Electronic Colloquium on Computational Complexity, Report No. 150 (2020)

Almost-Everywhere Circuit Lower Bounds from Non-Trivial Derandomization

Lijie Chen
MIT
lijieche@mit.edu

Xin Lyu
Tsinghua University
lvx17@mails.tsinghua.edu.cn

R. Ryan Williams
MIT
rrw@mit.edu

Abstract

In certain complexity-theoretic settings, it is notoriously difficult to prove complexity separations which hold *almost everywhere*, i.e., for all but finitely many input lengths. For example, a classical open question is whether $\mathsf{NEXP} \subset$ i.o.-NP; that is, it is open whether nondeterministic exponential time computations can be simulated on infinitely many input lengths by NP algorithms. This difficulty also applies to Williams' algorithmic method for circuit lower bounds [Williams, J. ACM 2014]. In particular, although [Murray and Williams, STOC 2018] proved $\mathsf{NTIME}[2^{\mathrm{polylog}(n)}] \not\subset \mathsf{ACC}^0$, it has remained an open problem to show that E^{NP} ($2^{O(n)}$ time with an NP oracle) is not contained in i.o.-ACC^0.

In this paper, we show how many infinitely-often circuit lower bounds proved by the algorithmic method can be adapted to establish almost-everywhere lower bounds.

- We show there is a function $f \in \mathsf{E}^{\mathsf{NP}}$ such that *for all sufficiently large input lengths* n and $\varepsilon \leq o(1)$, f cannot be $(1/2 + 2^{-n^{\varepsilon}})$-approximated by $2^{n^{\varepsilon}}$-size ACC^0 circuits on inputs of length n, improving lower bounds in [Chen and Ren, STOC 2020] and [Viola, ECCC 2020].
- We construct rigid matrices in P^{NP} for all but finitely many inputs, rather than infinitely often as in [Alman and Chen, FOCS 2019] and [Bhangale et al., FOCS 2020].
- We show there are functions in E^{NP} requiring constant-error probabilistic degree at least $\Omega(n/\log^2 n)$ for all large enough n, improving an infinitely-often separation of [Viola, ECCC 2020].

Our key to proving almost-everywhere worst-case lower bounds is a new "constructive" proof of an NTIME hierarchy theorem proved by [Fortnow and Santhanam, CCC 2016], where we show for every "weak" nondeterminstic algorithm (with smaller running-time and short witness), a "refuter algorithm" exists that can construct "bad" inputs for the hard language. We use this refuter algorithm to construct an almost-everywhere hard function. To extend our lower bounds to the average case, we prove a new XOR Lemma based on approximate linear sums, and combine it with the PCP-of-proximity applications developed in [Chen and Williams, CCC 2019] and [Chen and Ren, STOC 2020]. As a byproduct of our new XOR Lemma, we obtain a nondeterministic pseudorandom generator for poly-size ACC^0 circuits with seed length $\mathrm{polylog}(n)$, which resolves an open question in [Chen and Ren, STOC 2020].

ISSN 1433-8092

1 Introduction

Proving unconditional circuit lower bounds for explicit functions (with the flagship problem of $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$) is one of the central problems in theoretical computer science. In the 1980s, considerable progress was made in proving lower bounds for constant-depth circuits, as first steps towards lower bounds for general circuits. The classical works [Ajt83, FSS84, Yao85, Hås89] culminated in exponential lower bounds for AC^0 (constant depth circuits consisting of unbounded fan-in AND/OR gates). The works [Raz87, Smo87] established exponential lower bounds for $\mathsf{AC}^0[q]$ (AC^0 circuits with MOD_q gates) for prime power q.

Unfortunately, the progress in the 1980s did not go much further: lower bounds for $\mathsf{AC}^0[m]$ have been extremely difficult to establish for composite m, although it has been conjectured that $\mathsf{AC}^0[m]$ cannot compute the Majority function. In fact, it was a notorious open question whether NEXP (nondeterministic exponential time) has polynomial-size ACC^0 circuits.[1] Several years ago, Williams [Wil14] finally proved such a lower bound, via an *algorithmic* approach to circuit lower bounds [Wil13]. Combining many results from classical complexity, such as the nondeterministic time hierarchy theorem [SFM78, Žák83], hardness vs randomness [NW94], and the PCP Theorem [ALM+98, AS98], Williams' work shows how nontrivial circuit-analysis algorithms can be generically applied to prove circuit lower bounds.

Developments after $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$. The separation $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ had several drawbacks compared to the classical lower bounds of the 80s. The most significant drawback was that NEXP is a much larger class than our ultimate goal NP (previous lower bounds for AC^0 or $\mathsf{AC}^0[p]$ usually work for functions in P). Murray and Williams improved this state of affairs significantly by showing $\mathsf{NQP} := \mathsf{NTIME}[2^{\mathrm{polylog}(n)}]$ is not contained in ACC^0 [MW20].[2]

Another drawback is that the algorithmic approach [Wil14, MW20] only yielded worst-case lower bounds, while prior lower bounds for AC^0 or $\mathsf{AC}^0[p]$ can often be adapted to hold in the average case (e.g., [HRST17]). A line of recent work [COS18, CW19, Che19, CR20] generalizes the algorithmic approach to the average-case setting, culminating in the result that NQP cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by (polynomial-size) ACC^0 circuits [CR20].

The Infinitely-Often Separation Drawback. All the aforementioned developments significantly expand the reach of the algorithmic method. However, there has remained a subtle but important drawback of the algorithmic method: it only achieves infinitely-often separations. For example, [MW20] shows there is an NQP function f such that, for every polynomial-size ACC^0 circuit family $\{C_n\}$, *there are infinitely many input lengths n such that C_n fails to compute f on n-bit inputs*. This certainly implies the separation $\mathsf{NQP} \not\subset \mathsf{ACC}^0$, but it could be the case that *for nearly every input length* NQP is easy for ACC^0, and NQP is only hard on extremely rare input lengths n, e.g., $n = 2^{2^{2^k}}$ for $k \in \mathbb{N}$. In a case where the hard input lengths are so far apart, *practically* the situation is not very different from $\mathsf{NQP} \subset \mathsf{ACC}^0$. In fact, it has remained open whether E^{NP} is contained *infinitely-often* in ACC^0.

The Infinitely-Often Barrier in Complexity Theory. Ideally, we desire *almost-everywhere separations*: we want a function $f = \{f_n : \{0,1\}^n \to \{0,1\}\}$ so that for all sufficiently large input lengths n, f_n cannot be computed by any ACC^0 circuit (in notation, we would say $f \notin$ i.o.-ACC^0). Most previous lower bounds for AC^0 and $\mathsf{AC}^0[p]$ are almost-everywhere: they show $f \notin$ i.o.-AC^0 or $f \notin$ i.o.-$\mathsf{AC}^0[p]$ for some f. Indeed, most combinatorial/algebraic lower bound approaches argue hardness for each input length separately, so they naturally give lower bounds for all input lengths. However, in structural complexity theory, arguments often involve different input lengths simultaneously, and it is common that in some settings

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

[2] Note that there is another notion of NQP [ADH97] (Nondeterministic Quantum Polynomial-Time) in the literature based on quantum complexity, but it turns out to equal $\mathsf{coC}_=\mathsf{P}$ [FGHP99].

almost-everywhere separations are much harder to achieve than corresponding infinitely-often separations. Two classical examples include:

- (An Almost-Everywhere NTIME Hiearchy is Open.) It is known that $\mathsf{NTIME}[2^n] \not\subset \mathsf{NTIME}[2^n/n]$ [SFM78, Žák83], but it is open whether $\mathsf{NTIME}[2^n] \subset \text{i.o.-}\mathsf{NTIME}[n \log n]$. (Indeed, there is an oracle O such that $\mathsf{NEXP}^O \subset \text{i.o.-}\mathsf{NP}^O$ [BFS09].)
- (An Almost-Everywhere Super-Linear Circuit Lower Bound for $\mathsf{MATIME}[2^n]$ is Open.) It is known that $\mathsf{MA}_{/1} \not\subset \mathsf{SIZE}(n^k)$ for all k and $\mathsf{MATIME}[2^n] \not\subset \mathsf{P}_{/\mathsf{poly}}$, but it is open whether $\mathsf{MATIME}[2^n] \subset \text{i.o.-}\mathsf{SIZE}(O(n))$. (Indeed, it is even open whether $\Sigma_2\mathsf{TIME}[2^n] \subset \text{i.o.-}\mathsf{SIZE}(O(n))$).

Other examples include fixed-polynomial lower bounds for the complexity classes $\mathsf{NP}^{\mathsf{NP}} = \Sigma_2\mathsf{P}$ [Kan82], $\mathsf{ZPP}^{\mathsf{NP}}$ [BCG+96, KW98], $\mathsf{S}_2\mathsf{P}$ [Cai07, CCHO05], PP [Vin05, Aar06], time-space trade-off for solving SAT [FLvMV05, Wil07], and hierarchy theorems such as [Bar02, FS04, vMP06]. All of these lower bounds only provide an infinitely-often separation, and it is open to prove an almost-everywhere separation. There are also interesting algorithmic results motivated by complexity concerns, which are only guaranteed to work for infinitely many input lengths (*e.g.*, [Kab01, Wil16, OS17]).

1.1 Our Results

In this work, we achieve almost-everywhere circuit lower bounds with the algorithmic approach. To formally discuss our results, we briefly recall two circuit-analysis problems.

1. CAPP**:** Given a circuit C of size S, estimate the probability that C accepts a uniformly random input within an additive error of $1/S$.
2. Gap-UNSAT**:** Given a circuit C, distinguish between the case that C is unsatisfiable and the case that C has at least $2^n/3$ satisfying assignments.

1.1.1 Almost-Everywhere Circuit Lower Bounds From Non-Trivial Derandomization

Our first result is that "non-trivial derandomization" for a circuit class $\mathscr{C}$ implies almost-everywhere $\mathscr{C}$-circuit lower bounds for E^{NP}. In the following, we say that a circuit class $\mathscr{C}$ is *typical* if $\mathscr{C}$ is closed under projections and negations. (See Section 3 for a formal definition.)

Theorem 1.1. *There are universal constants $\varepsilon \in (0,1)$, $K \geq 1$ satisfying the following. Let $\mathscr{C}$ be typical, and let $s(n)$ be a non-decreasing time-constructible function with $n \leq s(n) \leq 2^{\varepsilon n}$ for all n. If* Gap-UNSAT *on* $\mathsf{AND} \circ \mathsf{OR} \circ \mathscr{C}$ *circuits of size $s(n)^K$ can be solved deterministically in $2^n/n^{\omega(1)}$ time, then there are functions in E^{NP} that do not have $\mathscr{C}$ circuits of size $s(n/2)$, for every sufficiently large n.*

An Extension to Average-Case Lower Bounds. Combining PCPs of Proximity and a new XOR Lemma (see Section 1.2.2), we can extend the above theorem to prove strong average-case lower bounds. We say that a function $f\colon \{0,1\}^n \to \{0,1\}$ *cannot be $(1/2+\varepsilon)$-approximated* by circuits of type $\mathcal{C}$, if every circuit from $\mathcal{C}$ computes f correctly on less than $(1/2+\varepsilon)2^n$ of the n-bit inputs. For a language $L\colon \{0,1\}^\star \to \{0,1\}$, we use $L_n\colon \{0,1\}^n \to \{0,1\}$ to denote its restriction to n-bit inputs.

Theorem 1.2. *Let $\mathscr{C}$ be typical. Suppose there is an $\varepsilon > 0$ such that* CAPP *of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a language $L \in \mathsf{E}^{\mathsf{NP}}$ and a constant $\delta > 0$ such that, for every sufficiently large n, L_n cannot be $(1/2+2^{-n^\delta})$-approximated by $\mathscr{C}$ circuits of size 2^{n^δ}.*

The above results have several applications to complexity lower bounds and pseudorandom generators. We will discuss them separately.

1.1.2 Applications in Complexity Lower Bounds

Almost-Everywhere Strong Average-Case Exponential Lower Bounds for $\mathsf{ACC}^0 \circ \mathsf{THR}$.[3] Combining Theorem 1.2 and the corresponding #SAT algorithm from [Wil18a] for $\mathsf{ACC}^0 \circ \mathsf{THR}$, the almost-everywhere strongly average-case lower bound for E^{NP} against $\mathsf{ACC}^0 \circ \mathsf{THR}$ follows immediately.

Recall that for a given $d, m \in \mathbb{N}$, $\mathsf{AC}^0_d[m]$ is the class of circuit families of depth d, with unbounded fan-in $\mathsf{AND}, \mathsf{OR}, \mathsf{MOD}_m$ gates, and $\mathsf{ACC}^0 := \bigcup_{d,m} \mathsf{AC}^0_d[m]$.

Corollary 1.3. *For every $d, m \in \mathbb{N}$, there is an $\varepsilon = \varepsilon_{d,m}$ and a language $L \in \mathsf{E}^{\mathsf{NP}}$ such that L_n cannot be $(1/2 + 2^{-n^{\varepsilon}})$-approximated by $\mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuits of $2^{n^{\varepsilon}}$ size, for every sufficiently large n.*

Corollary 1.3 compares favorably with prior circuit lower bounds for problems in E^{NP}. Williams [Wil14, Wil18a] proved that E^{NP} cannot be worst-case computed by $2^{n^{o(1)}}$ size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. Following the work of Rajgopal, Santhanam and Srinivasan [RSS18], Viola [Vio20] recently proved E^{NP} cannot be $(1/2 + 1/n^{1-\varepsilon})$-approximated by $2^{n^{o(1)}}$-size $\mathsf{AC}^0[\oplus]$ circuits. Chen and Ren [CR20] recently proved that E^{NP} cannot be $(1/2 + g(n)^{-1})$-approximated by $g(n)$-size ACC^0 circuits, where g is any sub-half-exponential function.[4] All of these lower bounds are only infinitely-often separations, and yield strictly weaker average-case lower bounds than Corollary 1.3.

We also remark that [FS17] devised a notion of "significant separation", which is stronger than infinitely-often separation while weaker than almost-everywhere separation.[5] They showed a significant separation of NEXP and ACC^0. This is incomparable with almost-everywhere separation for E^{NP}.

Almost-Everywhere Construction of Rigid Matrices with an NP Oracle. The problem of efficiently constructing *rigid matrices* is a longstanding open problem in complexity theory [Val77, Lok09].

Definition 1.4. Let $\mathbb{F}$ be a field. For $r, n \in \mathbb{N}$ and a matrix $M \in \mathbb{F}^{n \times n}$, the r-rigidity of M, denoted as $\mathcal{R}_M(r)$, is the minimum Hamming distance between M and a matrix of rank at most r.

Alman and Chen [AC19] recently showed that rigid matrices over the field $\mathbb{F}_2$ (similar results hold for all fields of constant order) with interesting parameters (considered by [Raz89] for connections to communication complexity) could be constructed *infinitely often* in P^{NP} via the algorithmic method. Their proof has been simplified and improved by Bhangale *et al.* [BHPT20]. Formally, [BHPT20] construct a P^{NP} algorithm M which, for infinitely many n, $M(1^n)$ outputs a matrix H_n such that $\mathcal{R}_{H_n}(2^{\log^{1-\varepsilon} n}) \geq \delta n^2$ over $\mathbb{F}_2$.[6]

Applying similar ideas from the proof of Theorem 1.1 and Theorem 1.2, we can strengthen their construction to an almost-everywhere one.

Theorem 1.5. *There is a $\delta > 0$ such that, for all finite fields $\mathbb{F}$ and $\varepsilon > 0$, there is a P^{NP} algorithm which on input 1^n outputs an $n \times n$ matrix H satisfying $\mathcal{R}_H(2^{\log^{1-\varepsilon} n}) \geq \delta n^2$ over $\mathbb{F}$, for all large enough n.*

Almost-Everywhere Probabilistic Degree Lower Bounds. The notion of probabilistic degree for Boolean functions has been studied extensively for decades. Let us recall the definition.

[3] $\mathsf{ACC}^0 \circ \mathsf{THR}$ denotes the class of constant-depth circuits comprised of AND, OR and MOD_m gates (for a constant $m > 1$), with a bottom layer of gates computing arbitrary linear threshold functions.

[4] We say that g is *sub-half-exponential* if $g(g(n)) = 2^{n^{o(1)}}$.

[5] Roughly speaking, "significant separation" means that when the separation holds for an input length n, there is another input length at most polynomially larger than n such that the separation also holds. That is, the hardness cannot be very "sparsely distributed".

[6] [BHPT20] (and also [AC19]) indeed achieved a *nondeterministic construction*. That is, there is a nondeterministic algorithm M such that for infinitely many n: (1) On a computational path, $M(1^n)$ either outputs a valid rigid matrix or "failed". (2) There exists at least one computational path that $M(1^n)$ outputs a valid rigid matrix. In this paper we focus on P^{NP} constructions since they give a *single rigid matrix*.

Definition 1.6. The *ε-error probabilistic degree* of a function $f\colon \{0,1\}^n \to \{0,1\}$ is the minimum d such that there is a distribution $\mathcal{D}$ on $\mathbb{F}_2$-polynomials of degree at most d such that $\Pr_{P\sim\mathcal{D}}[P(x) \neq f(x)] \leq \varepsilon$. When ε is not specified, it is assumed to be $1/3$ by default.

Very recently, Viola [Vio20] proved an $\Omega(n/\log^2 n)$ probabilistic degree lower bound for E^{NP} using the algorithmic method. We extend his result to the almost-everywhere case.

Theorem 1.7. *There is a language $L\colon \{0,1\}^\star \to \{0,1\}$ in E^{NP} such that L_n has $1/3$-error probabilistic degree at least $\Omega(n/\log^2 n)$, for every sufficiently large n.*

Almost-Everywhere Exponential Correlation Bounds against $n^{1/2-\delta}$-Degree $\mathbb{F}_2$-Polynomials. Combining the proof technique of the main theorem and an improved XOR Lemma (introduced in the next subsection), we can also prove a strong inapproximability result for low-degree polynomials for a problem in E^{NP}.

Theorem 1.8. *For all $\delta > 0$, there is a language $L\colon \{0,1\}^\star \to \{0,1\}$ in E^{NP} such that L_n cannot be $(1/2 + 2^{-n^{\Omega(1)}})$-approximated by $n^{1/2-\delta}$ degree $\mathbb{F}_2$-polynomials, for every sufficiently large n.*

The previous best known correlation bound against $n^{1/2-\delta}$-degree $\mathbb{F}_2$-polynomials was only $1/2 + n^{-\delta}$ for the Majority function [Raz87, Smo87, Smo93], and this degree/approximation tradeoff is indeed tight for Majority [Vio19].

1.1.3 Applications to Pseudorandom Generators

Following the known connection between average-case hardness and construction of PRGs [NW94], we obtain two different constructions of PRGs for ACC^0, both with the near-optimal $\mathrm{polylog}(n)$ seed length. Our first PRG works for all input lengths, while is only computable in E^{NP}. Our second construction obtains a *nondeterministic pseudorandom generator* (NPRG) (see Section 3.3 for a formal definition), which is a weaker class compared to E^{NP}-computable PRGs. But the NPRG only works for infinitely many input lengths.

The following E^{NP}-computable PRG is a direct consequence of our average-case lower bound for ACC^0 (Corollary 1.3).

Theorem 1.9. *For all constants d and m, there is $\delta = \delta(d,m) > 0$ and an E^{NP}-computable PRG which, takes an n-bit seed and outputs a 2^{n^δ}-bit string, fooling $\mathsf{AC}^0_d[m]$ circuits of size 2^{n^δ}.*

Our proof technique can also be used to construct an infinitely-often NPRG against ACC^0 circuits with $\mathrm{polylog}(n)$ seed length. This significantly improves upon the previous work by Chen and Ren [CR20], and answers one of their open questions.

Theorem 1.10. *For all constants d and m, there is $\delta = \delta(d,m) > 0$ and an i.o.-NPRG which takes n-bit seeds, runs in $2^{O(n)}$ time, and outputs 2^{n^δ}-bit strings fooling $\mathsf{AC}^0_d[m]$ circuits of size 2^{n^δ}.*

Remark 1.11. We remark that our NPRG and E^{NP}-computable PRG also work for other circuit classes $\mathscr{C}$, given non-trivial CAPP algorithms for slightly larger circuit classes $\mathscr{C}'$. See Section 6 and Section 7 for the details.

1.2 Two Technical Tools

To achieve our almost-everywhere strongly average-case lower bounds, we develop two new technical tools. The first is a "constructive" proof of the almost-everywhere sublinear witness NTIME hierarchy of Fortnow

and Santhanam [FS16] which builds a P^{NP} algorithm that can explicitly find inputs on which the weak algorithms make mistakes. The second is an XOR Lemma based on computations by approximate linear sums. We believe both results are interesting in their own right, and will likely have other applications in computational complexity. In the following we state both of them informally. Check Section 2 for a more in-depth discussion on these two new tools, and why they come up naturally in our lower bound proofs.

1.2.1 An Almost-Everywhere (Sublinear Witness) NTIME Hierarchy with Refuter

A critical piece of Williams' proof that $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ (and later work) is the NTIME hierarchy [SFM78, Žák83]. However, as mentioned earlier, that hierarchy is only known to hold infinitely-often; consequently, the resulting circuit lower bounds fail to be almost-everywhere, and extending the NTIME hierarchy to hold almost-everywhere is notoriously open.

Nevertheless, Fortnow and Santhanam [FS16] managed to prove an almost-everywhere NTIME hierarchy for a restricted subclass of NTIME, where the "weak" nondeterministic machines (being diagonalized against) use witnesses of length less than n bits. Let $\mathsf{NTIMEGUESS}[t(n), g(n)]$ be the class of languages decided by nondeterministic algorithms running in $O(t(n))$ steps and guessing at most $g(n)$ bits. Fortnow and Santhanam proved there is a language L in nondeterministic $O(T(n))$ time that is not decidable, even infinitely-often, by nondeterministic $o(T(n))$-time $n/10$-guess machines:

Theorem 1.12. *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, $\mathsf{NTIME}[T(n)] \not\subset$ i.o.-$\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.*

Our most important new ingredient is the construction of a "refuter" for the hierarchy of Theorem 1.12: an algorithm with an NP oracle which can efficiently find bad inputs for any $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ machine.

