otherwise. Clearly, V runs in $2^n \mathrm{poly}(n)$ time.

For the sake of contradiction, we assume V has witnesses of $0.99f(\ell(n))$ depth. (That is, the witness y, as a truth table of a function over $\ell(n)$ bits, can be computed by a circuit of $0.99f(\ell(n))$ depth.) The goal is to use the Gap-UNSAT algorithm to speed up V and contradict the nondeterministic time hierarchy.

On the input $x = 1^n$, we first guess a circuit C of depth $0.99f(\ell(n))$ on $\ell(n)$ bits, and hope that the truth table of C is the correct proof π for x. Then we compute the circuit

$$E(r) = \neg\mathsf{Dec}(C(\mathsf{Query}(r)_1), C(\mathsf{Query}(r)_2), \ldots, C(\mathsf{Query}(r)_t)),$$

which has depth $0.99f(\ell(n)) + O(\log n)$ and $\ell(n)$ input bits. Since $0.99f(\ell(n)) + O(\log n) \leq f(\ell(n))$, we can apply the Gap-UNSAT algorithm to E. For a particular randomness $r \in \{0,1\}^{\ell(n)}$, if V_{PCP} accepts r then $E(r) = 0$, otherwise $E(r) = 1$. We then accept x if E has no satisfying assignments, and reject x if E has at least $(1 - 1/\ell(n)^{10}) \cdot 2^{\ell(n)}$ satisfying assignments.

If $x \in L$, then there is a circuit C such that E has no satisfying assignments, and we accept x. If $x \notin L$, then for every circuit C, E has at least $(1 - 1/\ell(n)^{10}) \cdot 2^{\ell(n)}$ satisfying assignments, and we reject x. The whole algorithm runs in $2^{\ell(n)}/\ell(n)^{\omega(1)} < 2^n/n^{10}$ time, contradicting the nondeterministic time hierarchy.

Therefore, the verifier V does not have witnesses of $0.99f(\ell(n))$ depth. Although V requires a proof of length $2^{\ell(n)}$ (rather than 2^n), by a simple padding we can construct a verifier V' that receives a proof of length 2^n, and certifies $0.99f(n)$ depth. □

E.2 Proof of Lemma 5.3

Next we prove Lemma 5.3, which is an adaption of [CW19, Lemma 6.1]. Recall that for a function $f : \{0,1\}^n \to \mathbb{R}$, bin_f is the Boolean function closest to f.

Reminder of Lemma 5.3. *For $S \in \mathbb{N}$, suppose we are given S reals $\{\alpha_i\}_{i\in[S]}$, S $\mathscr{C}$ circuits $\{C_i\}_{i\in[S]}$, and a parameter $\varepsilon < 0.01$. Let $\alpha_{\mathsf{tot}} = \sum_{i\in[S]} |\alpha_i|$ and $\delta = \frac{\varepsilon^2}{2(\alpha_{\mathsf{tot}}+1)^4}$. Suppose the Average-Product of 4 $\mathscr{C}$ circuits on n bits can be estimated within error δ in $T(n)$ time. Let $f = \sum_{i=1}^{S} \alpha_i \cdot C_i$. There is an algorithm A running in $O(T(n) \cdot (S+1)^4)$ time such that:*

- *If $\|f - \mathsf{bin}_f\|_\infty \leq \varepsilon$, then A always accepts;*

- *if* $\|f - \mathsf{bin}_f\|_2 \geq 3\varepsilon$*, then* A *always rejects;*
- *otherwise,* A *can output anything.*

Proof. We define a polynomial $P(z) = z^2(1-z)^2$, and let

$$p = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}[P(f(\mathbf{x}))].$$

In the proof of Lemma 6.1 of [CW19], it is shown that:

- If $\|f - \mathsf{bin}_f\|_\infty \leq \varepsilon$, then $p \leq \varepsilon^2 \cdot (1+0.01)^2$;
- if $\|f - \mathsf{bin}_f\|_2 \geq 3 \cdot \varepsilon$, then $p \geq (3/2)^2 \cdot \varepsilon^2$.

Therefore, it suffices to estimate p within error $\varepsilon^2/2$.

For convenience, we denote the circuit $1 - f$ as $\sum_{i=1}^{S+1} \beta_i \cdot D_i$. That is:

- For $i \in [S]$, $\beta_i = -\alpha_i$, $D_i = C_i$;
- $\beta_{S+1} = 1$, and D_{S+1} is the circuit that outputs the constant 1.