Theorem 1.13 (Refuter with an NP Oracle, Informal)**.** *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, there is a language $L \in \mathsf{NTIME}[T(n)]$ and an algorithm $\mathcal{R}$ such that:*

1. ***Input.*** *The input to $\mathcal{R}$ is a pair $(M, 1^n)$, with the promise that M describes a nondeterministic Turing machine running in $o(T(n))$ time and guessing at most $n/10$ bits.*
2. ***Output.*** *For every fixed M and every sufficiently large n, $\mathcal{R}(M, 1^n)$ outputs a string $x \in \{0,1\}^n$ such that $M(x) \neq L(x)$.*
3. ***Complexity.*** *$\mathcal{R}$ runs in $\mathrm{poly}(T(n))$ time with adaptive access to an SAT oracle.*

Since $\mathcal{R}$ can find counterexamples to any faster algorithm attempting to decide L, we call $\mathcal{R}$ a refuter.

Applying the above refuter construction instead of the general NTIME hierarchy in the original proof of [Wil14], we can achieve almost-everywhere circuit lower bounds.

Other Explicit Refuters for Complexity Separations. It is instructive to compare our refuter construction to other refuter constructions, such as [BTW10, GST07, Ats06]. They showed that, assuming certain complexity separations ($\mathsf{NP} \neq \mathsf{P}$, $\mathsf{NP} \not\subset \mathsf{BPP}$ or $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{poly}}$), one can construct a refuter which takes a corresponding algorithm A claimed to solve SAT, and outputs for infinitely many n a counter-example x_n of length n such that A fails to solve SAT on x_n. All these refuters are conditional, in the sense that they assumed the (unproven) hypothesis such as $\mathsf{NP} \neq \mathsf{P}$, while our refuter is designed to witness the already proven NTIME hierarchy theorem of [FS16].

1.2.2 An XOR Lemma Based on Approximate Linear Sums of Circuits

Our second important technical ingredient—critical to our average case lower bounds—is a new XOR Lemma based on approximate linear sums of circuits. The XOR Lemma (originally due to Yao [GNW95]) says that if an n-bit Boolean function f cannot be *weakly approximated* (e.g., 0.99-approximated) by small circuits, then the kn-bit Boolean function $f^{\oplus k}$ cannot be *strongly approximated* (e.g., $(1/2 + 2^{-\Omega(k)})$-approximated) by smaller and simpler circuits.[7]

It is tempting to apply the XOR Lemma directly, to try to prove strong average-case lower bounds for ACC^0 (or $\mathsf{AC}^0[2]$) given that weak bounds are known. However, when we apply the XOR Lemma to a restricted circuit class $\mathscr{C}$, the most refined analysis of the XOR Lemma [Imp95, Kli01] still only shows that 0.99-inapproximability for $\mathsf{MAJORITY}_{t^2} \circ \mathscr{C}$ computing f implies $(1/2 + 1/t)$-inapproximability for $\mathscr{C}$ computing $f^{\oplus k}$, where $k = O(\log t)$. That is, applying the XOR Lemma to show strong-average case lower bounds for ACC^0, evidently requires proving weak average-case lower bounds for $\mathsf{MAJORITY} \circ \mathsf{ACC}^0$. But this task seems just as hard as proving strong average-case lower bounds in the first place!

We avoid this issue by proving a new kind of XOR Lemma, based on Levin's proof of the XOR Lemma [Lev87]. For a circuit class $\mathscr{C}$, we define the class of linear combinations $[0,1]\mathsf{Sum} \circ \mathscr{C}$, where an n-input circuit C from the class has the form $C(x) := \sum_i \alpha_i \cdot C_i(x)$, where each $\alpha_i \in \mathbb{R}$ and $C_i \in \mathscr{C}$, and C satisfies the promise that $C(x) \in [0,1]$ for all $x \in \{0,1\}^n$. The *complexity* of C is defined to be the maximum of $\sum_i |\alpha_i|$ and the sum of the sizes of each C_i.

Our new XOR Lemma shows that if a Boolean function f cannot be well-approximated by linear combinations of $\mathscr{C}$ circuits *on average*, then $f^{\oplus k}$ is strongly average-case hard for $\mathscr{C}$ circuits. The flexibility afforded by linear combinations allows us to improve our results to strong average-case lower bounds.

Theorem 1.14 (New XOR Lemma, Informal)**.** *For every $\delta < \frac{1}{2}$, for every function $f\colon \{0,1\}^n \to \{0,1\}$, if there is no $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit $C\colon \{0,1\}^n \to [0,1]$ of complexity $\mathrm{poly}(s,n,1/\delta)$ such that $\mathbb{E}_{x\in\{0,1\}^n}|C(x) - f(x)| \leq \delta$, then $f^{\oplus k}$ cannot be $(1/2 + 1/s)$-approximated by s-size $\mathscr{C}$ circuits, for $k = \Theta\left(\delta^{-1}\log s\right)$.*

2 Technical Overview

In this section we provide more intuition behind our almost-everywhere lower bounds. We split the discussion into two parts, one for each main technical ingredient.

- In Section 2.1, we demonstrate how to use our new refuter concept (and why it comes up naturally) to prove almost-everywhere E^{NP} lower bounds. With this powerful concept, we can automatically strengthen most of the previous E^{NP} lower bounds proved via the algorithmic method, except for the strong average-case lower bounds in [CR20].
- In Section 2.2, we show how to use an XOR Lemma for approximate linear combinations of circuits, to prove a strong average-case almost-everywhere lower bounds for E^{NP}.

2.1 Almost-Everywhere Lower Bounds for E^{NP} and the Refuter

To explain the intuition behind our almost-everywhere circuit lower bounds, it is instructive to first recall how Williams [Wil14] proved that E^{NP} does not have $2^{n^{o(1)}}$-size ACC^0 circuits, and understand why that approach only achieves an infinitely-often separation.

[7] The function $f^{\oplus k}$ partitions its kn-bit input into k blocks $x_1, x_2, \ldots, x_k$ of length n each, and outputs $\bigoplus_{i=1}^k f(x_i)$.

2.1.1 Review of E^{NP} Not in ACC^0

A Nondeterministic Algorithm $\mathcal{A}_{L^{\mathsf{hard}}}$ That Can't Be Improved. By the NTIME hierarchy [SFM78, Žák83], we know there is a language $L^{\mathsf{hard}} \in \mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n]$. Let $\mathcal{A}_{L^{\mathsf{hard}}}$ be a nondeterministic $O(2^n)$-time algorithm deciding L^{hard}.

A "Cheating" Algorithm $\mathcal{A}_{\mathsf{PCP}}$ Trying to Speed Up $\mathcal{A}_{L^{\mathsf{hard}}}$. Assume we have non-trivial derandomization algorithms for ACC^0, i.e., there is a $2^n/n^{\omega(1)}$-time algorithm for deciding Gap-UNSAT on ACC^0 circuits of n inputs and $2^{n^{o(1)}}$ size. A key idea in [Wil14, Wil13] is to combine probabilistically checkable proofs (PCPs) and non-trivial Gap-UNSAT algorithms to design a nondeterministic algorithm $\mathcal{A}_{\mathsf{PCP}}$ that tries to "speed up" the algorithm $\mathcal{A}_{L^{\mathsf{hard}}}$, as follows:

- Given an input $x \in \{0,1\}^n$, $\mathcal{A}_{\mathsf{PCP}}$ applies an efficient PCP reduction (e.g., [BV14]) to $\mathcal{A}_{L^{\mathsf{hard}}}(x)$. For $\ell = n + \Theta(\log n)$, we obtain a verifier oracle circuit $\mathsf{VPCP}_x(r)\colon \{0,1\}^\ell \to \{0,1\}$ with the following properties.

  (Simplicity) VPCP_x is an oracle circuit with gates for a function $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$. VPCP_x has very simple structure, so that if $\mathcal{O}$ is an ACC^0 circuit, then the composed circuit $\mathsf{VPCP}_x^{\mathcal{O}}$ is also in ACC^0.

  (Completeness) If $x \in L^{\mathsf{hard}}$, then there is an oracle $\mathcal{O}$ such that $\Pr_{r\in\{0,1\}^\ell}[\mathsf{VPCP}_x^{\mathcal{O}}(r) = 1] = 1$.

  (Soundness) If $x \notin L^{\mathsf{hard}}$, then for all oracles $\mathcal{O}$, $\Pr_{r\in\{0,1\}^\ell}[\mathsf{VPCP}_x^{\mathcal{O}}(r) = 1] \le 1/3$.

- Next, $\mathcal{A}_{\mathsf{PCP}}$ guesses an ACC^0 circuit C of size $2^{n^{o(1)}}$. By the simplicity property, VPCP_x^C (with oracle C) is also an ACC^0 circuit of $2^{n^{o(1)}}$ size. Running the assumed Gap-UNSAT algorithm for ACC^0 on the circuit $D = \neg\mathsf{VPCP}_x^C$, we can distinguish between the case that D is unsatisfiable (which happens for some $\mathcal{O}$, if $x \in L^{\mathsf{hard}}$) and the case that $\Pr_x[D(x)] \ge 2/3$ (which happens for all $\mathcal{O}$, if $x \notin L^{\mathsf{hard}}$). Therefore, we accept if and only if our Gap-UNSAT algorithm returns "unsatisfiable".

Intuitively, $\mathcal{A}_{\mathsf{PCP}}$ wants to "cheat" in the computation of L^{hard} by *only considering oracles of small circuit complexity*.

The E^{NP} Lower Bound. Note that the nondeterministic algorithm $\mathcal{A}_{\mathsf{PCP}}$ indeed runs faster than 2^n time: the running time of $\mathcal{A}_{\mathsf{PCP}}$ is dominated by the running time of the non-trivial derandomization algorithm, which is $o(2^n/n)$. Therefore, we know that $\mathcal{A}_{\mathsf{PCP}} \in \mathsf{NTIME}[o(2^n/n)]$, and hence it *cannot* compute L^{hard} by the NTIME hierarchy theorem.

We conclude that, for infinitely many n, there is a "bad input" $x_n \in \{0,1\}^n$ such that $\mathcal{A}_{\mathsf{PCP}}(x_n)$ rejects but $x_n \in L^{\mathsf{hard}}$.[8] For those $x_n \in L^{\mathsf{hard}}$, the completeness of the PCP implies there is an oracle $\mathcal{O}$ such that $\mathsf{VPCP}_{x_n}^{\mathcal{O}}(r) = 1$ for all r, but no such oracle admits $2^{n^{o(1)}}$-size ACC^0 circuit —otherwise, $\mathcal{A}_{\mathsf{PCP}}$ would have guessed it, and $\mathcal{A}_{\mathsf{PCP}}(x_n)$ would accept instead. Therefore, constructing a description of the oracle $\mathcal{O}$ is tantamount to constructing a function without small ACC^0 circuits.

We can now design the hard E^{NP} language as follows: on an input x of length $2n + O(\log n)$, split x into two parts x_1 and x_2 such that $|x_1| = n$. Using an NP oracle, we search for the lexicographically-first oracle $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$ for the verifier VPCP_{x_1} (that is, $\mathsf{VPCP}_{x_1}^{\mathcal{O}}(r) = 1$ for all r). Finally, we output $\mathcal{O}(x_2)$. If there is an $x_n \in \{0,1\}^n$ such that $\mathcal{A}_{\mathsf{PCP}}(x_n)$ rejects but $x_n \in L^{\mathsf{hard}}$, this E^{NP} algorithm has high ACC^0 circuit complexity on input length $2n + O(\log n)$.

[8] Note it is impossible that $\mathcal{A}_{\mathsf{PCP}}(x_n)$ accepts but $x_n \notin L^{\mathsf{hard}}$, as $\mathcal{A}_{\mathsf{PCP}}$ only guesses over a proper subset of all possible witnesses.

Input and Advice. In the above, the input x is split into two parts x_1 and x_2. The part x_1 behaves as "advice" specifying the "bad input" on which $\mathcal{A}_{\mathsf{PCP}}$ and L^{hard} differ. The part x_2 is the input to the constructed oracle $\mathcal{O}$. Luckily for us, the length of the advice is roughly the same as the input length to the oracle, so it does not affect the hardness of the overall function. For example, a hardness result superpolynomial in $|x_2|$ is also superpolynomial in $|x_1| + |x_2|$, since we assumed $|x_1| = O(|x_2|)$.

The NTIME Hierarchy Barrier. Let us examine the above proof outline more carefully. The analysis shows that the language decided by our E^{NP} algorithm is hard on inputs of length $2n$, provided that $\mathcal{A}_{\mathsf{PCP}}$ and L^{hard} give different outputs when they are restricted to inputs of length n. From the NTIME hierarchy and the fact that $\mathcal{A}_{\mathsf{PCP}} \in \mathsf{NTIME}[O(2^n/n)]$, we conclude there must be infinitely many such n.

The above argument would yield an almost-everywhere separation, if we could show that L^{hard} and $\mathcal{A}_{\mathsf{PCP}}$ are different *on all sufficiently large input lengths*. However, this would apparently require showing $\mathsf{NTIME}[2^n] \not\subset \text{i.o.-}\mathsf{NTIME}[2^n/n]$, and such an almost-everywhere separation is a notoriously hard open problem—it is even open whether $\mathsf{NEXP} \subset \text{i.o.-NP}$! It seems hopeless to make progress using the above framework, without making breakthrough progress on an almost-everywhere NTIME hierarchy.

First Observation: $\mathcal{A}_{\mathsf{PCP}}$ Guesses Short Witnesses. Here we make an important observation that bypasses the above barrier. For the above proof to work, L^{hard} only needs to be hard for the specific algorithm $\mathcal{A}_{\mathsf{PCP}}$, not necessarily *all nondeterministic $o(2^n/n)$-time algorithms*. In other words, we do not need the full power of an almost-everywhere NTIME hierarchy. Therefore, it is natural to examine what properties of the specific algorithm $\mathcal{A}_{\mathsf{PCP}}$ we can exploit.

One way in which $\mathcal{A}_{\mathsf{PCP}}$ is very different from a general $O(2^n)$-time nondeterministic algorithm is that it makes a considerably smaller number of guesses: only $2^{n^{o(1)}}$. Such restricted versions of NTIME have been studied before, under the guise of *bounded nondeterminism*. We use $\mathsf{NTIMEGUESS}[t(n), g(n)]$ to denote the class of languages decidable by nondeterministic algorithms using $O(t(n))$ steps and guessing at most $g(n)$ witness bits. Fortnow and Santhanam [FS16] showed that when $g(n)$ is sublinear, one can establish an almost-everywhere NTIME hierarchy.

Reminder of Theorem 1.12. *For every time-constructible function $T(n)$ such that $n \le T(n) \le 2^{\mathrm{poly}(n)}$,* $\mathsf{NTIME}[T(n)] \not\subset \textit{i.o.-}\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.

Trying a New Approach. A natural proposal is to apply Theorem 1.12 instead of the general NTIME hierarchy. For that purpose, we have to choose our parameters carefully so that $\mathcal{A}_{\mathsf{PCP}}$ makes few guesses. Let us check what happens when we rely on Fortnow and Santhanam's almost-everywhere hierarchy instead in our design of $\mathcal{A}_{\mathsf{PCP}}$. Our pseudocode below will fail, but studying how to fix it will lead to a correct lower bound proof.

1. Suppose we have a non-trivial circuit analysis algorithm (for CAPP or Gap-UNSAT) for ACC^0 circuits of size 2^{n^ε}. Set $k < \frac{1}{\varepsilon}$ and $T(n) = 2^{\log^k n}$. Let L^{hard} be a language in $\mathsf{NTIME}[T(n)]$ but not in $\text{i.o.-}\mathsf{NTIMEGUESS}[o(T(n)), n/10]$. Let $\mathcal{A}_{L^{\mathsf{hard}}}$ be an $O(T(n))$-time nondeterministic algorithm deciding L.

2. Given an input x of length n, $\mathcal{A}_{\mathsf{PCP}}$ applies the PCP reduction (e.g. [BV14]) to $\mathcal{A}_{L^{\mathsf{hard}}}(x)$. For $\ell = \log^k n + O(\log\log n)$, we obtain a verifier oracle circuit $\mathsf{VPCP}_x \colon \{0,1\}^\ell \to \{0,1\}$, with three properties.

   (Simplicity) VPCP_x calls an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$. VPCP_x has very simple structure such that if $\mathcal{O}$ is an ACC^0 circuit, then so is the composed circuit $\mathsf{VPCP}_x^{\mathcal{O}}$.

   (Completeness) If $x \in L^{\mathsf{hard}}$, then there is an oracle $\mathcal{O}$ such that $\Pr_{r\in\{0,1\}^\ell}[\mathsf{VPCP}_x^{\mathcal{O}}(r) = 1] = 1$.

(Soundness) If $x \notin L^{\mathsf{hard}}$, then for all oracle $\mathcal{O}$, $\Pr_{r\in\{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}_x(r)=1] \le 1/3$.

3. Next, $\mathcal{A}_{\mathsf{PCP}}$ guesses an ACC^0 circuit C of size $2^{\ell^\varepsilon} \le o(n)$ (since $k < \varepsilon^{-1}$). Note that VPCP^C_x is also an ACC^0 circuit of $2^{O(\ell^\varepsilon)}$ size, by the simplicity property. Then, we use our non-trivial circuit analysis algorithm to estimate the acceptance probability of $\mathsf{VPCP}^C_x(\cdot)$, and **accept** if and only if the estimate is $\ge 1/2$.

This new $\mathcal{A}_{\mathsf{PCP}}$ runs in $o(T(n))$ time, and guesses at most $n/10$ bits of witness, so its language is in $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$. Hence, for all large enough n, there is an $x_n \in \{0,1\}^n$ such that $x_n \in L^{\mathsf{hard}}$ while $\mathcal{A}_{\mathsf{PCP}}$ rejects x_n. Consequently, by a similar analysis as in Section 2.1.1, we have: (1) there is an oracle $\mathcal{O}$ such that $\mathsf{VPCP}^{\mathcal{O}}_x(r)=1$ for all r, and (2) no such oracle has 2^{ℓ^ε} size ACC^0 circuits.

Therefore, for all large enough n, there is an $x_n \in \{0,1\}^n$ such that the lexicographically-first correct oracle $\mathcal{O}_n\colon \{0,1\}^{\ell(n)} \to \{0,1\}$ for VPCP_x does not have $2^{\ell(n)^\varepsilon}$ size ACC^0 circuits. On input x of length m, we can set $n \approx 2^{m^{1/k}}$ so that $\ell(n) = m$, use an NP oracle to find the $\mathcal{O}_n$, and output $\mathcal{O}_n(x)$.

A Problem of Input and Advice. The above plan sounds nice, but there is a major problem: we now need a much longer advice! Following the Section 2.1.1, on an input x of length n, we must split $x = x_1x_2$ such that $|x_2| \approx \log T(|x|) \approx \log^k(n)$, use x_1 as advice to specify the "bad input", construct the oracle $\mathcal{O}$ for PCP and finally outputs $\mathcal{O}(x_2)$. In this way, we can only obtain a hardness of $2^{|x_2|^\varepsilon} < n/10$, which becomes a trivial lower bound compared to the input length n.

Solution: A 'Refuter' Algorithm for Theorem 1.12. As discussed above, we cannot afford appending x_n to the end of the input as in Section 2.1.1. Therefore, in order to make sense of the above proposal, we have to *generate the required x_n ourselves*. Our key observation is that, since we are proving lower bounds for E^{NP} anyway, we can also try to use an NP oracle to *algorithmically* find the desired x_n such that $\mathcal{A}_{\mathsf{PCP}}(x_n) \neq L^{\mathsf{hard}}(x_n)$, given n.

To this end, we introduce the concept of a *refuter* $\mathcal{R}$. For our purpose, $\mathcal{R}$ is a deterministic algorithm with access to an NP oracle which takes as input the (code of) a nondeterministic algorithm $\mathcal{A}$, and 1^n for $n \in \mathbb{N}$, with the promise that $\mathcal{A}$ runs in $o(T(n))$ time and guesses at most $n/10$ bits of witness. For all large enough n, $\mathcal{R}$ outputs a string $x_n \in \{0,1\}^n$ such that $\mathcal{A}(x_n) \neq L^{\mathsf{hard}}(x_n)$. We call such an algorithm a *refuter* for L^{hard}, since it can explicitly refute any faster algorithm $\mathcal{A}$ attempting to decide L^{hard}.

How can we construct a refuter $\mathcal{R}$? A natural idea is to enumerate all input strings of length n, then use an NP oracle to find the first $x \in \{0,1\}^n$ such that $\mathcal{A}(x) \neq L^{\mathsf{hard}}(x)$. This algorithm can find the required input x_n correctly since such an x_n exists by Theorem 1.12. However, this method is extremely inefficient, having running time $\Omega(2^n)$.

Open Up The Black Box! We observed that the hard language L^{hard} established by Theorem 1.12 is quite special. Given its structure, we can indeed design a algorithm which binary-searches over all inputs of length n to find the desired x_n. With a careful analysis of the L^{hard} language, we design a refuter for L^{hard} that runs in time $O(T(n)\cdot\log(2^n)) \le O(n\cdot T(n))$.

Our final E^{NP} algorithm with a SAT oracle works as follows. On an input of y length m, set $n \approx 2^{m^{1/k}}$ so that $\ell(n) = m$, invoke the refuter to find an input $x_n \in \{0,1\}^n$ such that $\mathcal{A}_{\mathsf{PCP}}(x_n) \neq L^{\mathsf{hard}}(x_n)$, then use the SAT oracle to find the lexicographically-first oracle $\mathcal{O}\colon \{0,1\}^m \to \{0,1\}$ such that $\mathsf{VPCP}^{\mathcal{O}}_{x_n}(r)=1$ for all r. Finally, our algorithm outputs $\mathcal{O}(y)$. The running time can be bounded by $O(T(n)\cdot n) \approx 2^m$ with the help of the SAT oracle. It is not hard to see that the language decided by this E^{NP} algorithm will be almost-everywhere hard for $2^{n^{o(1)}}$-size ACC^0 circuits, which completes the proof.

Finally, we end this subsection by providing some intuitions on how the refuter for Theorem 1.12 is constructed.