We have

$$\begin{aligned} p &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}\left[\left(\sum_{i=1}^{S}\alpha_i \cdot C_i(\mathbf{x})\right)^2 \cdot \left(\sum_{i=1}^{S+1}\beta_i \cdot D_i(\mathbf{x})\right)^2\right] \\ &= \sum_{i=1}^{S}\sum_{j=1}^{S}\sum_{k=1}^{S+1}\sum_{l=1}^{S+1}(\alpha_i\alpha_j\beta_k\beta_l) \cdot \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}\left[C_i(\mathbf{x})C_j(\mathbf{x})D_k(\mathbf{x})D_l(\mathbf{x})\right]. \end{aligned} \tag{16}$$

For each $1 \leq i, j \leq S$, $1 \leq k, l \leq S+1$, we estimate

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{U}_n}\left[C_i(\mathbf{x})C_j(\mathbf{x})D_k(\mathbf{x})D_l(\mathbf{x})\right]$$

within error $\delta = \frac{\varepsilon^2}{2(\alpha_{\mathsf{tot}}+1)^4}$. Therefore we can estimate (16) within error

$$\sum_{i=1}^{S}\sum_{j=1}^{S}\sum_{k=1}^{S+1}\sum_{l=1}^{S+1}|\alpha_i\alpha_j\beta_k\beta_l| \cdot \delta \leq (\alpha_{\mathsf{tot}}+1)^4 \cdot \delta \leq \varepsilon^2/2.$$

The algorithm runs in $O\big(T(n) \cdot (S+1)^4\big)$ time. □

Remark E.1. It is easy to see that if our Average-Product estimation algorithm is *nondeterministic*, then the algorithm A in Lemma 5.3 is also nondeterministic, and also runs in $O(T(n) \cdot (S+1)^4)$ time.

E.3 Proof of Lemma 6.4

To prove Lemma 6.4, we need the following construction of sets with small pairwise intersections, a.k.a. *designs*.

Lemma E.2 ([Tre01]). *For all integers m, ℓ, and $a \le \ell$, there is a family of m sets $S_1, S_2, \ldots, S_m \subseteq [t]$ (denoted as an (m, t, ℓ, a)-design), such that*

- $t = O(\ell^2 \cdot m^{1/a}/a)$;
- *for every i, $|S_i| = \ell$;*
- *for every $i \neq j$, $|S_i \cap S_j| \le a$.*

Moreover, the family is constructible in deterministic $\mathrm{poly}(m, 2^t)$ *time.*

Now we are ready to prove Lemma 6.4.

Reminder of Lemma 6.4. *Let m, ℓ, a be integers such that $a \le \ell$, and $t = O(\ell^2 \cdot m^{1/a}/a)$. Let $\mathscr{C}$ be a circuit class closed under negation. There is a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that the following hold. For any function $Y : \{0,1\}^\ell \to \{0,1\}$ represented as a length-2^ℓ truth table, if Y cannot be $(1/2 + \varepsilon/m)$-approximated by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits (where the top $\mathscr{C}$ circuit has size S), then $G(Y, \mathcal{U}_t)$ ε-fools every $\mathscr{C}$ circuit (of size S). That is, for any $\mathscr{C}$ circuit C (of size S),*

$$\left| \Pr_{\mathbf{s} \sim \mathcal{U}_t}[C(G(Y, \mathbf{s})) = 1] - \Pr_{\mathbf{x} \sim \mathcal{U}_m}[C(\mathbf{x}) = 1] \right| \le \varepsilon.$$

Moreover, the function G is computable in $\mathrm{poly}(m, 2^t)$ *time.*

Proof. Let $S_1, S_2, \ldots, S_m$ be an (m, t, ℓ, a)-design specified in Lemma E.2, the PRG is defined as

$$G(Y, w) = Y(w|_{S_1}) \circ Y(w|_{S_2}) \circ \cdots \circ Y(w|_{S_m}),$$

where $w|_S$ is the $|S|$-bit string obtained by taking the bits in w with indices in S.

Suppose some $\mathscr{C}$ circuit C distinguishes $G(Y, \mathcal{U}_t)$ between $\mathcal{U}_m$ with advantage $\ge \varepsilon$. Let $\mathbf{w} \sim \mathcal{U}_t$. A standard hybrid argument implies that there is some $1 \le i \le m$ such that C distinguishes the following two distributions with advantage $\ge \varepsilon/m$:

$$\begin{aligned} \mathbf{D}_{i-1} &= Y(\mathbf{w}|_{S_1}) \circ Y(\mathbf{w}|_{S_2}) \circ \cdots \circ Y(\mathbf{w}|_{S_{i-1}}) \circ \mathcal{U}_{m-i+1}, \text{ and} \\ \mathbf{D}_i &= Y(\mathbf{w}|_{S_1}) \circ Y(\mathbf{w}|_{S_2}) \circ \cdots \circ Y(\mathbf{w}|_{S_i}) \circ \mathcal{U}_{m-i}. \end{aligned}$$

Since $\mathscr{C}$ is closed under negations, we may assume $\Pr[C(\mathbf{D}_{i-1}) = 1] \ge \Pr[C(\mathbf{D}_i) = 1] + \varepsilon/m$.

We construct a randomized $\mathscr{C} \circ \mathsf{Junta}_a$ circuit $\mathbf{C}'$ that $(1/2 + \varepsilon/m)$-approximates Y, contradicting the hardness of Y. Given a random input $\mathbf{x} \sim \mathcal{U}_\ell$, we fix a random seed $\mathbf{w}$ as follows. We let $\mathbf{w}|_{S_i} = \mathbf{x}$ and the other bits of $\mathbf{w}$ are independent and uniform random bits. It is easy to see that the distribution of $\mathbf{w}$ is exactly $\mathcal{U}_t$. We also choose $\mathbf{z} \sim \mathcal{U}_{m-i+1}$, to form an input

$$\mathbf{input} = Y(\mathbf{w}|_{S_1}) \circ Y(\mathbf{w}|_{S_2}) \circ \cdots \circ Y(\mathbf{w}|_{S_{i-1}}) \circ \mathbf{z}.$$

Then we let $\mathbf{C}'(\mathbf{x}) = C(\mathbf{input}) \oplus \mathbf{z}_i$.

We show that $\mathbf{C}'$ is correct w.p. $\ge 1/2 + \varepsilon/m$, where the probability space is over the random input $\mathbf{x}$ and the internal randomness of $\mathbf{C}'$. Let

$$p_{\mathsf{right}} = \Pr[C(\mathbf{input}) = 1 \mid \mathbf{z}_i = Y(\mathbf{x})], \quad \text{and} \quad p_{\mathsf{wrong}} = \Pr[C(\mathbf{input}) = 1 \mid \mathbf{z}_i \neq Y(\mathbf{x})].$$

Then

$$\Pr[C(\mathbf{D}_i) = 1] = p_{\mathsf{right}}, \quad \text{and} \quad \Pr[C(\mathbf{D}_{i-1}) = 1] = \frac{1}{2}(p_{\mathsf{wrong}} + p_{\mathsf{right}}),$$

and

$$\begin{aligned}\Pr[\mathbf{C}'(\mathbf{x}) = Y(\mathbf{x})] =& \frac{1}{2}p_{\mathsf{wrong}} + \frac{1}{2}(1 - p_{\mathsf{right}}) \\ =& \frac{1}{2} + \Pr[C(\mathbf{D}_{i-1}) = 1] - \Pr[C(\mathbf{D}_i) = 1] \\ \geq& \frac{1}{2} + \varepsilon/m.\end{aligned}$$

By averaging, we can fix the internal randomness of $\mathbf{C}'$ to obtain a deterministic circuit C' that $(1/2 + \varepsilon/m)$-approximates Y. Since for each $j < i$, $|S_j \cap S_i| \leq a$, each bit of **input** depends on at most a bits in $\mathbf{x}$. It follows that C' is a valid $\mathscr{C} \circ \mathsf{Junta}_a$ circuit, contradicting the hardness of Y. □

E.4 Proof of Theorem 7.4

We present the #SAT algorithm for PTF gates in [BKK+19, Section 3], and verify that it also works for the AND of 4 PTF gates.

First, we need to define *linear decision trees*, which is crucial to the algorithm.

Definition E.3. A *linear decision tree* (over n variables $z_1, z_2, \ldots, z_n \in \mathbb{Z}$) is a decision tree such that the following hold.