The A.E. NTIME Hierarchy. Before explaining our refuter, it is instructive to review the proof ideas of Theorem 1.12.

Let $\mathcal{A}$ be an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ machine. We define a new algorithm $\mathcal{B}$ based on $\mathcal{A}$ and show that $\mathcal{A}$ fails to compute $\mathcal{B}$ on all large enough input lengths. Specially, $\mathcal{B}$ works as follows: On an input x of length n, $\mathcal{B}$ rejects immediately if $\mathcal{A}$ rejects the witness (the $n/10$-bit prefix of) x on input 0^n. Otherwise, $\mathcal{B}$ simply outputs $\mathcal{A}(x+1)$. Here $x+1$ denotes the lexicographically next string after x. If there is no such $x+1$ (*i.e.*, $x = 1^n$), $\mathcal{B}$ just outputs 1.

One can check that $\mathcal{B}$ runs in $\mathsf{NTIME}[T(n)]$. Now we fix a large enough n, and suppose for the sake of contradiction that $\mathcal{A}(x) = \mathcal{B}(x)$ for every $x \in \{0,1\}^n$. (That is, $\mathcal{A}$ is a speed up version of $\mathcal{B}$.) There are two cases depending on the value of $\mathcal{A}(0^n)$:

1. $\mathcal{A}(0^n) = 1$. Consequently, we also have $\mathcal{B}(0^n) = 1$. This is only possible if $\mathcal{B}$ accepts every input of length n, which implies, by the definition of $\mathcal{B}$, that $\mathcal{A}$ rejects every witness on the input 0^n. Hence it follows that $\mathcal{A}(0^n) = 0$ and consequently $\mathcal{B}(0^n) = 0$ as well, a contradiction.

2. $\mathcal{A}(0^n) = 0$. Since now $\mathcal{A}$ rejects every witness on the input 0^n, we have $\mathcal{B}(0^n) = \mathcal{B}(0^n+1) = \cdots = \mathcal{B}(1^n) = 1$ by the definition $\mathcal{B}$, a contradiction to the assumption that $\mathcal{B}(0^n) = \mathcal{A}(0^n)$.

The above $\mathcal{B}$ is only hard for $\mathcal{A}$. To design a hard language against every $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm, we can add the description of that algorithm as part of input, which only adds a minor overhead. That is, the new algorithm $\mathcal{B}_{\mathsf{HRAD}}$ interprets the first $\log n$ bits as the code of an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ machine, and the rest being the witness mentioned in the definition of $\mathcal{B}$ above.

Constructing the Refuter. The above proof is nonconstructive (in the sense that it does not tell us on which input $\mathcal{A}$ and $\mathcal{B}$ differ) since it is a proof by contradiction. Our observation here is that the definition of the algorithm $\mathcal{B}$ allows us to consider a linear ordering of all inputs of length n (formed lexicographically, string x is followed by $x+1$). Let us focus on the second case above that $\mathcal{A}(0^n) = 0$ (the first case can be handled similarly, see the proof of Theorem 4.6 for details). Since $\mathcal{A}$ rejects every witness on input 0^n, we have $\mathcal{B}(x) = \mathcal{A}(x+1)$ for every x, except for $\mathcal{B}(1^n) = 1$.

Consider the following list of outputs $\mathcal{A}(0^n), \mathcal{A}(0^n+1), \ldots, \mathcal{A}(1^n), \mathcal{B}(1^n)$. Since the first and last outputs differ, one can use a binary search to find two adjacent different elements with $O(\log(2^n)) \leq O(n)$ queries to the list (see Lemma 4.4 for details). This is exactly what we want, since $\mathcal{A}(x) \neq \mathcal{A}(x+1)$ means $\mathcal{A}(x) \neq \mathcal{B}(x)$. Finally, an access to the above list can be simulated by an NP query, and we obtain the desired P^{NP} refuter.

Generalization to Other Lower Bounds. Our refuter framework is general enough that many similar E^{NP} lower bound proofs (based on Williams' algorithmic paradigm) can be adapted to the almost-everywhere setting as well, e.g., the construction of rigid matrices in [AC19, BHPT20] and the probabilistic degree lower bound in [Vio20]. See Section 8 for details.

2.2 Strong Average-Case Hardness Lower Bounds via a New XOR Lemma

In this subsection, we first explain why it is difficult to prove strong average-case lower bounds for ACC^0, and then show how we get around previous barriers with an improved XOR Lemma based on approximate linear combinations of circuits.

The Hardness Amplification Barrier. The traditional approach to average-case lower bounds is through *hardness amplification*, which ultimately aims to show how worst-case lower bounds imply average-case lower bounds. The key barrier to proving stronger average-case lower bounds for ACC^0 (or even $\mathsf{AC}^0[2]$) is the lack of an appropriate hardness amplification theorem for both classes. It is known that proving such

an amplification theorem requires computing *majority* [SV10, GSV18]. That is, to establish strong average-case lower bounds for ACC^0, one apparently needs to start with at least a worst-case $\mathsf{MAJORITY} \circ \mathsf{ACC}^0$ lower bound, which seems as hard as proving strong the average-case lower bounds for ACC^0 itself.

In a recent work, Chen and Ren [CR20] managed to bypass the above barrier and prove a strong average-case lower bounds for NQP against ACC^0, via a sophisticated win-win argument. However, this argument fails to achieve either almost-everywhere separations or sub-exponential hardness due to inherent limitation. Since the knowledge of [CR20] is not required to understand our new approach, we defer the discussion on the limitations of the techniques of [CR20] to the end of this subsection.

We show it is possible to remove the win-win argument. By directly applying a new XOR Lemma, we can prove almost-everywhere strong average-case lower bounds of sub-exponential hardness.

Algorithms and Lower Bounds for (Approximate) Linear Sums of $\mathscr{C}$. The key concept behind the new XOR Lemma is that of linear combinations of circuits. Here we recall their definition (from [Wil18b, CW19, CR20]). Let $\mathscr{C}$ be a class of functions from $\{0,1\}^n \to \{0,1\}$. We say $L\colon \{0,1\}^n \to \mathbb{R}$ is a $\mathsf{Sum} \circ \mathscr{C}$ circuit, if $L(x) := \sum_{i=1}^{t} \alpha_i \cdot C_i(x)$, where each $\alpha_i \in \mathbb{R}$ and each $C_i \in \mathscr{C}$. We define the *complexity* of L to be $\max\left(\sum_{i=1}^{t} |\alpha_i|, \sum_{i=1}^{t} \mathsf{SIZE}(C_i)\right)$.

We say $f\colon \{0,1\}^n \to \{0,1\}$ admits a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit, if there is a $\mathsf{Sum} \circ \mathscr{C}$ circuit $L\colon \{0,1\}^n \to \mathbb{R}$ such that $|L(x) - f(x)| \le \delta$ for all x. We set $\delta = 1/3$ by default when it is omitted. We also use the notation $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ to denote the class of languages which can be computed by a $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ circuit family of polynomial complexity.

For us, the most important aspect of linear combinations of circuits is that they can **preserve algorithms**. For example, if there are non-trivial algorithms solving #SAT for $\mathscr{C}$, then we can compute $\mathbb{E}_x[L(x)]$ as $\sum_{i=1}^{t} \alpha_i \cdot \mathbb{E}_x[C_i(x)]$, where $\mathbb{E}_x[C_i(x)]$ is computed by solving #SAT for $C_i \in \mathscr{C}$. Hence, if a Boolean function f has a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit L, then

$$\mathop{\mathbb{E}}_x[f(x)] - \mathop{\mathbb{E}}_x[L(x)] \le \max_x |f(x) - L(x)| \le \delta. \tag{1}$$

Therefore, we are able to estimate the acceptance probability of f in non-trivial time using non-trivial #SAT algorithms for $\mathscr{C}$. With the above observation, [CW19] applied the non-trivial #SAT algorithms for ACC^0 in [Wil14] to obtain a non-trivial CAPP algorithm for $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$, from which they proved E^{NP} cannot be computed by $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ circuits of $2^{n^{o(1)}}$ complexity.

2.2.1 Our Contribution: Direct Hardness Amplification With a Non-Standard XOR Lemma

As mentioned earlier, prior win-win arguments fail to achieve almost-everywhere separations (see Section 2.2.2 for details). Can one avoid a win-win argument and establish average-case hardness *directly*?

We show this is possible! Starting from a "ℓ_1-approximation lower bound" against $\mathsf{Sum} \circ \mathsf{ACC}^0$ circuits (slightly stronger than ℓ_∞-inapproximability), we show how strong average-case lower bounds can be established directly by applying a new non-standard version of the XOR Lemma (recalled below).

A Non-Standard XOR Lemma: For a Boolean function f and $\delta < 1/2$, if there is no $\mathsf{Sum} \circ \mathscr{C}$ circuit $C\colon \{0,1\}^n \to [0,1]$ of complexity $\mathrm{poly}(s, 1/\delta, n)$ such that $\mathbb{E}_{x\in\{0,1\}^n} |C(x) - f(x)| \le \delta$, then $f^{\oplus k}$ cannot be $(1/2 + 1/s)$-approximated by s-size $\mathscr{C}$ circuits, for $k = \Theta\left(\delta^{-1} \log s\right)$.

Note that we only consider circuits C such that $C(x) \in [0,1]$ for all x, which is crucial in our proof. We denote the class of such circuits as $[0,1]\mathsf{Sum} \circ \mathscr{C}$.

ℓ_1 Approximation Bounds Against $\mathsf{Sum} \circ \mathscr{C}$. Building on prior work on $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ circuits ([CW19, CR20]) with modifications, we show there is a function $f \in \mathsf{E}^{\mathsf{NP}}$ that cannot be approximated by $[0,1]\mathsf{Sum} \circ \mathsf{ACC}^0$ within a small constant ℓ_1 distance. Applying the new XOR Lemma above, we can establish an almost-everywhere strong average-case lower bound for E^{NP} against ACC^0.

The proof is involved, but the key insight is to see that inapproximability lower bound argument in (1) *does not* require the circuit L to approximate f on every point. That is, suppose f is a Boolean function and $L\colon \{0,1\}^n \to \mathbb{R}$ is a $\mathsf{Sum} \circ \mathscr{C}$ circuit satisfying $\mathbb{E}_x |L(x) - f(x)| \leq \delta$. In such a case, we say that f can be ℓ_1-approximated by L within distance δ. Note we still have that

$$\mathop{\mathbb{E}}_x[f(x)] - \mathop{\mathbb{E}}_x[L(x)] \leq \mathop{\mathbb{E}}_x |f(x) - L(x)| \leq \delta.$$

Let us use $\widehat{\mathsf{Sum}}_\delta \circ \mathscr{C}$ to denote the class of Boolean functions which can be ℓ_1-approximated by a family of $\mathsf{Sum} \circ \mathscr{C}$ circuits within distance δ. By the above reasoning, non-trivial CAPP algorithms for $\widehat{\mathsf{Sum}}_\delta \circ \mathscr{C}$ still follow from non-trivial $\#\mathsf{SAT}$ algorithms for $\mathscr{C}$ circuits! In spirit, this means we should be able to prove lower bounds against $\widehat{\mathsf{Sum}}_\delta \circ \mathsf{ACC}^0$, as we have a non-trivial CAPP algorithm for them (thanks to the $\#\mathsf{SAT}$ algorithm for ACC^0 of Williams [Wil14]).

Intuition for the New XOR Lemma. Now we sketch how the new XOR Lemma is proved. It is based on a careful examination of Levin's proof of the XOR Lemma [Lev87]. Let $\varepsilon = \varepsilon_k = \frac{1}{2} \cdot (1-\delta)^k$. We will prove the contrapositive, i.e., if $f^{\oplus k}$ can be $(1/2+\varepsilon)$-approximated by a $\mathscr{C}$ circuit C of size s, then we can construct a $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit of complexity $O(s \cdot n \cdot \varepsilon^{-2})$ which ℓ_1-approximates f within error $O(\delta)$.

The proof is by induction on k. For the case $k = 1$, we can take the $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit to be the circuit C itself. For the case $k > 1$, we argue as follows. For an input x to $f^{\oplus k}$, we write $x = yz$ such that $|y| = n$ and $|z| = (k-1)n$. For each $y \in \{0,1\}^n$, we consider the quantity:

$$\begin{aligned} T(y) &:= \Pr_z[f^{\oplus k}(y,z) = C(y,z)] && (2)\\ &= \Pr_z[f^{\oplus(k-1)}(z) = f(y) \oplus C(y,z)] && (3)\\ &= \Pr_z[f(y) = C(y,z) \oplus f^{\oplus(k-1)}(z)]. && (4) \end{aligned}$$

That is, $T(y)$ measures (2) how well $C(y,z)$ approximates $f^{\oplus k}$ when the first input is fixed to y; (3) how well $f(y) \oplus C(y,z)$ approximates $f^{\oplus(k-1)}$; (4) how often does $C(y,z) \oplus f^{\oplus(k-1)}(z)$ correctly predict $f(y)$, when z is uniformly random. By (2) and our assumption on C, we have

$$\mathop{\mathbb{E}}_y[T(y)] = \Pr_{y,z}[f^{\oplus k}(y,z) = C(y,z)] \geq 1/2 + \varepsilon.$$

Now there are two cases.

1. Some $T(y)$ is far from $1/2$. Suppose there is a $y \in \{0,1\}^n$ satisfying $|T(y) - 1/2| > \varepsilon/(1-\delta)$. By (3), $f^{\oplus(k-1)}(z)$ can be computed correctly by either $f(y) \oplus C(y,z)$ or by its negation on at least a $1/2 + \varepsilon_k/(1-\delta) = 1/2 + \varepsilon_{k-1}$ fraction of inputs (we treat y as fixed in the circuit). That is, this case can be reduced to the case of $k-1$.

2. All $T(y)$'s are close to $1/2$. This is the more interesting case. Suppose $|T(y) - 1/2| \leq \varepsilon/(1-\delta)$ for every y. Consider the following algorithm $\mathcal{A}$ trying to predict $f(y)$ for every y: take a uniformly random sample $z \in \{0,1\}^{(k-1)\cdot n}$, and output $C(y,z) \oplus f^{\oplus(k-1)}(z)$. For each y, by the interpretation (4), it follows that $\mathcal{A}(y)$ predicts $f(y)$ correctly with probability $T(y)$.

To ease our later discussion, in the following we specify a natural bijection between functions from $\{0,1\}^n \to [0,1]$ and probabilistic functions on $\{0,1\}^n$ with outputs in $\{0,1\}$. For a probabilistic function $\mathcal{F}\colon \{0,1\}^n \to \{0,1\}$, we define $\mathbb{I}_{\mathcal{F}}$ such that $\mathbb{I}_{\mathcal{F}}(x) := \Pr[\mathcal{F}(x) = 1]$. Conversely, for a function $f\colon \{0,1\}^n \to [0,1]$, we define $\mathbb{P}_f$ such that $\mathbb{P}_f(x)$ outputs 1 with probability $f(x)$, and 0 otherwise.

In the following, we define the correlation of two functions $g, h\colon \{0,1\}^n \to [0,1]$ as

$$\mathrm{Cor}(g,h) := \mathop{\mathbb{E}}_{x \leftarrow \{0,1\}^n}[(2g(x)-1) \cdot (2h(x)-1)]. \quad (5)$$

Note that for two functions $g\colon \{0,1\}^n \to [0,1]$ and $h\colon \{0,1\}^n \to \{0,1\}$, we have

$$\mathrm{Cor}(g,h) = \Pr_x[\mathbb{P}_g(x) = h(x)] - \Pr_x[\mathbb{P}_g(x) \neq h(x)] = 2\Pr_x[\mathbb{P}_g(x) = h(x)] - 1.$$

The following two properties of $\mathcal{A}$ are crucial for us:

1. The algorithm $\mathcal{A}$ has a non-trivial correlation with $f(y)$. More precisely, $\mathrm{Cor}(\mathbb{I}_{\mathcal{A}}, f) > 2\varepsilon$ since $\mathrm{Cor}(\mathbb{I}_{\mathcal{A}}, f) = 2 \cdot \Pr_y[\mathcal{A}(y) = f(y)] - 1 = 2\,\mathbb{E}_y[T(y)] - 1 \geq 2\varepsilon$.
2. On each input y, the algorithm $\mathcal{A}$ has little bias on its output. That is, $\mathbb{I}_{\mathcal{A}}(y) \in [1/2 - \varepsilon/(1-\delta), 1/2 + \varepsilon/(1-\delta)]$ for every y.

By a random sampling and applying a Chernoff bound, we can use a sum of $O((\varepsilon\delta)^{-2} \cdot n)$ $\mathscr{C}$ circuits to approximate $\mathbb{I}_{\mathcal{A}}(x)$ within an additive error of $\varepsilon\delta$ for every input x. We call this new $\mathsf{Sum} \circ \mathscr{C}$ circuit D. The following two properties of D follow from the above two properties of $\mathcal{A}$.

1. The circuit D and the function $f(y)$ have a correlation larger than $2\varepsilon \cdot (1-\delta)$. That is, $\mathrm{Cor}(D, f) > 2\varepsilon \cdot (1-\delta)$.
2. For every input y, $D(y) \in [1/2 - \varepsilon, 1/2 + \varepsilon]$. (Note that $\varepsilon\delta + \varepsilon/(1-\delta) = \varepsilon$.)

Now, observe that both of the bias on every input and the total correlation are all roughly ε, we can finally scale D properly to obtain a $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit D', which has a correlation larger than $(1-\delta)$ with f. Formally, we define D' so that

$$D'(y) = \frac{1 + (D(y) - 1/2)\varepsilon^{-1}}{2}.$$

The second property of D implies that D' is a $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit. And the first property of D implies that D' has a correlation $(1-\delta)$ with f (due to the definition (5)), which further implies that D' ℓ_1-approximates f with error $O(\delta)$. For the precise calculation, we refer to Appendix A.

2.2.2 An Indirect Argument for Average-Case Lower Bounds and its Limits [CR20]

In the remainder of this technical overview, we discuss an indirect hardness amplification argument (from $\mathsf{E}^{\mathsf{NP}} \not\subset \widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ to strong average-case lower bounds for E^{NP} against ACC^0) which captures the key insight in [CR20]. Our primary goal here is to highlight inherent barriers in this argument.

Lemma 2.1 (Win-Win Hardness Amplification)**.** E^{NP} *cannot be* $(1/2 + 1/\mathrm{poly}(n))$*-approximated by* ACC^0 *circuits.*

Proof Sketch. [CR20] consider a very structured NC^1-hard language $F \in \mathsf{P}$ with the following special property: If F_n can be $(\frac{1}{2} + \varepsilon)$-approximated by $\mathscr{C}$ circuits of size S, then F_n can be exactly computed by $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ circuits of complexity $\mathrm{poly}(S, 1/\varepsilon)$. Now the win-win argument goes as follows:

- **Case 1:** F **is average-case hard.** If $F \in \mathsf{P}$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by ACC^0 circuits, we are already done.
- **Case 2:** F **is average-case easy.** If F can be $1/2 + 1/\mathrm{poly}(n)$-approximated by ACC^0 circuits, then by the properties of F, NC^1 is contained in $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$. Then, it must be that E^{NP} is not in NC^1, as $\mathsf{E}^{\mathsf{NP}} \not\subset \widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$. Applying the XOR Lemma, we conclude that E^{NP} cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by NC^1 circuits, and hence neither by ACC^0. □

Combing with the worst-case $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ lower bounds in [CW19], the strong average-case lower bound for ACC^0 follows immediately. There are two inherent barriers to this win-win approach.

An Infinitely-Often Barrier. The win-win argument can only yield infinitely-often lower bounds, even assuming an almost-everywhere $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ lower bound for E^{NP}. In the first case, we only conclude that F is average-case hard on infinitely many input lengths. To make F in the first case almost-everywhere hard, we have to consider the cases (1) F is almost-everywhere average-case hard, and (2) F is infinitely-often average-case easy. But case (2) only implies a collapse from NC^1 to $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ on infinitely many input lengths. We cannot hope to establish an almost-everywhere average-case lower bound from that.

A Half-Sub-Exponential Barrier. The second barrier is subtler. Suppose for a function g we want to establish a $1/2 + 1/g(n)$ average-case lower bound against $g(n)$-size ACC^0 circuits. A careful analysis shows that $g(g(n)^{O(1)}) \leq 2^{n^{o(1)}}$, meaning g is *half-sub-exponential*.[9]

3 Preliminaries

3.1 Complexity Classes and Basic Definitions

We assume knowledge of basic complexity theory (see [AB09, Gol08] for excellent references). In particular, we assume familiarity with basic complexity classes such as P, NP, E^{NP}, AC^0, $\mathsf{AC}^0[m]$, ACC^0, and basic types of circuit gates $\mathsf{AND}, \mathsf{OR}$ and MOD_m. We will also use $\mathsf{AC}^0[\oplus]$ to denote $\mathsf{AC}^0[2]$.

Recall that Majority, denoted as $\mathsf{MAJ}\colon \{0,1\}^* \to \{0,1\}$, is the function that outputs 1 if the number of 1s in the input is at least the number of 0s, and outputs 0 otherwise. A linear threshold function (LTF) is a function $\Phi\colon \{0,1\}^n \to \{0,1\}$ of the form $\Phi(x) = \mathsf{sign}(\langle x, \omega \rangle - \theta)$ where $\omega \in \mathbb{R}^n$ and $\langle x, \omega \rangle$ is the standard inner product over $\mathbb{R}$. We use THR to denote the class of linear threshold functions.

Let $\mathsf{F} \in \{\mathsf{AND}, \mathsf{OR}, \mathsf{MOD}_m, \mathsf{MAJ}\}$ be a gate type, and $\mathscr{C}$ be a circuit class. We use $\mathscr{C} \circ \mathsf{F}$ to denote the composition of $\mathscr{C}$ with F: every $\mathscr{C} \circ \mathsf{F}$ circuit D can be described as $D(x) = C(f_1(x), \ldots, f_s(x))$ where C is a $\mathscr{C}$ circuit, and f_i are F gates. Similarly, we use $\mathsf{F} \circ \mathscr{C}$ to denote the composition of F with $\mathscr{C}$: every $(\mathsf{F} \circ \mathscr{C})$ circuit D can be described as $D(x) = f(C_1(x), \ldots, C_s(x))$ where C_i are $\mathscr{C}$ circuits, and f is a F 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 functions

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

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*.