- Every leaf node is labeled with an output (in some output space).
- Every nonleaf node is labeled with a linear inequality $\sum_{i=1}^{n} w_i z_i \geq \theta$, where w_i and θ are parameters of the node. Furthermore, every nonleaf node has a *Yes* child and a *No* child.
- Given an input $\vec{z} = (z_1, z_2, \ldots, z_n) \in \mathbb{Z}^n$, we start at the root and proceed until we reach a leaf node. At every nonleaf node, if the linear inequality labeled on the node is satisfied, we go to its *Yes* child, otherwise we go to its *No* child. At a leaf node, we output the label of that node and stop.

Given a linear decision tree T and an input $\vec{z}$, we denote $T(\vec{z})$ as the output (*i.e.* label of the visited leaf) of the query algorithm.

We need the following lemma from [BKK+19, KLMZ17], which constructs linear decision trees computing the number of common satisfying assignments of 4 PTF gates.

Lemma E.4. *Let k be an integer, $r = \sum_{i=0}^{k} \binom{m}{i}$ be the number of different monomials over m variables of degree $\leq k$. There is a randomized algorithm that on input m, produces a (randomized) linear decision tree $\mathbf{T}$, such that*

- *The input to $\mathbf{T}$ is the coefficients of 4 polynomials (i.e. $4r$ numbers).*
- *The coefficients (i.e. w_i in Definition E.3) in each nonleaf node are in $\{-2, -1, 0, 1, 2\}$.*
- *The decision tree depth is $\Delta = O(m^{k+1} \log m)$.*
- *For every m-variate polynomials $P_1, \ldots, P_4$ of degree k, let $\vec{z}$ be the list of coefficients of the 4 polynomials. Then $\mathbf{T}(\vec{z})$ is either the symbol ?, or the number of assignments $x \in \{0,1\}^m$ that makes the values of all these 4 polynomials nonnegative. Furthermore, $\Pr_{\mathbf{T}}[\mathbf{T}(\vec{z}) = ?] \leq 1/2$ (over the internal randomness of the algorithm).*

The randomized algorithm runs in time $2^{O(\Delta)}$.

Reminder of Theorem 7.4. *For every parameter* $k = k(n)$*, there is a* ZPP *algorithm that counts the number of satisfying assignments of any* $\mathsf{AND}_4 \circ \mathsf{PTF}_k$ *circuit over* n *variables in* $2^{n-m} \cdot \mathrm{poly}(n)$ *time, where* $m = n^{1/(k+1)} / \log n$*.*

Proof Sketch. Let $m = n^{1/(k+1)} / \log n$. We use $n_1 = 10n$ independent runs of Lemma E.4 on input m, and produce n_1 random linear decision trees $T_1, T_2, \ldots, T_{n_1}$.

Let $P_1, \ldots, P_4$ be 4 input polynomials. We enumerate all possible assignments of the first $n - m$ variables. For each such assignment and each $1 \leq i \leq 4$, we can restrict the polynomial P_i to a degree-k m-variate polynomial $\widetilde{P}_i$. We then use the previously constructed decision trees to compute the number of assignments $x_{n-m+1}, \ldots, x_n$, such that all of $\widetilde{P}_1, \ldots, \widetilde{P}_4$ are nonnegative. In particular, if any of the n_1 decision trees outputs a value other than ?, this value is the desired number, and we then add this number to our answer. If the output of every decision tree is ?, the algorithm fails.

By union bound, the probability that the algorithm fails is at most

$$2^{n-m} \cdot 2^{-n_1} \ll 1/2.$$

Since $m^{k+1} \log m \leq O(n / \log^k n)$, the time complexity of the algorithm is

$$O\Big(n_1 \cdot 2^{O(m^{k+1} \log m)} + 2^{n-m} \cdot m^{k+1} \log m \cdot \mathrm{poly}(n)\Big) = 2^{n-m} \cdot \mathrm{poly}(n). \qquad \square$$

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AC19] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1034–1055. IEEE Computer Society, 2019.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[Ajt83] M Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[App14] Benny Applebaum. *Cryptography in Constant Parallel Time*. Information Security and Cryptography. Springer, 2014.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1):2:1–2:54, 2009.

[Bab87] László Babai. Random oracles separate PSPACE from the polynomial-time hierarchy. *Inf. Process. Lett.*, 26(1):51–53, 1987.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =?NP question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BHLR19] Abhishek Bhrushundi, Kaave Hosseini, Shachar Lovett, and Sankeerth Rao. Torus polynomials: An algebraic approach to ACC lower bounds. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 13:1–13:16, 2019.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012.