We also study linear sums of $\mathscr{C}$ circuits.

Definition 3.1. $\mathsf{Sum} \circ \mathscr{C}$ denotes the set of circuits: every $C \in \mathsf{Sum} \circ \mathscr{C}$ can be described as $C(x) = \sum_i \alpha_i \cdot C_i(x)$ where each $\alpha_i \in \mathbb{R}$ and each $C_i(x)\colon \{0,1\}^n \to \{0,1\}$ is a $\mathscr{C}$ circuit. Moreover, if C satisfies the promise that $C(x) \in [0,1]$ for all $x \in \{0,1\}^n$, we also say C is a $([0,1]\mathsf{Sum} \circ \mathscr{C})$ circuit.

The size of C is defined as the total size of each C_i: $|C| = \sum_i |C_i|$. The *complexity* of C is defined as $\max(|C|, \sum_i |\alpha_i|)$. We use $[0,1]\mathsf{Sum} \circ \mathscr{C}[S(n)]$ to denote the set of functions can be computed by $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity $S(n)$.

[9] As before, consider two cases: If F cannot be $(1/2 + 1/g(n))$-approximated by $g(n)$-size ACC^0 circuits, we are done. If F can be $1/2 + 1/g(n)$ approximated by $g(n)$-size ACC^0, then NC^1 collapses to $\mathrm{poly}(g(n))$ size $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ circuits. Recalling E^{NP} cannot be computed by $2^{n^{o(1)}}$-size $\widetilde{\mathsf{Sum}} \circ \mathsf{ACC}^0$ circuits, it follows that E^{NP} cannot be computed by s-size NC^1 circuits such that $\mathrm{poly}(g(s)) = 2^{n^{o(1)}}$, and this can be amplified to $(1/2 + 1/s^{\Omega(1)})$ average-case lower bounds against $s^{\Omega(1)}$-size ACC^0 circuits. Since the unconditional lower bound at the end is the minimum of $g(n)$ and s, we want $s = g(n)$, which forces $g(g(n)^{O(1)}) \leq 2^{n^{o(1)}}$.

3.2 Elementary Properties of Norm and Inner Product

We recall some properties of norms and inner products of functions on the Boolean cube, which will be useful. For a function $f\colon \{0,1\}^n \to \mathbb{R}$, we define its ℓ_p-norm as $\|f\|_p = (\mathbb{E}_{x\sim\mathcal{U}_n}[|f(x)|^p])^{1/p}$. The ℓ_∞-norm of f is defined as the maximum absolute value of f: $\|f\|_\infty = \max_{x\in\{0,1\}^n} |f(x)|$. For two functions $f,g\colon \{0,1\}^n \to \mathbb{R}$, we define their inner product as $\langle f,g\rangle := \mathbb{E}_{x\sim\mathcal{U}_n}[f(x)\cdot g(x)]$. Note that the Cauchy-Schwarz inequality implies $|\langle f,g\rangle| \le \|f\|_2 \cdot \|g\|_2$. In particular, $\|f\|_1 = \langle |f|, \mathbf{1}\rangle \le \|f\|_2$ where $\mathbf{1}$ is the constant-one function.

Throughout this paper, we will use the notion of approximating functions within ℓ_1-distance, we state the definition below.

Definition 3.2. Given $\delta \in [0,1]$ and a function $f\colon \{0,1\}^n \to \{0,1\}$, we say a real-valued function $g\colon \{0,1\}^n \to \{0,1\}$ $(1-\delta)$-approximates f within ℓ_1-distance if $\|f-g\|_1 = \mathbb{E}_{x\leftarrow\mathcal{U}_n}[|f(x)-g(x)|] \le \delta$. Specially, when g is Boolean function, the approximation is consistent with the usual approximation, that is, $\Pr_{x\leftarrow\mathcal{U}_n}[f(x)\neq g(x)] = \|f-g\|_1 \le \delta$.

3.3 Pseudorandom Generators

We consider different types of Pseudorandom Generators (PRG).

Definition 3.3. Let $S,\ell\colon \mathbb{N}\to\mathbb{N}$ and $\varepsilon\colon \mathbb{N}\to(0,1)$ be functions, where S,ℓ,ε denote size, seed length and error respectively. Let $\mathscr{C}$ be a circuit class. A function $G\colon \{0,1\}^* \to \{0,1\}^*$ 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 $C\colon \{0,1\}^n \to \{0,1\}$ of size $S(n)$,

$$\left| \Pr_{\mathbf{s}\in\{0,1\}^{\ell(n)}}[C(G(\mathbf{s}))=1] - \Pr_{\mathbf{r}\in\{0,1\}^n}[C(\mathbf{r})=1] \right| < \varepsilon(n). \tag{6}$$

If (6) only holds for infinitely many lengths n, then we say G is an *infinitely often PRG* (i.o. PRG).

We also consider "nondeterministic pseudorandom generators" (NPRGs).

Definition 3.4. Let $S,\ell\colon \mathbb{N}\to\mathbb{N}$ and $\varepsilon\colon \mathbb{N}\to(0,1)$ be functions, and $\mathscr{C}$ be a circuit class. Let $M(x,y)$ be a deterministic Turing machine. 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 following holds:

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

$$\left| \Pr_{\mathbf{x}\in\{0,1\}^{\ell(n)}}[C(G_y(\mathbf{x}))=1] - \Pr_{\mathbf{z}\in\{0,1\}^n}[C(\mathbf{z})=1] \right| < \varepsilon(n),$$

   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.

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

Although an NPRG in general does not output *the same PRG* in its different nondeterministic branches, it could useful for many tasks such as derandomizing MA. The concept of NPRG is implicit in [IKW02], and is used explictly in [CMMW19, Che19, CR20].

We will use two standard PRG construction from literature. The first is the famous Nisan-Wigderson(NW) PRG ([NW94]), which uses a (presumably hard) function f in its design. Recall that Junta_k is the family of k-juntas, i.e. functions that only depend on k input bits. The key property of the NW PRG is that, given a $\mathscr{C}$ circuit that breaks the NW PRG based on some hard function, the complexity of approximating that hard function is $\mathscr{C} \circ \mathsf{Junta}_a$ for some parameter a. Therefore, in order to fool $\mathscr{C}$ circuits, the hard function f used by the NW PRG needs to be hard to approximate by $\mathscr{C} \circ \mathsf{Junta}_a$ circuits.

Lemma 3.5 ([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 typical circuit class. There is a function $G\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that the following hold. For any function $Y\colon \{0,1\}^\ell \to \{0,1\}$ 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 $G(Y, \mathcal{U}_t)$ [10] ε-fools every $\mathscr{C}$ circuit (of size S). That is, for any $\mathscr{C}$ circuit C (of size S),*

$$\left| \Pr_{\mathbf{s} \in \{0,1\}^t} [C(G(Y, \mathbf{s})) = 1] - \Pr_{\mathbf{x} \in \{0,1\}^m} [C(\mathbf{x}) = 1] \right| \leq \varepsilon.$$

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

We also need the following construction of PRG from [Uma03]. Roughly speaking, it shows that if a function f is hard against general circuits of a certain size, then f can be used to produce a powerful PRG.

Lemma 3.6 ([Uma03])**.** *There is a universal constant g and a function $G\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for every s and $Y\colon \{0,1\}^\ell \to \{0,1\}$, if Y cannot be computed by (general) circuits of size s^g, then $G(Y, \mathcal{U}_{g\ell})$ $1/s$-fools (general) circuits of size s. That is, for all circuit C of size at most s, it holds:*

$$\left| \Pr_{x \in \{0,1\}^{\ell g}} [C(Y, x) = 1] - \Pr_{x \sim \{0,1\}^s} [C(x) = 1] \right| \leq \frac{1}{s}.$$

Furthermore, G is computable in $\mathrm{poly}(|Y|)$ time.

3.4 Hardness Amplification

Hardness Amplification is crucial in our strong average-case lower bounds proof. We state our non-standard XOR lemma below. It is a careful adaption of Levin's proof of Yao's XOR Lemma [Lev87, GNW95]. The proof is presented in Appendix A.

Definition 3.7. Let $f\colon \{0,1\}^n \to \{0,1\}$, and let $k \geq 1$ be an integer, define the function $f^{\oplus k}\colon \{0,1\}^{kn} \to \{0,1\}$ to be $f^{\oplus k}(x_1, \ldots, x_k) := \oplus_{1 \leq i \leq k} f(x_i)$, where $x_i \in \{0,1\}^n$ for every $i \in [k]$.

Lemma 3.8. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a boolean function. Let $\delta < \frac{1}{2}$, For any $k \geq 1$, let $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right)$. If f cannot be $(1-\delta)$-approximated in ℓ_1 distance by $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity*[11] *$O\left(\frac{n \cdot s}{(\delta \cdot \varepsilon_k)^2}\right)$, then $f^{\oplus k}$ cannot be $(\frac{1}{2} + \varepsilon_k)$-approximated by $\mathscr{C}$ circuits of size s.*

[10] $\mathcal{U}_m$ denotes uniform distribution over $\{0,1\}^m$.

[11] Recall *complexity* of a $\mathsf{Sum} \circ \mathscr{C}$ circuit $C = \sum_i \alpha_i C_i$ is defined as $\max(\sum_i \mathsf{SIZE}(C_i), \sum_i |\alpha_i|)$.

We remark that Lemma 3.8 implies Theorem 1.14: In order to establish $(1/2+1/s)$-inapproximability result, it suffices to set k so that $\varepsilon_k \le 1/s$. Since $\varepsilon_k = (1-\delta)^{k-1}(1/2-\delta)$, we can choose $k \le O(\log_{(1-\delta)}(1/s)) \le O(\delta^{-1}\log s)$.

For brevity, we also say f cannot be $(1-\delta)$-approximated by, or is δ-far from low-complexity $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits in ℓ_1-distance, if the above condition for f holds.

We also need standard worst-case to strong average-case hardness amplification. We refer to [STV01] for an excellent exposition.

Lemma 3.9 ([STV01])**.** *There is a constant $c \ge 1$ such that, for any time-constructible function $S(n)$ and every $f\colon \{0,1\}^n \to \{0,1\}$ that does not have (general) circuits of size $S(n)$. There is a function $g\colon \{0,1\}^{O(n)} \to \{0,1\}$ that cannot be $(1/2+S(n)^{-1/c})$-approximated by circuits of size $S(n)^{1/c}$. Furthermore, given the 2^n-length truth table of f, the truth table of g can be constructed in $2^{O(n)}$ time.*

3.5 Probabilistically Checkable Proofs

We need two probabilistically checkable proof (PCP) systems from [BV14, CW19]. We state them below.

Lemma 3.10 ([BV14])**.** *Let M be an algorithm running in time $T = T(n) \ge n$ on inputs of the form (x,y) where $|x| = n$. Given $x \in \{0,1\}^n$, one can output in $\mathrm{poly}(n, \log T)$ time circuits $Q\colon \{0,1\}^r \to \{0,1\}^{rt}$ for $t = \mathrm{poly}(r)$ and $R\colon \{0,1\}^t \to \{0,1\}$ such that:*

- ***Proof length.*** *$2^r \le T \cdot \mathrm{polylog} T$.*
- ***Completeness.*** *If there is a $y \in \{0,1\}^{T(n)}$ such that $M(x,y)$ accepts then there is a map $\pi\colon \{0,1\}^r \to \{0,1\}$ such that for all $z \in \{0,1\}^r$, $R(\pi(q_1),\dots,\pi(q_t)) = 1$ where $(q_1,\dots,q_t) = Q(z)$.*
- ***Soundness.*** *If no $y \in \{0,1\}^{T(n)}$ causes $M(x,y)$ to accept, then for every map $\pi\colon \{0,1\}^r \to \{0,1\}$, at most $\frac{2^r}{n^{10}}$ distinct $z \in \{0,1\}^r$ have $R(\pi(q_1),\dots,\pi(q_t)) = 1$ where $(q_1,\dots,q_t) = Q(z)$.*
- ***Complexity.*** *Q is a projection, i.e., each output bit of Q is a bit of input, the negation of a bit, or a constant. R is a 3CNF.*

Note that this is an exremely efficient PCP: the 3CNF R and the projection Q collectively form the verifier for the PCP. The following lemma from [CW19, VW20] is a slight modification of the probabilistically checkable proof of proximity (PCPP) system in [BGH+06].

Lemma 3.11 ([CW19, VW20])**.** *There are constants $0 < s_{\mathsf{pcpp}} < c_{\mathsf{pcpp}} < 1$ and a polynomial-time transformation that, given a circuit D on n inputs of size $m \ge n$, outputs a 2-SAT instance F on the variable set $\mathcal{Y} \cup \mathcal{Z}$ where $|\mathcal{Y}| \le \mathrm{poly}(n)$, $|\mathcal{Z}| \le \mathrm{poly}(m)$, and the following hold for all $x \in \{0,1\}^n$:*

- *If $D(x) = 1$, then $F|_{\mathcal{Y}=\mathsf{Enc}(x)}$ on variable set $\mathcal{Z}$ has a satisfying assignment $\mathcal{Z}_x$ such that at least c_{pcpp}-fraction of the clauses are satisfied. Furthermore, there is a $\mathrm{poly}(m)$ time algorithm that given x outputs $\mathcal{Z}_x$.*
- *If $D(x) = 0$, then there is no assignment to the $\mathcal{Z}$ variables in $F|_{\mathcal{Y}=\mathsf{Enc}(x)}$ satisfies more than s_{pcpp}-fraction of the clauses.*

Moreover, the number of clauses in the 2-SAT instance F is a power of 2, and for each $i \in [|\mathcal{Y}|]$, $\mathsf{Enc}_i(x)$ is a parity function depending on at most $n/2$ bits of x.

3.6 Non-trivial #SAT Algorithms

In this paper we will make use of the following non-trivial #SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$.

Theorem 3.12 ([Wil18a]). *For every $d, m \in \mathbb{N}$, there is an $\varepsilon = \varepsilon(d,m) > 0$ such that the number of satisfying assignments to a 2^{n^ε}-size n-input $\mathsf{AC}^0_d[m] \circ \mathsf{THR}$ circuit can be counted deterministically in 2^{n-n^ε} time.*

4 Refuter for A.E. NTIME Hierarchy with Sublinear Witnesses

In this section we prove a key technical ingredient of this paper: an almost-everywhere sublinear witness NTIME hierarchy with a refuter algorithm (Theorem 1.12). We begin with some notation in Section 4.1. In Section 4.2, we define the hard language $\mathcal{A}^T_{\mathsf{FS}}$ and prove Theorem 1.12. In Section 4.3, we construct the corresponding refuter for Theorem 1.12. In Section 4.4, we discuss an adaptation to the "robustly often" lower bound of [FS11], which will be useful in the construction of rigid matrices.

4.1 Preliminaries

First, we recall the definition of the class of languages decided by nondeterministic algorithms with bounded nondeterminism.

Definition 4.1. Given two non-decreasing and time-constructible functions $T(n), g(n)$, we define

$$\mathsf{NTIMEGUESS}[T(n), g(n)]$$

to be the class of languages decidable by nondeterministic algorithms running in $O(T(n))$ steps and guessing at most $g(n)$ bits as witness.

We fix a natural enumeration of all (multitape) nondeterministic Turing machines. Note that the specific model does not really matter, see the discussion in [Wil13], Section 2.1. We use the integer M to denote the M-th nondeterministic Turing machine in the enumeration. For simplicity, we assume all machines are random-access machines in the following.

For a nondeterministic Turing machine M, an input x, and a witness z, let $\mathcal{V}_M(x,T,z)$ denote the result of running M on the input x for at most T steps with witness z. More precisely:

- $\mathcal{V}_M(x,T,z) = \perp$ if the simulation fails. That is, either (1) M does not stop after T steps, or (2) M guesses more than $|z|$ bits during the simulation.
- Otherwise, if the simulation succeeds, $\mathcal{V}_M(x,T,z) = 1$ if M accepts x with witness z, and $\mathcal{V}_M(x,T,z) = 0$ otherwise. (It is possible that M only uses a prefix of z as the witness.)

Note that for an M using $o(T(n))$ time and $g(n)$ guesses, we have

$$M(x) = 1 \Leftrightarrow \bigvee_{w \in \{0,1\}^{g(n)}} \mathcal{V}_M(x, T(n), w) = 1.$$

We also need a standard encoding of two integers and a Boolean string. Given $M, n \in \mathbb{N}$ and $z \in \{0,1\}^*$, we encode them by

$$\langle M, n, z \rangle = 1^M 01^{(n-M-2-|z|)} 0z.$$

Note that we require $n \geq M + 2 + |z|$ for the encoding to be valid. Observe that $|\langle M, n, z \rangle| = n$. Given an input $x \in \{0,1\}^\star$, one can unambiguously determine the triple M, n, z such that $\langle M, n, z \rangle = x$, or that x is not a valid encoding of any triple.

4.2 Almost-Everywhere NTIME Hierarchy with Sublinear Witness Length

To provide more intuition for our results, we first prove the following almost-everywhere NTIME hierarchy theorem of Fortnow and Santhanam [FS16].

Reminder of Theorem 1.12. *For every time-constructible function $T(n)$ such that $n \le T(n) \le 2^{\mathrm{poly}(n)}$, $\mathsf{NTIME}[T(n)] \not\subset$ i.o.-$\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.*

An Almost-Everywhere Diagonalizing Language. We start by defining a language constructed by diagonalization (adapted from [FS11, FS16]), which is used as the hard language in Theorem 1.12. Later in Section 4.3, we will construct an efficient refuter for this language as well.

Definition 4.2. For every time-constructible function $T(n)$ such that $n \le T(n) \le 2^{\mathrm{poly}(n)}$, we define a nondeterministic algorithm $\mathcal{A}^T_{\mathsf{FS}}$ as follows:

- Given an input x, parse it as $x = \langle M, n, z\rangle$, and reject immediately if there is no valid parsing, or if $|z| \neq n/10$.[12]
- $\mathcal{A}^T_{\mathsf{FS}}$ accepts x if and only if both of the following hold:
  - M rejects $\langle M, n, 0^{n/10}\rangle$ in $T(n)$ steps with witness z. That is, $\mathcal{V}_M(\langle M, n, 0^{n/10}\rangle, T(n), z) = 0$.
  - (This condition is only required for $z \neq 1^{n/10}$.) M accepts $\langle M, n, z+1\rangle$ in $T(n)$ steps while guessing at most $n/10$ bits.[13]

By the construction above, it is clear that $\mathcal{A}^T_{\mathsf{FS}}$ is an $\mathsf{NTIME}[T(n)]$ algorithm.

The following list of inputs $\{x_i^{M,(n)}\}$ greatly simplifies our discussion and the proof. For clarity, let $w_1^{(n)}, w_2^{(n)}, \ldots, w_{2^{n/10}}^{(n)}$ be the list of all $2^{n/10}$-length binary strings, sorted in lexicographical order. We define

$$x_i^{M,(n)} := \langle M, n, w_i^{(n)}\rangle.$$

When M is clear from the context, we omit the superscript M, and use $x_i^{(n)}$ to denote $x_i^{M,(n)}$. Note that when the encoding is valid, all strings $x_i^{(n)}$ have length exactly n.

The following lemma summarizes the behavior of $\mathcal{A}_{\mathsf{FS}}$ on the strings in the list $\{x_i^{M,(n)}\}$ when M is an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm, which will be used frequently in the rest of the section.

Lemma 4.3. *Suppose M runs in $o(T(n))$ time and guesses no more than $n/10$ bits of witness. For every sufficiently large n and $i \in \{1, \ldots, 2^{n/10}\}$, we have*[14]

$$\mathcal{A}_{\mathsf{FS}}(x_i^{(n)}) = \begin{cases} [\mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0] \wedge M(x_{i+1}^{(n)}) & 1 \le i < 2^{n/10} \\ [\mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0] & i = 2^{n/10}. \end{cases}$$

Proof. Since M is an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm, for every sufficiently large n, the algorithm $\mathcal{A}^T_{\mathsf{FS}}$ can simulate M faithfully on the entire collection of inputs $\{x_i^{(n)}\}$. The conclusion follows from the definition of $\mathcal{A}^T_{\mathsf{FS}}$. □

[12] Here, and in the rest of this section, $n/10$ means $\lfloor n/10\rfloor$ when n is not a multiple of 10.

[13] We use $z+1$ to denote the lexicographically next string after z in $\{0,1\}^{n/10}$.

[14] In the following, we use the Iverson bracket notation where for a Boolean-valued predicate P, $[P] = 1$ if P is true and $[P] = 0$ otherwise.

Now we turn to the proof of Theorem 1.12.

Proof of Theorem 1.12. It suffices to show that $L(\mathcal{A}^T_{\mathsf{FS}}) \notin$ i.o.-NTIMEGUESS$[o(T(n)), n/10]$, i.e., the language decided by $\mathcal{A}^T_{\mathsf{FS}}$ cannot be decided in $o(T(n))$ time with at most $n/10$ nondeterministic guess bits.

Let M be an NTIMEGUESS$[o(T(n)), n/10]$ machine. We will show that for every sufficiently large n, the machine M and $\mathcal{A}^T_{\mathsf{FS}}$ cannot agree on all inputs in $\{0,1\}^n$.

So for the sake of contradiction, suppose for large enough n that $M(x) = \mathcal{A}^T_{\mathsf{FS}}(x)$ holds for all $x \in \{0,1\}^n$. In particular, this means for all $i \in [2^{n/10}]$, we have $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = M(x_i^{(n)})$.

Consider one special input: $x_1^{(n)} = \langle M, n, w_1^{(n)} \rangle$. We consider the two possible outputs of $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)})$, and show that both of them lead to a contradiction.

- **Case 1: $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 1$.** By assumption, $M(x_1^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 1$. We show that $M(x_i^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$ by induction on i. The base case of $i = 1$ is already established. Assuming $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = M(x_i^{(n)}) = 1$, it is easy to see from Lemma 4.3 that $M(x_{i+1}^{(n)}) = 1$ as well, hence also $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = M(x_{i+1}^{(n)}) = 1$ by assumption.

  We conclude that $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$, which in turn implies (by Lemma 4.3)

  $$\text{for all } i \in [2^{n/10}], \quad \mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0.$$

  This means that M on $x_1^{(n)}$ rejects every possible witness, and hence $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = M(x_1^{(n)}) = 0$, a contradiction.

- **Case 2: $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 0$.** By assumption $M(x_1^{(n)}) = 0$. That is, M on $x_1^{(n)}$ rejects every witness, which means

  $$\text{for all } i \in [2^{n/10}], \quad \mathcal{V}_M(x_1^{(n)}, T(n), w_i^{(n)}) = 0. \tag{7}$$

  By Lemma 4.3, we have $M(x_{2^{n/10}}^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_{2^{n/10}}^{(n)}) = 1$. By a backward induction (from $i = 2^{n/10}$ down to $i = 1$) it follows that $M(x_i^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$. The base case of $i = 2^{n/10}$ is already established.

  Assuming $M(x_i^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$, it is easy to see from (7) and Lemma 4.3 that $M(x_{i-1}^{(n)}) = \mathcal{A}^T_{\mathsf{FS}}(x_{i-1}^{(n)}) = 1$ as well. Therefore, by a backward induction, we conclude $\mathcal{A}^T_{\mathsf{FS}}(x_i^{(n)}) = 1$ for all $i \in [2^{n/10}]$, which contradicts our assumption that $\mathcal{A}^T_{\mathsf{FS}}(x_1^{(n)}) = 0$.

Therefore, we conclude that for all such M and for every sufficiently large n, M fails to compute $L(\mathcal{A}^T_{\mathsf{FS}})$ on inputs of length n. This completes the proof. □

4.3 Construction of the Refuter

The proof above shows that every NTIMEGUESS$[o(T(n)), n/10]$ machine fails to compute $L(\mathcal{A}_{\mathsf{FS}})$ on all sufficiently large input lengths: for every such machine M and sufficiently large n, there is an $x_n \in \{0,1\}^n$ such that $M(x_n) \neq \mathcal{A}^T_{\mathsf{FS}}(x_n)$. Now we design an algorithm to find such an x_n efficiently.

To begin with, we need the following binary search algorithm.

Lemma 4.4. *There is an algorithm A satisfying the following.*

- ***Input.*** *A is given an explicit integer $n \geq 2$ (written in binary form) as input, together with oracle access to a list $(a_1, \ldots, a_n) \in \{0,1\}^n$ such that $a_1 \neq a_n$.*
- ***Output.*** *An index $p \in [1, n-1]$ such that $a_p \neq a_{p+1}$.*
- ***Efficiency.*** *A runs in $O(\log n)$ time, makes at most $O(\log n)$ queries to the list.*

Proof. The algorithm A works as follows.

1. **Initialize:** We set $L = 1, R = n$ and query a_L, a_R. From the promise on input we have $a_L \neq a_R$.
2. As long as $R - L \geq 2$, we set $q = \lfloor \frac{L+R}{2} \rfloor$ and query a_q. Since $a_L \neq a_R$, a_q cannot equal to both of them. If $a_q \neq a_L$, we update $R = q$, otherwise we update $L = q$. Note that the invariant $a_L \neq a_R$ always holds.
3. We repeat Step (2) until $R - L \leq 1$. Since $a_L \neq a_R$, it must be the case $R = L + 1$. Return L.

□

We need a special form of this algorithm, which we state as a corollary below.

Corollary 4.5. *There is an algorithm A' satisfying the following.*

- ***Input.*** *A' is given an explicit integer $n \geq 1$ (written in binary form) as input, together with oracle access to two lists $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in \{0,1\}^n$, with the promise that for every $i < n$, $a_i = b_{i+1}$, as well as $b_1 \neq a_n$.*
- ***Output.*** *An index $p \in [1, n]$ such that $a_p \neq b_p$.*
- ***Efficiency.*** *A' runs in $O(\log n)$ time, makes at most $O(\log n)$ queries to the list.*

Proof. A' simulates A from Lemma 4.4 with parameter $n + 1$ and the list $(b_1, \ldots, b_n, a_n)$. Since $a_n \neq b_1$ from our assumption, this list satisfies the promise of Lemma 4.4, and all queries to the list can be answered by querying the two lists $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ accordingly. A' then outputs the index $p \in [1, n]$ reported by A. Now we can see that $a_p \neq b_p$: If $p < n$, then $a_p = b_{p+1} \neq b_p$ as desired, otherwise we have $a_n \neq b_n$, which is also valid. □

We are now ready to construct a refuter algorithm that explicitly witnesses the hardness of the language $\mathcal{A}^T_{\mathsf{FS}}$ from Theorem 4.7.

Theorem 4.6. *For every time-constructible function $T(n)$ such that $n \leq T(n) \leq 2^{\mathrm{poly}(n)}$, there is an algorithm $\mathcal{R}^T$ such that:*

1. ***Input.*** *The input for $\mathcal{R}^T$ is a pair $(M, 1^n)$, with the promise that the M-th nondeterministic Turing machine runs in $o(T(n))$ time and guesses no more than $n/10$ bits of witness.*
2. ***Output.*** *For every fixed M and all large enough n, $\mathcal{R}^T(M, 1^n)$ outputs a string $x \in \{0,1\}^n$ such that $\mathcal{A}^T_{\mathsf{FS}}(x) \neq M(x)$.*
3. ***Complexity.*** *$\mathcal{R}^T$ is a deterministic algorithm running in $\mathrm{poly}(T(n))$ time with adaptive access to a SAT oracle.*

Since the output of $\mathcal{R}^T$ can explicitly refute any $o(T(n))$-time nondeterministic algorithm which claims to decide $L(\mathcal{A}^T_{\mathsf{FS}})$, we also call $\mathcal{R}^T$ a refuter.

Proof. Let $(M, 1^n)$ be an input to $\mathcal{R}^T$. Suppose M satisfies the stated promise. The proof of Theorem 1.12 shows that a *differing input* must exist in the list $\{x_i^{M,(n)}\}$ (that is, $M(x_j^{M,(n)}) \neq \mathcal{A}_{\mathsf{FS}}^T(x_j^{M,(n)})$ for some $j \in [2^{n/10}]$). In the following, we fix the machine M and write $x_i^{M,(n)}$ instead of $x_i^{(n)}$ for brevity. We now consider the following two cases:

- **Case 1:** $M(x_1^{(n)}) = 0$. In this case, M on $x_1^{(n)}$ rejects every witness, then Lemma 4.3 implies

$$\mathcal{A}_{\mathsf{FS}}^T(x_i^{(n)}) = \begin{cases} M(x_{i+1}^{(n)}) & \text{if } i < 2^{n/10}, \\ 1 & \text{if } i = 2^{n/10}. \end{cases}$$

  The pair of lists $(\mathcal{A}_{\mathsf{FS}}^T(x_1^{(n)}), \ldots, \mathcal{A}_{\mathsf{FS}}^T(x_{2^{n/10}}^{(n)}))$ and $(M(x_1^{(n)}), \ldots, M(x_{2^{n/10}}^{(n)}))$ satisfy the promise of Corollary 4.5 since $M(x_1^{(n)}) = 0 \neq \mathcal{A}_{\mathsf{FS}}^T(x_{2^{n/10}}^{(n)}) = 1$. We then invoke the algorithm in Corollary 4.5 to find an index p such that $\mathcal{A}_{\mathsf{FS}}^T(x_p^{(n)}) \neq M(x_p^{(n)})$. The query to the lists can be answered with the help of the SAT oracle in $\mathrm{poly}(T(n))$ time. Therefore, the whole algorithm runs in $\mathrm{poly}(T(n)) \cdot O(\log 2^{n/10}) = \mathrm{poly}(T(n))$ time.

- **Case 2:** $M(x_1^{(n)}) = 1$. In this case, M on $x_1^{(n)}$ accepts at least one witness. Let j be the minimum index from $[2^{n/10}]$ such that the witness $w_j^{(n)}$ is accepted by $M(x_1^{(n)})$. By a binary search, the index j can be found with the help of the SAT oracle in $\mathrm{poly}(T(n))$ time. Now we focus on the partial list $(x_1^{(n)}, \ldots, x_j^{(n)})$, given that $w_j^{(n)}$ is the first witness accepted by $M(x_1^{(n)})$, Lemma 4.3 implies

$$\mathcal{A}_{\mathsf{FS}}^T(x_i^{(n)}) = \begin{cases} M(x_{i+1}^{(n)}) & \text{if } i < j, \\ 0 & \text{if } i = j. \end{cases}$$

  The pair of lists $(\mathcal{A}_{\mathsf{FS}}^T(x_1^{(n)}), \ldots, \mathcal{A}_{\mathsf{FS}}^T(x_j^{(n)}))$ and $(M(x_1^{(n)}), \ldots, M(x_j^{(n)}))$ satisfy the promise of Corollary 4.5 since $M(x_1^{(n)}) = 1 \neq \mathcal{A}_{\mathsf{FS}}^T(x_j^{(n)}) = 0$. We can then invoke the algorithm in Corollary 4.5 to find an index p such that $\mathcal{A}_{\mathsf{FS}}^T(x_p^{(n)}) \neq M(x_p^{(n)})$. Each query to the lists can be answered with the help of the SAT oracle in $\mathrm{poly}(T(n))$ time. Therefore, the whole algorithm runs in $\mathrm{poly}(T(n)) \cdot O(\log 2^{n/10}) = \mathrm{poly}(T(n))$ time.

In summary, the algorithm works well in both cases and the theorem is proved.

□

4.4 Refuter for "Robustly Often" Lower Bounds

Finally, we generalize our refuter construction to the "robustly often" NTIME hierarchy (Theorem 4.7) by Fortnow and Santhanam [FS11].

Theorem 4.7. *For any non-decreasing time-constructible function $T(n)$ such that $T(n) \geq n$ and $T(n+1) = O(T(n))$, there exists a language $L \in \mathsf{NTIME}[T(n)]$ such that, for any $L' \in \mathsf{NTIME}[o(T(n))]$, for all but finitely many n, there exists $m \in [n, n+T(n)]$ such that L and L' cannot agree on all inputs of length m.*

Theorem 4.7 also yields a corresponding refuter. An advantage of this refuter is that it does not have restrictions on the size of the witness, which will be crucial later in our construction of rigid matrices. A drawback of this refuter is it can only output a "bad" input having length in an interval $[n, n+T(n)]$.

Theorem 4.8. *For any non-decreasing time-constructible function $T(n)$ such that $T(n) \geq n$ and $T(n+1) = O(T(n))$, there is an $\mathsf{NTIME}[T(n)]$ machine $\mathcal{A}^T_{\mathsf{RO}}$ and an algorithm $\mathcal{R}^T_{\mathsf{RO}}$ such that:*

1. ***Input.*** *The input for $\mathcal{R}^T_{\mathsf{RO}}$ is a pair $(M, 1^n)$ with the promise that the M-th nondeterministic Turing machine runs in $o(T(n))$ time.*
2. ***Output.*** *For every fixed M and all large enough n, $\mathcal{R}^T_{\mathsf{RO}}(M, 1^n)$ outputs a string x such that $|x| \in [n, n + T(n)]$ and $\mathcal{A}^T_{\mathsf{RO}}(x) \neq M(x)$.*
3. ***Complexity.*** *$\mathcal{R}^T_{\mathsf{RO}}$ is a deterministic algorithm running in $\mathrm{poly}(T(\mathrm{poly}(T(n))))$ time with adaptive access to a SAT oracle.*

The proof is similar to the proof of Theorem 1.12 and Theorem 4.6. We include it in Appendix B for completeness.

5 Almost-Everywhere Worst-Case Lower Bounds for E^{NP}

Given the powerful refuter, most previous infinitely-often lower bounds for E^{NP} proved by Williams' algorithmic method can now be improved to almost-everywhere separations.

As a warm-up, we prove Theorem 1.1, showing that non-trivial $\mathsf{Gap\text{-}UNSAT}$ algorithm for $\mathsf{AND} \circ \mathsf{OR} \circ \mathscr{C}$ (note that this requirement is weaker than the existence of a nontrivial CAPP algorithm) implies E^{NP} is almost-everywhere hard for $\mathscr{C}$.

Reminder of Theorem 1.1. *There are universal constants $\varepsilon \in (0,1)$, $K \geq 1$ satisfying the following. Let $\mathscr{C}$ be typical, and let $s(n)$ be any non-decreasing time-constructible function with $n \leq s(n) \leq 2^{\varepsilon n}$ for every n. If $\mathsf{Gap\text{-}UNSAT}$ on $\mathsf{AND} \circ \mathsf{OR} \circ \mathscr{C}$ circuits of size $s(n)^K$ can be solved deterministically in $2^n / n^{\omega(1)}$ time, then there are functions in E^{NP} that do not have $\mathscr{C}$ circuits of size $s(n/2)$, for every sufficiently large n.*

Proof of Theorem 1.1. Let $C, K \in \mathbb{N}$ be two sufficiently large constants, we also set $\varepsilon = \frac{1}{10C}$.

Let $T(n) = n^C$. Our proof will follow the outline in the Section 2.1: we consider the algorithm $\mathcal{A}^T_{\mathsf{FS}}$ from Definition 4.2, define a "cheating" algorithm $\mathcal{A}_{\mathsf{PCP}}$ which tries to speed up $\mathcal{A}^T_{\mathsf{FS}}$, and apply the refuter from 4.6 to efficiently construct a differing input between $\mathcal{A}^T_{\mathsf{FS}}$ and $\mathcal{A}_{\mathsf{PCP}}$, i.e. an input x such that $\mathcal{A}^T_{\mathsf{FS}}(x) \neq \mathcal{A}_{\mathsf{PCP}}(x)$. The witness of $\mathcal{A}^T_{\mathsf{FS}}(x)$ will be the truth-table of the desired hard function.

Construction of Algorithm $\mathcal{A}_{\mathsf{PCP}}$. Our proof will use $\mathcal{A}^T_{\mathsf{FS}}$. We also define another nondeterministic algorithm $\mathcal{A}_{\mathsf{PCP}}$, which tries to simulate $\mathcal{A}^T_{\mathsf{FS}}$ by applying the PCPs from Lemma 3.10.

- Given an input z to $\mathcal{A}^T_{\mathsf{FS}}$, $\mathcal{A}_{\mathsf{PCP}}$ first applies the PCP reduction from Lemma 3.10 to $\mathcal{A}^T_{\mathsf{FS}}(z)$, to compute an AC^0 oracle circuit[15] VPCP_z, which takes $\ell(n) = \log T(n) + O(\log\log T(n)) = C\log n + O(\log\log n)$ random bits as input, and queries an oracle $\mathcal{O}$ which also takes ℓ bits as input. All oracle queries in VPCP_z are projections of the random bits, and VPCP_z satisfies two conditions:
  - If $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$, there exists an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ such that

$$\Pr_{x \in \{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}_z(x) = 1] = 1. \qquad \text{(PCP Completeness)}$$

[15] The top of circuit is actually $\mathsf{AND} \circ \mathsf{OR}_3$.

– If $\mathcal{A}^T_{\mathsf{FS}}(z) = 0$, for all oracles $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$, it follows

$$\Pr_{x\in\{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}_z(x) = 1] \le 1/2. \qquad \text{(PCP Soundness)}$$

- $\mathcal{A}_{\mathsf{PCP}}$ then guesses a $\mathscr{C}$ circuit C of size $s(\ell)$ on ℓ bits input: note that the circuit can be described using $\widetilde{O}(s(\ell)) \le \widetilde{O}(2^{C\varepsilon \log n}) \le n/10$ bits for every sufficiently large n. Feeding C into the PCP oracle circuit VPCP_z, we obtain VPCP^C_z, which is an $\mathsf{AND} \circ \mathsf{OR}_3 \circ \mathscr{C}$ circuit of size $\mathrm{poly}(|C|)$.

- Finally, $\mathcal{A}_{\mathsf{PCP}}$ uses the non-trivial Gap-UNSAT algorithm from the assumption to distinguish between (1) $\Pr_{x\in\{0,1\}^\ell}[\mathsf{VPCP}^C_z(x) = 1] = 1$ and (2) $\Pr_{x\in\{0,1\}^\ell}[\mathsf{VPCP}^C_z(x) = 1] \le 1/2$. $\mathcal{A}_{\mathsf{PCP}}$ accepts if case (1) holds and rejects otherwise.

  Crucially, by the construction it follows that VPCP^C_z is an $\mathsf{AND} \circ \mathsf{OR} \circ \mathscr{C}$ circuits of size $\le s(\ell)^K$ on ℓ input bits. The Gap-UNSAT algorithm from the assumption runs $2^{\ell - \omega(\log \ell)} \le o(T(n))$ time.

Therefore, we can see that $\mathcal{A}_{\mathsf{PCP}}$ runs in $o(T(n))$ time while guessing no more than $n/10$ bits. Hence, the refuter $\mathcal{R}^T$ from Theorem 4.6 can be applied to find a differing input between it and $\mathcal{A}^T_{\mathsf{FS}}$.

The following observation is crucial in our proof. It basically says that $\mathcal{A}_{\mathsf{PCP}}$ makes only "one-sided" errors.

Lemma 5.1. *For every z, $\mathcal{A}_{\mathsf{PCP}}(z) \le \mathcal{A}^T_{\mathsf{FS}}(z)$.*

Proof. The statement is equivalent to $\mathcal{A}^T_{\mathsf{FS}}(z) = 0$ implies $\mathcal{A}_{\mathsf{PCP}}(z) = 0$, which follows directly from the soundness condition of the PCP. □

The Hard Language $\mathcal{A}_{\mathsf{HARD}}$. Now we are ready to finish the proof. We will construct a hard language in $\mathcal{A}_{\mathsf{HARD}} \in \mathsf{E}^{\mathsf{NP}}$, as follows: on an input y of length m, we first construct (in E^{NP}) a function $f_m\colon \{0,1\}^m \to \{0,1\}$ (which is only related to m, but independent of y) such that f_m is hard against $\mathscr{C}$ circuits of size $s(m/2)$. Then we output $f_m(y)$. The function f_m is constructed as follows.

1. Let $n = 2^{m/2C}$. Applying the refuter $\mathcal{R}^T$ of Theorem 4.6, we can find in $\mathrm{poly}(T(n))$ time with a SAT oracle a string $z \in \{0,1\}^n$ such that $\mathcal{A}^T_{\mathsf{FS}}(z) \ne \mathcal{A}_{\mathsf{PCP}}(z)$. By Lemma 5.1, it must be the case that $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$ while $\mathcal{A}_{\mathsf{PCP}}(z) = 0$.

2. Consider the oracle circuit VPCP_z constructed by the PCP reduction. Since $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$, there is an oracle $\mathcal{O}$ such that $\mathsf{VPCP}^{\mathcal{O}}_z$ is a tautology by the completeness of the PCP. We can find the lexicographically first such oracle $\mathcal{O}$ in $2^{O(\ell(n))} = \mathrm{poly}(T(n))$ time: using a SAT oracle to fix the oracle $\mathcal{O}$ bit by bit, each time we can check if there is a way to complete remaining bits of the oracle $\mathcal{O}$ such that $\mathsf{VPCP}^{\mathcal{O}}_z$ is a tautology.

   Note that $\mathcal{O}$ has $\ell(n) = C\log n + O(\log\log n) \in [m/2, m]$ inputs. Since $\mathcal{A}_{\mathsf{PCP}}(z) = 0$, it also follows that $\mathcal{O}$ does not have $\mathscr{C}$ circuits of size $s(\ell) \ge s(m/2)$: otherwise, $\mathcal{O}$ could be guessed by $\mathcal{A}_{\mathsf{PCP}}$ and therefore $\mathcal{A}_{\mathsf{PCP}}(z)$ would be 1, a contradiction.

3. Finally, the hard function $f_m\colon \{0,1\}^m \to \{0,1\}$ is defined as $f_m(y) = \mathcal{O}(y_{\le \ell})$, where $y_{\le\ell}$ is the prefix of y of length ℓ. Note that f_m does not have $\mathscr{C}$ circuits of size $s(m/2)$.

To summarize, $\mathcal{A}_{\mathsf{HARD}}$ runs in $\mathsf{DTIME}[\mathrm{poly}(T(n))]^{\mathsf{NP}} = \mathsf{DTIME}[2^{O(m)}]^{\mathsf{NP}}$, and for large enough m, $\mathcal{A}_{\mathsf{HARD}}$ restricted to m-length inputs does not have $s(m/2)$-size $\mathscr{C}$ circuits, which completes the proof. □

6 Almost-Everywhere Strong Average-Case Lower Bounds

In this section, we prove the main theorem (Theorem 1.2) of the paper: Non-trivial CAPP algorithms for $\mathscr{C}$ imply strong almost-everywhere average-case lower bounds against $\mathscr{C}$ circuits. In addition to the refuter construction from Theorem 4.6, the proof also carefully combines the new XOR Lemma and PCPs of Proximity.

6.1 An Algorithm for the Average-of-Product of $\mathsf{Sum} \circ \mathscr{C}$ Circuits

First, it will be useful to define the following Average-of-Product problem, variants of which are also studied in [Wil18b, CW19, CR20].

Definition 6.1. Given $k, n \in \mathbb{N}$, the Average-of-Product problem takes as input k functions $f_1, f_2, \ldots, f_k \colon \{0,1\}^n \to \{0,1\}$ (may be implicitly given), and the task is to compute $\mathbb{E}_{x \in \{0,1\}^n}[\prod_{i=1}^{k} f_i(x)]$.

We need the following lemma showing that a non-trivial CAPP algorithm for $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be used to solve the Average-of-Product problem for four $\mathsf{Sum} \circ \mathscr{C}$ circuits.