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: - new simple PRF candidates and their applications. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, pages 699–729, 2018.

[BKK+19] Swapnam Bajpai, Vaibhav Krishan, Deepanshu Kush, Nutan Limaye, and Srikanth Srinivasan. A #SAT algorithm for small constant-depth circuits with PTF gates. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 8:1–8:20, 2019.

[BKT18] Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: tight quantum query bounds via dual polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 297–310, 2018.

[BT17] Mark Bun and Justin Thaler. A nearly optimal lower bound on the approximate degree of AC^0. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 1–12, 2017.

[BV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 163–173, 2014.

[Che19] Lijie Chen. Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1281–1304. IEEE Computer Society, 2019.

[CHLT19] Eshan Chattopadhyay, Pooya Hatami, Shachar Lovett, and Avishay Tal. Pseudorandom generators from the second fourier level and applications to AC^0 with parity gates. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 22:1–22:15, 2019.

[COS18] Ruiwen Chen, Igor Carboni Oliveira, and Rahul Santhanam. An average-case lower bound against ACC^0. In *LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings*, pages 317–330, 2018.

[CW19] Lijie Chen and R. Ryan Williams. Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity. In Amir Shpilka, editor, *34th Computational Complexity Conference (CCC 2019)*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:43, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 2–12. IEEE Computer Society, 1991.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 316–324, 2004.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[FSUV13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013.

[GGH+07] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 440–449, 2007.

[GGH+08] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. A (de)constructive approach to program checking. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 143–152. ACM, 2008.

[GHR92] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. Majority gates VS. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32, 1989.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GR08] Dan Gutfreund and Guy N. Rothblum. The complexity of local list decoding. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques,*

11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings, pages 455–468, 2008.

[GSV18] Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 956–966, 2018.

[HAB02] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.

[Har04] Prahladh Harsha. *Robust PCPs of proximity and shorter PCPs*. PhD thesis, Massachusetts Institute of Technology, 2004.

[Hås89] Johan Håstad. Almost optimal lower bounds for small depth circuits. *Advances in Computing Research*, 5:143–170, 1989.

[HMP+93] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *J. Comput. Syst. Sci.*, 46(2):129–154, 1993.

[HP10] Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Exact threshold circuits. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*, pages 270–279, 2010.

[HS74] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems (ISTCS '97)*, ISTCS '97, pages 174–, Washington, DC, USA, 1997. IEEE Computer Society.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[Kan12] Daniel M. Kane. A structure theorem for poorly anticoncentrated gaussian chaoses and applications to the study of polynomial threshold functions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 91–100, 2012.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31, 1988.

[KKL17] Valentine Kabanets, Daniel M. Kane, and Zhenjian Lu. A polynomial restriction lemma with applications. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 615–628, 2017.

[KL18] Valentine Kabanets and Zhenjian Lu. Satisfiability and Derandomization for Small Polynomial Threshold Circuits. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:19, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[KLMZ17] Daniel M. Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 355–366, 2017.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[Mur71] Saburo Muroga. *Threshold logic and its applications*. Wiley, 1971.

[MW18] Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901, 2018.

[MZ13] Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM J. Comput.*, 42(3):1275–1301, 2013.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Raz87] Alexander A Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[San09] Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.

[SFM78] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[She18] Alexander A. Sherstov. Algorithmic polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 311–324, 2018.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82, 1987.

[Smo93] Roman Smolensky. On representations by low-degree polynomials. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 130–138, 1993.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005.

[SV10] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

[Vio09a] Emanuele Viola. On approximate majority and probabilistic time. *Computational Complexity*, 18(3):337–375, 2009.

[Vio09b] Emanuele Viola. On the power of small-depth computation. *Foundations and Trends® in Theoretical Computer Science*, 5(1):1–72, 2009.

[VW20] Nikil Vyas and Ryan Williams. Lower bounds against sparse symmetric functions of ACC circuits: Expanding the reach of #SAT algorithms. 2020. To appear in STACS 2020.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[Wil14a] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 194–202, 2014.

[Wil14b] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

[Wil16] R. Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016.

[Wil18] Richard Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, ReLUs, and low-degree polynomials. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 6:1–6:24, 2018.

[Yao82] Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 1–10, 1985.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

[Žák83] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il