Lemma 6.2. *Let $\mathscr{C}$ be a typical circuit class, and let $S, E \colon \mathbb{N} \to \mathbb{N}$ be such that $S(n) \leq o(E(n))$. Suppose there is an algorithm solving* CAPP *on $\mathsf{AND}_4 \circ \mathscr{C}$ circuits of size $E(n)$ and n inputs within an additive error of $1/E(n)$, running in $2^n/E(n)$ time. Then given four $\mathsf{Sum} \circ \mathscr{C}$ circuits $C_1, \ldots, C_4 \colon \{0,1\}^n \to \mathbb{R}$, each of complexity at most $S(n)$, the Average-of-Product of $\{C_i(x)\}_{i \in [4]}$ can be computed within an additive error of $S(n)^4/E(n)$ in $O\left(S(n)^4 \cdot 2^n/E(n)\right)$ time.*

Proof. Let $C_i = \sum_{j=1}^{m_i} \alpha_{i,j} \cdot C_{i,j}$. From the assumption, it follows that each $C_{i,j}$ is of at most $S(n)$ size, $m_i \leq S(n)$, and $\sum_{j=1}^{m_i} |\alpha_{i,j}| \leq S(n)$. By adding some dummy coefficients and dummy circuits, we can assume without loss of generality that all the m_i are equal to $m \leq S(n)$.

For each tuple $(j_1, j_2, j_3, j_4) \in [m]^4$, we run the promised CAPP algorithm $\mathcal{A}_{\mathsf{CAPP}}$ on $\bigwedge_{i \in [4]} C_{i,j_i}$ to output an estimate $\mathcal{A}_{\mathsf{CAPP}}(\bigwedge_{i \in [4]} C_{i,j_i})$ such that

$$\left| \mathcal{A}_{\mathsf{CAPP}}\left(\bigwedge_{i \in [4]} C_{i,j_i} \right) - \mathop{\mathbb{E}}_{x} \left[\prod_{i=1}^{4} C_{i,j_i}(x) \right] \right| \leq 1/E(n).$$

Hence, we can estimate $\mathbb{E}_x[\prod_{i=1}^{4} C_i(x)]$ by computing the quantity

$$\sum_{(j_1,j_2,j_3,j_4) \in [m]^4} \mathcal{A}_{\mathsf{CAPP}}\left(\bigwedge_{i \in [4]} C_{i,j_i} \right) \cdot \prod_{i=1}^{4} \alpha_{i,j_i}. \tag{8}$$

The error can be bounded by

$$\begin{aligned}
&\left| \sum_{(j_1,j_2,j_3,j_4) \in [m]^4} \mathcal{A}_{\mathsf{CAPP}}\left(\bigwedge_{i \in [4]} C_{i,j_i} \right) \cdot \prod_{i=1}^{4} \alpha_{i,j_i} - \sum_{(j_1,j_2,j_3,j_4) \in [m]^4} \mathop{\mathbb{E}}_{x} \left[\prod_{i=1}^{4} C_{i,j_i}(x) \right] \cdot \prod_{i=1}^{4} \alpha_{i,j_i} \right| \\
&\leq \sum_{(j_1,j_2,j_3,j_4) \in [m]^4} \prod_{i=1}^{4} |\alpha_{i,j_i}| \cdot E(n)^{-1} \\
&\leq E(n)^{-1} \cdot \prod_{i=1}^{4} \left(\sum_{j=1}^{m} |\alpha_{i,j}| \right) \\
&\leq S(n)^4/E(n).
\end{aligned}$$

Computing (8) can be done in $O(2^n/E(n)\cdot m^4) \le O(2^n/E(n)\cdot S(n)^4)$ time, which completes the proof. □

6.2 Main Theorem

Now we are ready to prove Theorem 1.2 (restated below).

Reminder of Theorem 1.2. *Let $\mathscr{C}$ be typical. Suppose there is an $\varepsilon > 0$ such that CAPP of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits can be deterministically solved in 2^{n-n^ε} time. Then there is a language $L \in \mathsf{E}^{\mathsf{NP}}$ and a constant $\delta > 0$ such that, for every sufficiently large n, L_n cannot be $(1/2+2^{-n^\delta})$-approximated by $\mathscr{C}$ circuits of size 2^{n^δ}.*

Combining Theorem 1.2 and the 2^{n-n^ε}-time #SAT algorithm for 2^{n^ε}-size $\mathsf{AC}^0_d[m]\circ\mathsf{THR}$ circuits from Theorem 3.12, the following corollary is immediate.

Reminder of Corollary 1.3. *For every $d, m \in \mathbb{N}$, there is an $\varepsilon = \varepsilon_{d,m}$ and $L \in \mathsf{E}^{\mathsf{NP}}$ such that L_n cannot be $(1/2+2^{-n^\varepsilon})$-approximated by $\mathsf{AC}^0_d[m]\circ\mathsf{THR}$ circuits of 2^{n^ε} size, for every sufficiently large n.*

Proof Outline. The reminder of this subsection is devoted to proving Theorem 1.2. To make the presentation clear, we will state and use several technical lemmas during the proof, and defer their proof to the end of the section.

The proof is similar to the proof of Theorem 1.1, but is much more complicated. We again consider the diagonalizing language $\mathcal{A}^T_{\mathsf{FS}}$ from Definition 4.2, and design an algorithm $\mathcal{A}_{\mathsf{PCPP}}$ which tries to speed up $\mathcal{A}^T_{\mathsf{FS}}$. $\mathcal{A}_{\mathsf{PCPP}}$ combines the PCP of Lemma 3.10 and the PCPP from Lemma 3.11. $\mathcal{A}_{\mathsf{PCPP}}$ guesses a small circuit as proof of the PCP just as in Theorem 1.1, as well as additional $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits as proof for the corresponding PCPP. Then $\mathcal{A}_{\mathsf{PCPP}}$ applies the CAPP algorithm for $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits to verify the proof in $o(T(n))$ time. Setting the parameters carefully puts the language of $\mathcal{A}_{\mathsf{PCPP}}$ into $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$.

Then one can use the refuter $\mathcal{R}^T$ from Theorem 4.6 to find a conflicting input z_n between $\mathcal{A}^T_{\mathsf{FS}}$ and $\mathcal{A}_{\mathsf{PCPP}}$ for every sufficiently large input length n. Given such z_n, we show how to construct a function f which cannot be ℓ_1-approximated within a small constant by low-complexity $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits. Finally we apply Lemma 3.8 to construct the required average-case hard function.

6.2.1 Construction of the Algorithm $\mathcal{A}_{\mathsf{PCPP}}$

We now design an algorithm $\mathcal{A}_{\mathsf{PCPP}}$ which tries to speed up $\mathcal{A}^T_{\mathsf{FS}}$. In the following we set $k = 1/\varepsilon$, and $T(n) = 2^{\log^k n}$.

Given an input z to algorithm $\mathcal{A}^T_{\mathsf{FS}}$, $\mathcal{A}_{\mathsf{PCPP}}$ first applies the PCP reduction from Lemma 3.10 to obtain an oracle circuit VPCP_z. This step is the same as the first step of Theorem 1.1. For brevity, we just write the oracle circuit as VPCP from now on. Let $\ell = \log^k n + O(\log\log n)$ be the length of the input to VPCP and its oracle. We also set $\delta = \frac{(s_{\mathsf{pcpp}}-c_{\mathsf{pcpp}})^2}{10^6}$, where s_{pcpp} and c_{pcpp} are the corresponding parameters in Lemma 3.11.

$\mathcal{A}_{\mathsf{PCPP}}$ guesses a (general, fan-in 2) circuit $C\colon \{0,1\}^\ell \to \{0,1\}$ of size at most $2^{\ell^{\varepsilon/4}}$. Setting this circuit as the oracle in VPCP, $\mathcal{A}_{\mathsf{PCPP}}$ obtains a composed circuit $\mathsf{VPCP}^C\colon \{0,1\}^\ell \to \{0,1\}$. By Lemma 3.10 and similar reasoning as in the proof of Theorem 1.1, we have the following claim.

Claim 1. *The following statements hold.*

1. *If $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$, then there an oracle $\mathcal{O}$ such that $\mathsf{VPCP}^{\mathcal{O}}(x) = 1$ for every $x \in \{0,1\}^\ell$.*

2. *It* $\mathcal{A}_{\mathsf{FS}}^{T}(z) = 0$*, then for every oracle* $\mathcal{O}$*, it holds that* $\Pr_{x\in\{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}(x) = 1] \le \frac{1}{n^{10}} \le \frac{1}{\ell^{10}}$.

$\mathcal{A}_{\mathsf{PCPP}}$ attempts to distinguish the two cases in Claim 1.

Notation, and the PCPP Reduction. Then $\mathcal{A}_{\mathsf{PCPP}}$ applies the PCPP reduction from lemma 3.11 to the circuit $\mathsf{VPCP}^{C}(\cdot)$. The PCPP reduction gives us a 2-SAT instance over variables $\mathcal{Y} \cup \mathcal{Z}$ with $m = 2^{\ell^{\varepsilon/3}}$ clauses. Let $\{\mathsf{Cons}_i\}_{i=1}^{m}$ be the set of clauses[16], where each clause is an OR of two variables in $\mathcal{Y} \cup \mathcal{Z}$ or their negations. For $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, we use $\mathcal{Y}_s$ and $\mathcal{Z}_t$ to denote the s-th variable in $\mathcal{Y}$ and the t-th variable in $\mathcal{Z}$, respectively.

To elegantly discuss the algorithm and its analysis, we introduce some useful notation. For each clause Cons_i, it extends to a degree-2 polynomial, denoted as $\widetilde{\mathsf{Cons}_i}$.[17]

1. By a "real-valued proof" we mean a pair of two lists of proof functions (Y, Z) for PCPP, where $Y = (Y_s)_{s\in[|\mathcal{Y}|]}$, $Z = (Z_t)_{t\in[|\mathcal{Z}|]}$ and each Y_s and Z_t is a function from $\{0,1\}^\ell \to \mathbb{R}$. Based on (Y, Z), we define the following terminologies:

   - Recall each clause Cons_i involves two variables. We define indicators $T_{i1}^{(Y,Z)}$ and $T_{i2}^{(Y,Z)}$ to indicate the corresponding functions in (Y, Z). We also let $\mathsf{From}(T_{ij}^{(Y,Z)}) \in \mathcal{Y} \cup \mathcal{Z}$ be the variable it corresponds to.
   - Each clause Cons_i extends to a polynomial $\widetilde{\mathsf{Cons}_i}$, we define $F_i^{(Y,Z)} := \widetilde{\mathsf{Cons}_i}(T_{i1}^{(Y,Z)}, T_{i2}^{(Y,Z)})$.

   Note that these objects all depend on the given proof (Y, Z), when the context is clear, we also omit the superscript, and simply write them as T_{ij} and F_i.

2. By a "Boolean-valued proof" we mean a pair of two lists of proof functions $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ where $\widehat{Y}_s(x) = \mathsf{Enc}_s(x)$ for every $x \in \{0,1\}^\ell$ and $s \in [|\mathcal{Y}|]$, $\widehat{Z} = (\widehat{Z}_t)_{t\in[|\mathcal{Z}|]}$, and each $\widehat{Z}_t$ is function from $\{0,1\}^\ell \to \{0,1\}$. Recall that $\mathsf{Enc}\colon \{0,1\}^\ell \to \{0,1\}^{|\mathcal{Y}|}$ is the fixed $\mathbb{F}_2$-linear error correcting code used in Lemma 3.11. Similar to the case of real-valued proofs, the proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ induces $\widehat{T}_{ij}^{(\widehat{Y},\widehat{Z})}$ and $\widehat{F}_i^{(\widehat{Y},\widehat{Z})}$. When the context is clear, we omit the superscript and write them as $\widehat{T}_{ij}$ and $\widehat{F}_i$.

To clarify, we always use (Y, Z) to denote a real-valued proof, and $(\widehat{Y}, \widehat{Z})$ to denote a Boolean-valued proof.

From Lemma 3.11, we have the following claim.

Claim 2. *The following statements hold.*

1. *If* VPCP^{C} *is a tautology, then there is a Boolean proof* $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ *such that*

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} \mathop{\mathbb{E}}_{i\in[m]} \widehat{F}_i(x) \ge c_{\mathsf{pcpp}}.$$

2. *If* $\mathsf{VPCP}^{C}(x) = 1$ *for at most a* $\frac{1}{\ell^{10}} = o(1)$ *fraction of* x*, then for every sufficiently large* n*, for every Boolean proof* $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$*, we have*

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} \mathop{\mathbb{E}}_{i\in[m]} \widehat{F}_i(x) < c_{\mathsf{pcpp}} - \frac{9}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).$$

[16] Cons_i is also called "constraints", we use "clauses" and "constraints" interchangeably.

[17] We use the natural arithmetization: The Boolean 0 (false) and 1 (true) correspond to real 0 and 1, respectively. Boolean AND corresponds to real multiplication. Boolean OR corresponds to the real polynomial $\mathsf{OR}(a, b) = 1 - (1 - a) \cdot (1 - b)$.

Guessing Succinct $\mathsf{Sum} \circ \mathscr{C}$ Circuits. Then $\mathcal{A}_{\mathsf{PCPP}}$ guesses a list of proof circuits $Y_i(x), Z_j(x)$ such that $Y_i, Z_j \in (\mathsf{Sum} \circ \mathscr{C})[2^{\ell^{\varepsilon/2}}]$ for every $i \in [|\mathcal{Y}|]$ and $j \in [|\mathcal{Z}|]$.

Note that ideally we only want to consider $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits, but it is not clear how to verify that a given $\mathsf{Sum} \circ \mathscr{C}$ circuit satisfies the $[0,1]\mathsf{Sum} \circ \mathscr{C}$ promise. Therefore, $\mathcal{A}_{\mathsf{PCPP}}$ applies a certain validity test on the guessed proof (Y, Z), and reject immediately if the test fails. Although passing the test does not guarantee Y, Z are $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits, it does mean they are "close enough" to them so that our analysis still goes through. The validity test will be described later.

Since $F_i(x) := \widetilde{\mathsf{Cons}}(T_{i1}(x), T_{i2}(x))$ is a degree-2 polynomial, evaluating $\mathbb{E}_x[F(x)]$ reduces to Average-of-Product problems over (T_{i1}, T_{i2}). By Lemma 6.2 and the assumed CAPP algorithm for 2^{ℓ^ε}-size $\mathsf{AND}_4 \circ \mathscr{C}$ circuits, we can estimate $\mathbb{E}_x[F_i(x)]$ within an additive error of $2^{O(\ell^{\varepsilon/2})-\Omega(\ell^\varepsilon)} \le o(1)$ in $2^{\ell-\ell^\varepsilon+O(\ell^{\varepsilon/2})}$ time.

Let $\widetilde{\mathbb{E}}_x(F_i)$ be the output of the estimation algorithm when evaluating $\mathbb{E}_x[F_i(x)]$. $\mathcal{A}_{\mathsf{PCPP}}$ accepts if and only if

$$\mathop{\mathbb{E}}_{i\in[m]} \widetilde{\mathbb{E}}_x(F_i) \ge c_{\mathsf{pcpp}} - \frac{5}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}). \tag{9}$$

Putting everything together, $\mathcal{A}_{\mathsf{PCPP}}$ takes $m \cdot 2^{\ell-\ell^\varepsilon+O(\ell^{1/2})} = 2^{\ell-\ell^\varepsilon+O(\ell^{1/2})+\ell^{1/3}} = 2^{\ell-\Omega(\ell^\varepsilon)} \le o(T(n))$ time after guessing the circuits.

This completes the construction of $\mathcal{A}_{\mathsf{PCPP}}$ except for the validity test, which is described next.

6.2.2 Validity Test on Guessed $\mathsf{Sum} \circ \mathscr{C}$ Circuits

For every $T_{ij}(x)$, consider the function

$$P_{ij}(x) = \begin{cases} T_{ij}(x)^2(1-T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij}) \in \mathcal{Z}, \\ (\mathsf{Enc}_s(x) - T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij}) = \mathcal{Y}_s \text{ for } s \in [|\mathcal{Y}|], \end{cases} \tag{10}$$

and $Q_{ij}(x) = T_{ij}(x)^2$. We want to estimate the expectations

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} P_{ij}(x) \text{ and } \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} Q_{ij}(x) \text{ for each } i,j \in [m]\times[2].$$

It is clear that for every $(i,j) \in [m]\times[2]$, $Q_{ij}(x)$ is a polynomial over $T_{ij}(x)$ of degree 4. For $P_{ij}(x)$, it is also a degree-4 polynomial over $T_{ij}(x)$ when $\mathsf{From}(T_{ij}) \in \mathcal{Z}$. Hence, in these two cases, the evaluation of these expectations reduces to computing Average-of-Product problems for the $T_{i,j}$, which can in turn be estimated by Lemma 6.2. When $\mathsf{From}(T_{ij}) = \mathcal{Y}_s$, $\mathsf{Enc}_s(x)$ depends on at most $\frac{\ell}{2}$ bits (the moreover part of Lemma 3.11). We can then enumerate all these bits, and solve the Average-of-Product problem on the remaining part of inputs to estimate $\mathbb{E}_{x\in\{0,1\}^\ell} P_{ij}(x)$.[18] This test runs in time

$$2^{\ell^{\varepsilon/3}} \cdot \left(2^{\ell-\ell^\varepsilon+O(\ell^{\varepsilon/2})} + 2^{\ell/2} \cdot 2^{(\ell/2)-(\ell/2)^\varepsilon+O(\ell^{\varepsilon/2})}\right) = o(T(n)).$$

Recall that we use $\widetilde{\mathbb{E}}_x(D)$ to denote the estimation of $\mathbb{E}_x[D(x)]$. The proof (Y, Z) passes the test if and only if both of the following conditions hold:

1. $\mathbb{E}_{i,j\in[m]\times[2]} \widetilde{\mathbb{E}}_x(P_{ij}) \le 2\delta$.
2. $\widetilde{\mathbb{E}}_x(Q_{ij}) \le 1+\delta$ for every $(i,j) \in [m]\times[2]$.

$\mathcal{A}_{\mathsf{PCPP}}$ rejects immediately if (Y, Z) does not pass the test.

[18] After fixing all the bits that $\mathsf{Enc}_s(x)$ depends on, $\mathsf{Enc}_s(x)$ can be replaced by a constant, so $P_{ij}(x)$ becomes a degree-4 polynomial over T_{ij}.

6.2.3 Analysis of the Algorithm $\mathcal{A}_{\mathsf{PCPP}}$

Now we prove several lemmas and facts about the algorithm $\mathcal{A}_{\mathsf{PCPP}}$.

Completeness and Soundness of the Validity Test. Recall that we have defined $F_i(x) := \widetilde{\mathsf{Cons}_i}(T_{i1}(x), T_{i2}(x))$, where $\widetilde{\mathsf{Cons}_i}$ is the polynomial extension of Cons_i. For a Boolean-valued proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, we also defined $\widehat{F}_i(x) := \widetilde{\mathsf{Cons}_i}(\widehat{T}_{i1}(x), \widehat{T}_{i2}(x))$. The following lemma summarizes the properties we need from the validity test. We defer its proof to the end of this section.

Lemma 6.3. *We have the following completeness and soundness conditions on the validity test.*

1. ***Completeness.*** *Every (Y, Z) satisfying the following conditions passes the test:*

   (1.a) For every $(s,t) \in [|\mathcal{Y}|] \times [|\mathcal{Z}|]$ and every input $x \in \{0,1\}^n$, $Y_s(x), Z_t(x) \in [0,1]$.

   (1.b) There is a Boolean-valued proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z}(x))$ such that

   $$\mathop{\mathbb{E}}_{i,j \in [m] \times [2]} \left\| T_{ij} - \widehat{T}_{ij} \right\|_1 \leq \delta.$$

2. ***Soundness.*** *If (Y, Z) passes the test, the following statements hold.*

   (2.a) There is a Boolean-valued proof $(\widehat{Y}(x) = \mathsf{Enc}(x), \widehat{Z}(x))$ such that

   $$\mathop{\mathbb{E}}_{i,j \in [m] \times [2]} \left\| T_{ij} - \widehat{T}_{ij} \right\|_2 \leq \sqrt{12\delta}.$$

   (2.b) For any Boolean-valued proof $(\widehat{Y}(x) = \mathsf{Enc}(x), \widehat{Z}(x))$, for every $i \in [m]$, it holds that

   $$\left\| F_i - \widehat{F}_i \right\|_1 \leq 6 \cdot \mathop{\mathbb{E}}_{j \in [2]} \left\| T_{ij} - \widehat{T}_{ij} \right\|_2.$$

The Running Time and Witness Length of $\mathcal{A}_{\mathsf{PCPP}}$. Putting everything together, $\mathcal{A}_{\mathsf{PCPP}}$ runs in $o(T(n))$ time and guesses at most $2^{\ell^{\varepsilon/4}} + 2^{O(\ell^{\varepsilon/2})} \leq o(n)$ bits (recall that $\ell = \log^k n + O(\log \log n)$ and $k = \varepsilon^{-1}$), so $\mathcal{A}_{\mathsf{PCPP}}$ is an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm.

Soundness of $\mathcal{A}_{\mathsf{PCPP}}$. We also verify that for every sufficiently long input z, $\mathcal{A}_{\mathsf{PCPP}}$ never outputs 1 when $\mathcal{A}^T_{\mathsf{FS}}(z) = 0$. The proof is deferred to later subsections.

Lemma 6.4. *For all but finitely many z, it holds that $\mathcal{A}_{\mathsf{PCPP}}(z) \leq \mathcal{A}^T_{\mathsf{FS}}(z)$.*

6.2.4 Average-Case Hardness from Conflicting Inputs

Since $\mathcal{A}_{\mathsf{PCPP}} \in \mathsf{NTIMEGUESS}[o(T(n)), n/10]$, we can apply the refuter $\mathcal{R}^T$ from Theorem 4.6 to it. That is, for every sufficiently large n, we can find in $O(T(n) \cdot n)$ time (with a SAT oracle) an input z of length n such that $\mathcal{A}_{\mathsf{PCPP}}(z) \neq \mathcal{A}^T_{\mathsf{FS}}(z)$. By Lemma 6.4, it must be the case that $\mathcal{A}_{\mathsf{PCPP}}(z) = 0$ and $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$.

Depending on whether there is a circuit C of $2^{\ell^{\varepsilon/4}}$ size such that $\mathsf{VPCP}^C(x)$ is a tautology (this can be checked with a call to a SAT oracle in $2^{O(\ell)}$ time), we consider the following two cases.

Case 1. There is no circuit of $2^{\ell^{\varepsilon/4}}$ size such that $\mathsf{VPCP}^C(x)$ is a tautology. In this case, we simply find the truth table of the lexicographically first oracle $f\colon \{0,1\}^\ell \to \{0,1\}$ (with a SAT oracle) such that $\mathsf{VPCP}^f(x)$ is a tautology. Such an f exists by Item (1) of Claim 1 and the fact that $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$. Note that f does not have $2^{\ell^{\varepsilon/4}}$-size circuits, from the assumption of Case 1.

By Lemma 3.9, we can construct from f a new function $f_{\mathsf{amp}}\colon \{0,1\}^{O(\ell)} \to \{0,1\}$, which cannot be $(\frac{1}{2} + 2^{-\ell^{\Omega(1)}})$-approximated by circuits of size $2^{\ell^{\Omega(1)}}$.

Case 2. There is a circuit of size $2^{\ell^{\varepsilon/4}}$ such that $\mathsf{VPCP}^C(x)$ is a tautology. Then given access to a SAT oracle, in $2^{O(\ell)}$ time we can find the lexicographically first such circuit C and the first Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ such that

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} \widehat{F}_i(x) \geq c_{\mathsf{pcpp}}.$$

The Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$ above exists by Item (1) of Claim 2. Let $r = \log m = \ell^{\varepsilon/3}$. Consider the function $f\colon \{0,1\}^{r+1+\ell} \to \{0,1\}$ defined as

$$f(i,j,u) := \widehat{T}_{ij}(u) \text{ for } (i,j,u) \in [m] \times \{0,1\} \times \{0,1\}^\ell.$$

In the above, we identify $\{0,1\}^r$ with $[2^r] = [m]$ in the natural way, so that $i \in [m]$ can also be interpreted as a string in $\{0,1\}^r$. The following lemma shows that f cannot be approximated well by $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits of low complexity. We defer the proof to the next subsection, as it involves straightforward (but heavy) calculations.

Lemma 6.5. *For every sufficiently large n, in Case 2, the function f defined above is δ-far from every $[0,1]\mathsf{Sum} \circ \mathscr{C}[2^{\ell^{\varepsilon/4}}]$ circuit. That is, for every $[0,1]\mathsf{Sum} \circ \mathscr{C}[2^{\ell^{\varepsilon/4}}]$ circuit D, it holds that $\|D - f\|_1 > \delta$.*

This lemma shows that f is δ-far (in ℓ_1 distance) from $[0,1]\mathsf{Sum} \circ \mathscr{C}[2^{\ell^{\varepsilon/4}}]$ circuits. Using Lemma 3.8, we conclude that $f^{\oplus d}$ cannot be $\left(\frac{1}{2} + (1-\delta)^{d-1}\right)$-approximated by $\mathscr{C}$ circuits of $2^{\ell^{\varepsilon/4} - O_\delta(d)}$ size. Setting $d = \ell^{\varepsilon/5}$, it follows that the function $f^{\oplus d}$, taking $O(\ell^{1+\varepsilon/5})$ bits of input, cannot be $\left(\frac{1}{2} + 2^{-\ell^{\Omega(1)}}\right)$-approximated by $\mathscr{C}$ circuits of size $2^{\ell^{\Omega(1)}}$.

6.2.5 The Hard Language $\mathcal{A}_{\mathsf{avgHARD}}$

Finally, we are ready to design the hard language $\mathcal{A}_{\mathsf{avgHARD}}$. On an input y of length m, let $n = 2^{m^{1/3k}}$. Applying the refuter $\mathcal{R}^T$ from Theorem 4.6, we can find in $\mathrm{poly}(T(n))$ time with a SAT oracle a string $z \in \{0,1\}^n$ such that $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$ while $\mathcal{A}_{\mathsf{PCPP}}(z) = 0$.

Recall that $\ell = \ell(n) = \log^k n + O(\log\log n)$. Using a SAT oracle, we can decide whether there is a circuit C of $2^{\ell^{\varepsilon/4}}$ size such that VPCP^C is a tautology in $2^{O(\ell)}$ time. From the discussion above, in both cases we can construct an average-case hard function f on inputs of length $\ell' \leq O(\max(\ell, \ell^{1+\varepsilon/5})) \leq \ell^2$ in $2^{O(\ell)}$ time with a SAT oracle, such that f cannot be $(1/2 + 2^{-\ell^{\Omega(1)}})$-approximated by $2^{\ell^{\Omega(1)}}$-size $\mathscr{C}$ circuits.

Note that $\ell = \log^k n + O(\log\log n) \leq m^{1/2}$. Therefore, we can pad the input length to m in the natural way. That is, we define $g\colon \{0,1\}^m \to \{0,1\}$ such that $g(y) = f(y')$ where y' is the first ℓ' bits of y. It follows that g cannot be $\left(1/2 + 2^{-m^{\Omega(1)}}\right)$-approximated by $\mathscr{C}$ circuits of size $2^{\ell^{\Omega(1)}} = 2^{m^{\Omega(1)}}$. Therefore, $\mathcal{A}_{\mathsf{avgHARD}}$ simply outputs $g(y)$ on the input y. Note that it runs in $2^{O(\ell)} \leq 2^{O(m)}$ time with a SAT oracle, which completes the proof of Theorem 1.2.

6.3 Proof of Lemma 6.4 and Lemma 6.5

Now we present the proof of Lemma 6.4.

Proof of Lemma 6.4. Fix a sufficiently long input z and assume $\mathcal{A}^T_{\mathsf{FS}}(z) = 0$. To prove the lemma, it suffices to show that $\mathcal{A}_{\mathsf{PCPP}}(z) = 0$ as follows.

Consider the PCP system VPCP in $\mathcal{A}_{\mathsf{PCPP}}(z)$. Suppose that $\mathcal{A}_{\mathsf{PCPP}}$ guessed a circuit C as the oracle for VPCP and (Y, Z) as its proof circuits. By Item (2) of Claim 1, VPCP^C outputs 1 on at most a $\frac{1}{\ell^{10}}$ fraction of inputs. In the following we assume (Y, Z) passes the validity test, as otherwise $\mathcal{A}_{\mathsf{PCPP}}$ immediately rejects.

For every Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, by Item (2) of Claim 2, we have

$$\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} \widehat{F}_i(x) < c_{\mathsf{pcpp}} - \frac{9}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).$$

By Item (2.a) of Lemma 6.3, for some Boolean proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, it holds that

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 \le \sqrt{12\delta}.$$

By Item (2.b) of Lemma 6.3, it follows that

$$\begin{aligned}
\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} F_i(x) &\le \mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} \widehat{F}_i(x) + \mathop{\mathbb{E}}_{i\in[m]} \left\|\hat{F}_i - F_i\right\|_1 \\
&\le \mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} \widehat{F}_i(x) + 6 \cdot \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 \\
&< c_{\mathsf{pcpp}} - \frac{7}{10}\left(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}\right). \qquad (11)
\end{aligned}$$

The last inequality holds since we set $\delta = \frac{(s_{\mathsf{pcpp}} - c_{\mathsf{pcpp}})^2}{10^6}$ and $6 \cdot \sqrt{12\delta} \le \frac{1}{5} \cdot (c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}})$.

Recall that we use $\widetilde{\mathbb{E}}_x(F_i)$ to denote the output of the estimation algorithm on $\mathbb{E}_x[F_i(x)]$. Since T_{i1} and T_{i2} are $\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity $2^{O(\ell^{\varepsilon/2})}$, by Lemma 6.2 it follows that

$$\left| \mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} F_i(x) - \mathop{\mathbb{E}}_{i\in[m]} \widetilde{\mathbb{E}}_x(F_i) \right| \le o(1) \le \frac{1}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}). \qquad (12)$$

By (11) and (12), $\mathcal{A}_{\mathsf{PCPP}}$ rejects on (Y, Z). Therefore, $\mathcal{A}_{\mathsf{PCPP}}(z) = 0$ since it rejects every C and (Y, Z), and the conclusion follows. □

Next we prove Lemma 6.5.

Proof of Lemma 6.5. Suppose there is a circuit $D \in [0,1]\mathsf{Sum} \circ \mathscr{C}[2^{\ell^{\varepsilon/4}}]$ such that $\left\|D - f\right\|_1 \le \delta$. We are going to construct a proof (Y, Z) which makes $\mathcal{A}_{\mathsf{PCPP}}$ accept, contradicting $\mathcal{A}_{\mathsf{PCPP}}(z) = 0$.

Construction of the Proof (Y, Z) Making $\mathcal{A}_{\mathsf{PCPP}}$ Accept. Similar to f, we also think of D as a function on $\{0,1\}^r \times \{0,1\} \times \{0,1\}^\ell$, and often use i and j to denote the first two parts of input. Recall that we identify $[m]$ with $\{0,1\}^r$.

For $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, we define

$$Y_s(x) := \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=\mathcal{Y}_s} D(i,j,x) \quad \text{and} \quad Z_t(x) := \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=\mathcal{Z}_t} D(i,j,x).$$

Since $m = 2^{\ell^{\varepsilon/3}}$ and for each fixed $(i,j) \in [m] \times [2]$, $D(i,j,\cdot)$ is a $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit of complexity at most $2^{\ell^{\varepsilon/4}}$. It is clear that for every $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, Y_s and Z_t are $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits of complexity at most $2^{\ell^{\varepsilon/2}}$.

We will show that $\mathcal{A}_{\mathsf{PCPP}}(z)$ accepts on the proof (Y,Z) defined above (together with the circuit C as the guessed oracle in VPCP^C). Let T_{ij} and $\widehat{T}_{ij}$ be the indicators for (Y,Z) and $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})$, respectively. (Recall that $\widehat{Y}$ and $\widehat{Z}$ are the lexicographically first proof we constructed in Case 2 of the Section 6.2.4).

(Y,Z) Passes the Validity Test. We first apply Item (1) of Lemma 6.3 to prove that (Y,Z) passes the validity test in $\mathcal{A}_{\mathsf{PCPP}}$. Since Item (1.a) of Lemma 6.3 (all the Y_s and Z_t are $[0,1]$-valued) is already satisfied, it suffices to verify Item (1.b) of Lemma 6.3, which is

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|\widehat{T}_{i,j} - T_{i,j}\right\| \le \delta.$$

We have

$$\begin{aligned}\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 &= \frac{1}{2m} \cdot \sum_{i,j\in[m]\times[2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \\ &= \frac{1}{2m} \cdot \sum_{X\in\mathcal{Y}\cup\mathcal{Z}} \sum_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1. \end{aligned} \tag{13}$$

To bound (13), we need the following simple fact.

Fact 6.6. *For every $(v_1, v_2, \ldots, v_d) \in [0,1]^d$ and $b \in \{0,1\}$, it holds that*

$$\frac{1}{d}\sum_i |v_i - b| = \left|\frac{1}{d}\sum_i v_i - b\right|.$$

To verify the fact is true, note that when $b = 0$, it is equivalent to $\frac{1}{d}\sum_i |v_i| = \left|\frac{1}{d}\sum_i v_i\right|$, which is true since all $v_i \ge 0$. The case for $b = 1$ is symmetric, since all $v_i \le 1$.

By Fact 6.6, for every $X \in \mathcal{Y} \cup \mathcal{Z}$, it follows that

$$\begin{aligned}&\mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \\ &= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \mathop{\mathbb{E}}_{x} \left|T_{ij}(x) - f(i,j,x)\right| && \text{(Definition of } f\text{)} \\ &= \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|T_{ij}(x) - f(i,j,x)\right| \\ &= \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|D(i,j,x) - f(i,j,x)\right| \\ &= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|D(i,j,\cdot) - f(i,j,\cdot)\right\|_1. \end{aligned} \tag{14}$$

The second-last equality above holds by fixing a particular x and setting v as the collection of the $D(i,j,x)$ for all $\mathsf{From}(T_{ij}) = X$ and $b = f(i,j,x)$ (note that b only depends on $\mathsf{From}(T_{i,j})$ since x is fixed), and applying Fact 6.6 to show

$$\begin{aligned}\mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|D(i,j,x) - f(i,j,x)\right| &= \frac{1}{d}\sum_i |v_i - b| \\ &= \left|\frac{1}{d}\sum_i v_i - b\right| \\ &= \mathop{\mathbb{E}}_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left|T_{ij}(x) - f(i,j,x)\right|. \end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 &= \frac{1}{2m} \cdot \sum_{X\in\mathcal{Y}\cup\mathcal{Z}} \sum_{i,j \text{ s.t. } \mathsf{From}(T_{ij})=X} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \\
&= \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|f(i,j,\cdot) - D(i,j,\cdot)\right\|_1 && \text{(by (14))} \\
&= \left\|f - D\right\|_1 \\
&\le \delta. && (15)
\end{aligned}$$

Hence, by (15) and Item (1) of Lemma 6.3, (Y,Z) passes the validity test.

$\mathcal{A}_{\mathsf{PCPP}}$ Accepts (Y,Z). Now we further argue that $\mathcal{A}_{\mathsf{PCPP}}$ accepts (Y,Z). Note that

$$\|T_{ij} - \widehat{T}_{ij}\|_2 \le \|T_{ij} - \widehat{T}_{ij}\|_\infty^{1/2} \cdot \|T_{ij} - \widehat{T}_{ij}\|_1^{1/2} \le \|T_{ij} - \widehat{T}_{ij}\|_1^{1/2}. \tag{16}$$

The first inequality above follows from the fact that $\mathbb{E}_i[a_i^2] \le \max_i |a_i| \cdot \mathbb{E}_i |a_i|$ for every real vector a, and the second inequality is implied by $\|T_{ij} - \widehat{T}_{ij}\|_\infty \le 1$.

Hence, it follows from (15), (16), and Jensen's inequality that

$$\mathop{\mathbb{E}}_{ij\in[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 \le \mathop{\mathbb{E}}_{ij} \|T_{ij} - \widehat{T}_{ij}\|_1^{1/2} \le \left(\mathop{\mathbb{E}}_{ij} \|T_{ij} - \widehat{T}_{ij}\|_1\right)^{1/2} \le \sqrt{\delta}. \tag{17}$$

Finally, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} F_i(x) &\ge \mathop{\mathbb{E}}_i \mathop{\mathbb{E}}_x \widehat{F}_i(x) - \mathop{\mathbb{E}}_i \left\|F_i - \widehat{F}_i\right\|_1 \\
&\ge \mathop{\mathbb{E}}_i \mathop{\mathbb{E}}_x \widehat{F}_i(x) - 6 \cdot \mathop{\mathbb{E}}_{ij} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 && \text{(Item (2.b) of Lemma 6.3)} \\
&\ge \mathop{\mathbb{E}}_i \mathop{\mathbb{E}}_x \widehat{F}_i(x) - 6\sqrt{\delta} && \text{(by (17))} \\
&\ge c_{\mathsf{pcpp}} - \frac{1}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).
\end{aligned}$$

The last inequality follows from $\mathbb{E}_i \mathbb{E}_x \widehat{F}_i(x) \ge c_{\mathsf{pcpp}}$ and $\delta = \frac{(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}})}{10^6}$.

Finally, by (12), it follows that

$$\mathop{\mathbb{E}}_{i\in[m]} \widetilde{\mathbb{E}}_x(F_i) \ge \mathop{\mathbb{E}}_{i\in[m]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} F_i(x) - \delta \ge c_{\mathsf{pcpp}} - \frac{5}{10}(c_{\mathsf{pcpp}} - s_{\mathsf{pcpp}}).$$

It shows that (Y,Z) and C are accepted by $\mathcal{A}_{\mathsf{PCPP}}(z)$, which implies $\mathcal{A}_{\mathsf{PCPP}}(z) = 1$, a contradiction. □

6.4 Proof of Lemma 6.3

Finally, we present the proof of Lemma 6.3. We need the following simple fact.

Lemma 6.7. *For every sufficiently large n and every $(i,j) \in [m] \times [2]$, it holds that*

$$\left|\widetilde{\mathbb{E}}_x(P_{ij}) - \mathop{\mathbb{E}}_x P_{ij}(x)\right| \le \delta \text{ and } \left|\widetilde{\mathbb{E}}_x(Q_{ij}) - \mathop{\mathbb{E}}_x Q_{ij}(x)\right| \le \delta.$$

Proof. This follows from Lemma 6.2 by setting $S(\ell) = 2^{O(\ell^{\varepsilon/2})}$ and $E(\ell) = 2^{\Omega(\ell^\varepsilon)}$. □

Proof of Lemma 6.3. First we establish the completeness condition.

Completeness. Suppose that (Y, Z) satisfies Item (1.a) and (1.b). That is, (1.a) for every $s \in [|\mathcal{Y}|]$ and $t \in [|\mathcal{Z}|]$, Y_s and Z_t are valid $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuits, and (1.b) there is a Boolean-valued proof $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z}(x))$ such that

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \|T_{ij} - \widehat{T}_{ij}\|_1 \le \delta.$$

Recall that $Q_{ij}(x) = T_{ij}(x)^2$, it is clear that $\mathbb{E}_x Q_{ij}(x) \in [0,1]$ since $T_{ij}(x) \in [0,1]$.
Now we consider $P_{ij}(x)$. Recall that

$$P_{ij}(x) = \begin{cases} T_{ij}(x)^2(1 - T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij}) \in \mathcal{Z}, \\ (\mathsf{Enc}(x)_s - T_{ij}(x))^2, & \text{if } \mathsf{From}(T_{ij}) = \mathcal{Y}_s \text{ for } s \in [|\mathcal{Y}|]. \end{cases} \tag{18}$$

Since $T_{ij}(x) \in [0,1]$ and $\widehat{T}_{ij}(x) \in \{0,1\}$, it follows that

$$T_{ij}(x) \cdot (1 - T_{ij}(x)) \le |\widehat{T}_{ij}(x) - T_{ij}(x)|$$

and

$$P_{ij}(x) \le \left(\widehat{T}_{ij}(x) - T_{ij}(x)\right)^2 \le \left|\widehat{T}_{ij}(x) - T_{ij}(x)\right|.$$

Then we have

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} P_{ij}(x) \le \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|\widehat{T}_{ij} - T_{ij}\right\|_1 \le \delta.$$

Hence, by Lemma 6.7,

$$\begin{cases} \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \widetilde{\mathbb{E}}_x(P_{ij}) \le \delta + \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x} P_{ij}(x) \le 2\delta, \\ \widetilde{\mathbb{E}}_x(Q_{ij}) \le \delta + \mathop{\mathbb{E}}_{x} Q_{ij}(x) \le 1 + \delta \text{ for every } (i,j) \in [m] \times [2]. \end{cases}$$

Therefore, (Y, Z) passes the validity test.

Soundness. Now, suppose (Y, Z) passes the test. That is, the following two conditions are satisfied:

$$\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \widetilde{\mathbb{E}}_x(P_{ij}) \le 2\delta \text{ and} \tag{19}$$

$$\widetilde{\mathbb{E}}_x(Q_{ij}) \le 1 + \delta \text{ for every } (i,j) \in [m] \times [2]. \tag{20}$$

In the following we prove Item (2.a) and (2.b) separately.

(2.a) We let $\widehat{Y}$ be determined by $\mathsf{Enc}(x)$, and define $\widehat{Z}$ as

$$\widehat{Z}_i(x) = \begin{cases} 0, & Z_i(x) \le \frac{1}{2} \\ 1, & Z_i(x) > \frac{1}{2} \end{cases}. \tag{21}$$

We need the following claim for the proof.

Claim 3. *For every $(i,j) \in [m] \times [2]$,*

$$(T_{ij}(x) - \widehat{T}_{ij}(x))^2 \le 4 \cdot P_{ij}(x). \tag{22}$$

Let us now prove the claim. Depending on whether $\mathsf{From}(T_{ij}) = \mathcal{Y}_s$ or $\mathsf{From}(T_{ij}) = \mathcal{Z}_t$, there are two cases.

1. ($\mathsf{From}(T_{ij}) = \mathcal{Y}_s$.) In this case,

$$P_{ij}(x) = (\mathsf{Enc}_s(x) - T_{ij}(x))^2 = \left(\widehat{T}_{ij}(x) - T_{ij}(x)\right)^2$$

since $\widehat{T}_{ij}(x) = \widehat{Y}_s(x) = \mathsf{Enc}_s(x)$.

2. ($\mathsf{From}(T_{ij}) = \mathcal{Z}_t$.) In this case, note that $\left|Z_t(x) - (1 - \widehat{Z}_t(x))\right| \geq 1/2$ by the definition of $\widehat{Z}_t(x)$. Hence

$$\begin{aligned}(T_{ij}(x) - \widehat{T}_{ij}(x))^2 &= \left(\widehat{Z}_i(x) - Z_i(x)\right)^2 \\ &\leq \left(\widehat{Z}_t(x) - Z_t(x)\right)^2 \cdot 4 \cdot \left(Z_t(x) - (1 - \widehat{Z}_t(x))\right)^2 \\ &= 4 \cdot (Z_t(x) - 0)^2 \cdot (Z_t(x) - 1)^2 \qquad (\widehat{Z}_t(x) \in \{0,1\}) \\ &= 4 \cdot P_{ij}(x).\end{aligned}$$

This completes the proof of Claim 3.

It follows that

$$\begin{aligned}\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2 &\leq \mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \left(\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} 4 \cdot P_{ij}(x)\right)^{1/2} && \text{(Claim 3)} \\ &\leq \left(\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell} 4 \cdot P_{ij}(x)\right)^{1/2} && \text{(Jensen's inequality)} \\ &\leq \left(\mathop{\mathbb{E}}_{i,j\in[m]\times[2]} 4 \cdot \left(\widetilde{\mathbb{E}}_x(P_{ij}) + \delta\right)\right)^{1/2} && \text{(Lemma 6.7)} \\ &\leq \sqrt{12\delta}. && \text{(by (19))}\end{aligned}$$

(2.b) Note that by the Cauchy-Schwarz inequality, we have

$$\begin{aligned}\left\|T_{i1} \cdot T_{i2} - \widehat{T}_{i1} \cdot \widehat{T}_{i2}\right\|_1 &\leq \left\|(T_{i1} - \widehat{T}_{i1}) \cdot T_{i2}\right\|_1 + \left\|\widehat{T}_{i1} \cdot (\widehat{T}_{i2} - T_{i2})\right\|_1 \\ &\leq \left\|(T_{i1} - \widehat{T}_{i1})\right\|_2 \cdot \left\|T_{i2}\right\|_2 + \left\|\widehat{T}_{i1}\right\|_2 \cdot \left\|(\widehat{T}_{i2} - T_{i2})\right\|_2 \\ &\leq 2 \cdot \sum_{j=1,2} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2.\end{aligned}$$

The last inequality above follows from

$$\begin{aligned}\|T_{ij}\|_2 &= \left(\mathop{\mathbb{E}}_x Q_{ij}(x)\right)^{1/2} && \text{(Definition of } Q_{ij}(x)) \\ &\leq \left(1 + \widetilde{\mathbb{E}}_x(Q_{ij})\right)^{1/2} \leq 2. && \text{(by (20))}\end{aligned}$$

Recall that $F_i(x) = \widetilde{\mathsf{Cons}_i}(T_{i1}(x), T_{i2}(x))$, and $\widehat{F}_i(x) = \widetilde{\mathsf{Cons}_i}(\widehat{T}_{i1}(x), \widehat{T}_{i2}(x))$. Since Cons_i is an OR on two input bits or their negations, one can write

$$\widetilde{\mathsf{Cons}_i}(y_1, y_2) = \sum_{S\subseteq[2]} \alpha_S \cdot \prod_{i\in S} y_i,$$

such that all $|\alpha_S| \leq 1$. Therefore, it follows that

$$\|F_i - \widehat{F}_i\|_1 \leq \sum_{j=1}^{2} \left\|T_{ij} - \widehat{T}_{ij}\right\|_1 + \left\|T_{i1} \cdot T_{i2} - \widehat{T}_{i1} \cdot \widehat{T}_{i2}\right\|_1 \leq 6 \cdot \mathop{\mathbb{E}}_{j\in[2]} \left\|T_{ij} - \widehat{T}_{ij}\right\|_2. \qquad \square$$

6.5 Almost-Everywhere PRG for ACC^0 with an NP Oracle

As an immediate consequence of the new average-case lower bound, we can construct almost-everywhere E^{NP}-computable PRGs for $\mathscr{C}$ circuits given that certain non-trivial CAPP algorithms exist.

Theorem 6.8. *Let $\mathscr{C}$ be a typical circuit class. Suppose that for some $\varepsilon > 0$, CAPP of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{Junta}_{n^\varepsilon}$ circuits can be deterministically solved in $O(2^{n-n^\varepsilon})$ time. Then there is a constant $\delta > 0$ and an E^{NP} algorithm $\mathcal{G}$ such that, on input x of length m, outputs a 2^{m^δ}-bit string such that for all but finitely many m, $\mathcal{G}_m \colon \{0,1\}^m \to \{0,1\}^{2^{m^\delta}}$ is a PRG fooling $\mathscr{C}$ circuits of size 2^{m^δ}.*

Proof Sketch. By Theorem 1.2 and the hypothesis, there is a $\beta > 0$ (which is a function of ε) and an E^{NP} language L such that L_n cannot be $(1/2 + 2^{-n^\beta})$-approximated by $\mathscr{C} \circ \mathsf{Junta}_{n^{\Omega(\varepsilon)}}$ circuits of size 2^{n^β}.[19]

We define $\mathcal{G}$ as follows: On an input of length m, let $n = m^{1/3}$. We first compute $L_n \colon \{0,1\}^n \to \{0,1\}$. Then we feed L_n into the Nisan-Wigderson PRG (Lemma 3.5). This results in a PRG of seed length $\leq n^3 \leq m$, which fools $\mathscr{C}$ circuits of size $2^{m^{\Omega(\beta)}}$. Finally we can set $\delta > 0$ so that this size is 2^{m^δ}. □

Observe that $\mathsf{AC}^0_d[m] \circ \mathsf{Junta}_a$ circuits of size s can be easily simulated by $\mathsf{AC}^0_{d+2}[m]$ circuits of size $O(a \cdot 2^a \cdot s)$. Therefore, for $d, m \in \mathsf{N}$ and a constant $\varepsilon = \varepsilon(d,m) > 0$, Theorem 3.12 yields a #SAT algorithm for $2^{n^{\varepsilon/2}}$-size $\mathsf{AC}^0_d[m] \circ \mathsf{Junta}_{n^{\varepsilon/2}}$ circuits that runs in time $O(2^{n-n^\varepsilon})$. Combining this algorithm with Theorem 6.8, we obtain the following.

Reminder of Theorem 1.9. *For all constants d and m, there is $\delta = \delta(d,m) > 0$ and an E^{NP}-computable PRG which, takes an n-bit seed and outputs a 2^{n^δ}-bit string, fooling $\mathsf{AC}^0_d[m]$ circuits of size 2^{n^δ}.*

7 From Strong Average-Case Lower Bounds to an NPRG

In this section, we construct the infinitely-often nondeterministic PRG.

Theorem 7.1. *For a circuit class $\mathscr{C}$, if for some $\varepsilon > 0$, CAPP of 2^{n^ε}-size $(\mathscr{C} \circ \mathsf{Junta}_{n^\varepsilon})$ circuits can be deterministically solved in 2^{n-n^ε} time, then there exists an i.o.-NPRG with seed length n which fools $\mathscr{C}$ circuits of size $2^{n^{\Omega(1)}}$, running in time $2^{O(n)}$.*

Proof. Let $L^{\mathsf{hard}} \in \mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n]$ be a unary language. We define another algorithm $\mathcal{A}_{\mathsf{fast}}$ which tries to compute L^{hard} faster. $\mathcal{A}_{\mathsf{fast}}$ is similar to $\mathcal{A}_{\mathsf{PCPP}}$. It runs the following four steps.

1. $\mathcal{A}_{\mathsf{fast}}$ rejects all non-unary inputs. On input $z = 1^n$, $\mathcal{A}_{\mathsf{fast}}$ applies PCP from Lemma 3.10 first, obtaining an oracle circuit VPCP_{1^n}. For brevity, we just write it as $\mathsf{VPCP}_{(n)}$ from now on. Recall that $\mathsf{VPCP}_{(n)}$ and its oracle take input of length $\ell = \ell(n) = n + O(\log n)$.
2. Then $\mathcal{A}_{\mathsf{fast}}$ guesses a $2^{\ell^{\varepsilon/4}}$-size general circuit C. Feeding C into $\mathsf{VPCP}_{(n)}$ we obtain $\mathsf{VPCP}^C_{(n)}$.
3. $\mathcal{A}_{\mathsf{fast}}$ applies the PCPP from Lemma 3.11 and guesses a list of $\mathsf{Sum} \circ \mathscr{C} \circ \mathsf{Junta}_{\ell^{\varepsilon/4}}$ proof circuits (Y, Z), each of complexity $2^{\ell^{\varepsilon/2}}$. $\mathcal{A}_{\mathsf{fast}}$ runs the validity test (described in last section, see Lemma 6.3), and rejects immediately if (Y, Z) does not pass the validity test.
4. Finally, $\mathcal{A}_{\mathsf{fast}}$ verifies the proof circuits (Y, Z) for $\mathsf{VPCP}_{(n)}$ using a fast CAPP algorithm, via Lemma 6.2.

Applying the assumed CAPP algorithm, it follows $\mathcal{A}_{\mathsf{fast}} \in \mathsf{NTIME}[2^n/n]$. Therefore, for infinitely many n, we have $1^n \in L^{\mathsf{hard}} \setminus L(\mathcal{A}_{\mathsf{fast}})$. Let $\mathcal{S}$ be the set of all such integers n: $\mathcal{S} = \{n : 1^n \in L^{\mathsf{hard}} \setminus L(\mathcal{A}_{\mathsf{fast}})\}$. Depending on whether there is a succinct circuit for $\mathsf{VPCP}_{(n)}$ as a correct oracle (meaning that $\mathsf{VPCP}^C_{(n)}(r) = 1$ for all r), we will use different constructions.

[19] This is implicit in the proof of Theorem 1.2, as the input length increases only polynomially through the hardness amplification.

Case 1. There are infinitely many $n \in \mathcal{S}$ such that, there is no circuit C of size $2^{\ell^{\varepsilon/4}}$ such that $\mathsf{VPCP}^C_{(n)}$ is a tautology. In the following we only consider these n, as there are infinitely many of them. Then we construct our NPRG as follows:

1. For a given seed-length parameter m, let $n = \frac{m}{K}$ for a large constant K. We consider the PCP oracle circuit $\mathsf{VPCP}_{(n)}$ and guess the truth table $f\colon \{0,1\}^{\ell(n)} \to \{0,1\}$ such that $\mathsf{VPCP}^f_{(n)}$ is a tautology (the verification can be done in $2^{O(\ell(n))}$ time). Note that such f exists as $1^n \in L^{\mathsf{hard}}$.

2. Then, f is hard against circuits of size $2^{\ell^{\varepsilon/4}}$, by our assumption. We plug f into Umans' PRG construction from Lemma 3.6, and get the desired PRG $G_{g\ell}\colon \{0,1\}^{g\ell} \to \{0,1\}^{2^{\ell^{\Omega(1)}}}$. Recall $\ell = n + O(\log n)$, we can then pad input length from $g\ell$ to m.

Case 2. For all but finitely many $n \in \mathcal{S}$, there exists a $2^{\ell^{\varepsilon/4}}$-size circuit C such that $\mathsf{VPCP}^C_{(n)}$ is a tautology. In the following we only consider these n. We construct our NPRG as follows.

1. For a given seed-length parameter m, let $n = m^{1/3}$. We apply the PCP reduction to get $\mathsf{VPCP}_{(n)}$. Then we guess an oracle C of size $2^{\ell(n)^{\varepsilon/4}}$ such that $\mathsf{VPCP}^C_{(n)}$ is a tautology. Note that such a C exists for the integers n we are considering, by assumption.

2. Next, we guess a list of proof functions $(\widehat{Y} = \mathsf{Enc}(x), \widehat{Z})\colon \{0,1\}^{\ell} \to \{0,1\}$ such that
$$\mathop{\mathbb{E}}_{i}\mathop{\mathbb{E}}_{x}\widehat{F}_i(x) \geq c_{\mathsf{pcpp}}.$$
Let $r = \ell^{\varepsilon/3}$ be the total amount of randomness in the PCPP. Consider the function
$$\begin{aligned} f\colon \{0,1\}^{r+1+\ell} &\to \{0,1\} \\ (i,j,u) &\mapsto \widehat{T}_{ij}(u). \end{aligned}$$
By analogy with Lemma 6.5, f is δ-far from $[0,1]\mathsf{Sum} \circ \mathscr{C} \circ \mathsf{Junta}_{\ell^{\varepsilon/4}}[2^{\ell^{\varepsilon/4}}]$ (since $1^n \notin L(\mathcal{A}_{\mathsf{fast}})$). Then we use Lemma 3.8 to apply hardness amplification with parameter $k = \ell^{\varepsilon/16}$. This gives us a function $f\colon \{0,1\}^{O(\ell^{1+\varepsilon/16})} \to \{0,1\}$ which cannot be $(\frac{1}{2} + 2^{-\ell^{\Omega(1)}})$-approximated by $\mathscr{C} \circ \mathsf{Junta}_{\ell^{\varepsilon/4}}$ circuits of size $2^{\ell^{\Omega(1)}}$.

3. Finally, we use the Nisan-Wigderson PRG from Lemma 3.5 with function f. The resulting seed length is bounded by $\ell^3 \leq m$, and the obtained PRG can $2^{-\ell^{\Omega(1)}}$-fool $\mathscr{C}$ circuits of size $2^{\ell^{\Omega(1)}}$.

Summary. To summarize, in both cases, we have an i.o.-NPRG construction which takes m-bit seeds and outputs $2^{m^{\Omega(1)}}$-bit strings in $2^{O(m)}$ time, fooling $2^{m^{\Omega(1)}}$-size $\mathscr{C}$ circuits. □

As shown in the last section, we have a CAPP algorithm for 2^{n^ε}-size $\mathsf{AC}^0_d[m] \circ \mathsf{Junta}_{n^\varepsilon}$ circuits running in 2^{n-n^ε} time. The following is immediate.

Reminder of Theorem 1.10. *For all constants d and m, there is $\delta = \delta(d,m) > 0$ and an i.o.-*NPRG *which takes n-bit seeds, runs in $2^{O(n)}$ time, and outputs 2^{n^δ}-bit strings fooling $\mathsf{AC}^0_d[m]$ circuits of size 2^{n^δ}.*

8 More Applications

In this section, we apply our new refuter to strengthen several previous infinitely-often separations into almost-everywhere separations.

8.1 Almost-Everywhere Rigid Matrix Constructions

A recent result by Alman and Chen [AC19] gave an explicit infinitely-often P^{NP} construction of rigid matrices via Williams' algorithmic approach. Their proof has been simplified and improved by [BHPT20]. Using our refuter, we can improve their construction to an almost-everywhere one.

8.1.1 Preliminaries

We recall the definition of matrix rigidity.

Reminder of Definition 1.4. *Let $\mathbb{F}$ be a finite field. For any $r, n \in \mathbb{N}$ and matrix $M \in \mathbb{F}^{n\times n}$, the r-rigidity of M, denoted $\mathcal{R}_M(r)$, is the minimum Hamming distance between M and any matrix of rank at most r.*

We need a series of technical ingredients from previous results. We start with a fast algorithm for solving a problem called $\#\mathsf{OV}_{n,d}$-$\mathbb{F}_q$.

Definition 8.1. For a prime power $q = p^r$, an instance of $\#\mathsf{OV}_{n,d}$-$\mathbb{F}_q$ consists of two collections of vectors from $\mathbb{F}_q^d$: $A = \{a_1, \ldots, a_n\} \subseteq \mathbb{F}_q^d$ and $B = \{b_1, \ldots, b_n\} \subseteq \mathbb{F}_q^d$. The goal is to compute the number of pairs (a_i, b_j) such that $\langle a_i, b_j\rangle = 0$, where the inner product is taken over $\mathbb{F}_q$.

Note that the number returned by $\#\mathsf{OV}_{n,d}$-$\mathbb{F}_q$ is an integer in $\{0, 1, \ldots, n^2\}$ (we are not computing this quantity over $\mathbb{F}_q$!).

Lemma 8.2 ([AC19, CW16])**.** *For all fixed prime powers $q = p^r$ and $d = n^{o(1)}$, there is an $n^{2-\Omega(1/\log(d/\log n))}$ time deterministic algorithm for computing $\#\mathsf{OV}_{n,d}$-$\mathbb{F}_q$.*

In a recent improvement on rigid matrix constructions, [BHPT20] gave a PCP construction with an "almost-rectangular" property, which will be crucial for our proof.[20]

Lemma 8.3 ([BHPT20, Theorem 8.2 and Remark 8.3])**.** *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where $|x| = n$. For any odd constant integer $m \in \mathbb{N}$ such that $T(n)^{1/m} \geq n$, given $x \in \{0,1\}^n$ one can output in time $\mathrm{poly}(n, T^{1/m})$ a PCP system $\mathsf{VrecPCP}_x$ with proof length ${N_1}^2$, randomness $|r| = \log T(n) + O(m \log\log n)$, completeness 1 and soundness s such that $0 < s < 1$ and the following hold.*

- ***Shortness.*** *${N_1}^2 \leq 2^r \leq T \cdot \mathrm{polylog}(T)$.*
- ***Rectangular.*** *The randomness r can be split into three parts $r = (r_{\mathsf{row}}, r_{\mathsf{col}}, r_{\mathsf{shared}})$ such that $|r_{\mathsf{row}}| = |r_{\mathsf{col}}| \geq \frac{m-6}{2m} \log T(n)$. For a given proof π and fixed r_{shared}, we view $\mathsf{VrecPCP}_x^{\pi}(\cdot, \cdot, r_{\mathsf{shared}})$ as a $[2^{|r_{\mathsf{row}}|}] \times [2^{|r_{\mathsf{col}}|}]$ matrix, where $\mathsf{VrecPCP}_x^{\pi}(r_i, r_j, r_{\mathsf{shared}})$ is the output of verifier given $r = (r_i, r_j, r_{\mathsf{shared}})$ as its randomness.*

  If a proof H is given by its low-rank decomposition $H = A \cdot B$ where A, B are $N_1 \times R$ and $R \times N_1$ matrices respectively, then $\mathsf{VrecPCP}_x^{H}(\cdot, \cdot, r_{\mathsf{shared}})$ is an $\mathbb{R}$-linear combination of constant many matrices:

$$\mathsf{VrecPCP}_x^{H}(\cdot, \cdot, r_{\mathsf{shared}}) = \sum_{i=1}^{\ell} c_i \cdot M_i,$$

  where each of M_i has $\mathbb{F}_2$-rank $O(R)$. Moreover, each M_i's low-rank decomposition and each c_i can be computed in $O(2^{|r_{\mathsf{row}}|} \cdot R)$ time.

[20] Here we omit the exact technical definition of almost-rectangular PCPs, and only state its desired implications for simplicity. We will also just say rectangular PCPs instead of almost-rectangular ones for convenience.

- ***Completeness.*** *If there is a y such that $M(x,y)$ accepts, then there is an $N_1 \times N_1$ matrix H such that:*

$$\Pr_r[\mathsf{VrecPCP}_x^H(r) = 1] \geq 1.$$

- ***Soundness.*** *If no y causes $M(x,y)$ to accept, then for every matrix H, we have:*

$$\Pr_r[\mathsf{VrecPCP}_x^H(r) = 1] \leq s.$$

- ***Smoothness.*** *For every pair $(s,t) \in [N_1] \times [N_1]$, the quantity:*

$$\left|\{r : \mathsf{VrecPCP}_x^H(r) \text{ depends on } H(s,t)\}\right|$$

is the same, i.e., each $H(s,t)$ "contributes" to the same number of clauses.

The following simple fact shows we can "pad" a rigid matrix into a larger rigid matrix with the same rigidity measure.

Lemma 8.4 (Lemma 2.7 in [AC19])**.** *Let $\mathbf{1}_M$ be the all-ones $M \times M$ matrix. For any field $\mathbb{F}$ and any matrix $A \in \mathbb{F}^{N \times N}$, we have:*

$$\mathcal{R}_{A \otimes \mathbf{1}_M}(r) = \mathcal{R}_A(r) \cdot M^2.$$

8.1.2 Construction of Rigid Matrices

Now we are ready to prove Theorem 1.5. In previous work, the high-level idea was to use the "non-existence" of rigid matrices in P^{NP} to contradict the NTIME Hierarchy. Here, we will use the P^{NP} refuter to construct rigid matrices directly. A subtlety is that we need to apply a refuter (with a SAT oracle) based on the "robustly often" NTIME hierarchy (Theorem 4.8) rather than the refuter based on the almost-everywhere sublinear witness NTIME hierarchy (Theorem 4.6). The main reason is that, when we guess a low-rank matrix as a proof for a language $L \in \mathsf{NTIME}[T(n)]$, we have to guess at least $\sqrt{T(n)}$ bits (specifying even a rank-one matrix of size $\sqrt{T(n)} \times \sqrt{T(n)}$ requires at least $\Omega(\sqrt{T(n)})$ bits). Hence, making the witness length sublinear would require us to set $T(n) \leq n^2$, but the PCP reduction from Lemma 8.3 may already require more than quadratic time.

Reminder of Theorem 1.5. *There is a $\delta > 0$ such that, for every finite field $\mathbb{F}$ and $\varepsilon > 0$, there is a P^{NP} algorithm which on input 1^n outputs an $n \times n$ matrix H such that $\mathcal{R}_H(2^{\log^{1-\varepsilon} n}) \geq \delta n^2$ over $\mathbb{F}$, for all large enough n.*

Proof Sketch. For simplicity, we only prove the theorem for the case of $\mathbb{F}_2$, but a similar construction works for any field of constant order.

Let $c = 1$ and s be the completeness and soundness parameters from Lemma 8.3. Fix $\delta = \frac{c-s}{C_1}$ for a sufficiently large constant C_1. Let m be a large enough odd constant, and let $T(n) = n^C$ for a large constant $C > m$. Now, we consider the algorithm $\mathcal{A}_{\mathsf{RO}}^T$ from Theorem 4.8 and the following algorithm $\mathcal{A}_{\mathsf{rectPCP}}$ which tries to speed up $\mathcal{A}_{\mathsf{RO}}^T$:

1. On an input z of length n, $\mathcal{A}_{\mathsf{rectPCP}}$ applies the PCP reduction from Lemma 8.3 with the parameter m. Then $\mathcal{A}_{\mathsf{rectPCP}}$ guesses a pair of matrices $A \in \mathbb{F}_2^{N_1 \times R}, B \in \mathbb{F}_2^{R \times N_1}$ with parameter $R = 2^{\log^{1-\frac{\varepsilon}{2}} N_1} = {N_1}^{o(1)}$.

2. Let $H = A \cdot B$. For any fixed r_{shared}, consider $\mathsf{VrecPCP}_z^H(\cdot,\cdot,r_{\mathsf{shared}})$. By the rectangular property, we can write

$$\mathsf{VrecPCP}_z^H(\cdot,\cdot,r_{\mathsf{shared}}) = \sum_{i=1}^{\ell} c_i \cdot M_i,$$

where each M_i can be written as a product of two matrices of rank $O(R)$ over $\mathbb{F}_2$. For each M_i, we first find the corresponding decomposition $M_i = Q_{i,1} \cdot Q_{i,2}$, then apply the fast $\#\mathsf{OV}_{n,d}$-$\mathbb{F}_2$ algorithm to count the number of 1s in M_i[21]. Finally, by adding all the ℓ terms together, we can count the number of 1s in $\mathsf{VrecPCP}_z^H(\cdot,\cdot,r_{\mathsf{shared}})$ in time

$$2^{|r_{\mathsf{row}}|+|r_{\mathsf{col}}|-\Omega\left(\frac{|r_{\mathsf{row}}|}{\log(R/|r_{\mathsf{row}}|)}\right)}.$$

3. By enumerating all possible r_{shared}, we evaluate

$$p = \Pr_r[\mathsf{VrecPCP}_z^H(r) = 1].$$

Finally, we *accept* if and only if $p > \frac{c+s}{2}$. This completes the construction of $\mathcal{A}_{\mathsf{rectPCP}}$. When m is large enough, the running time of this step is bounded by

$$2^{|r_{\mathsf{shared}}|+|r_{\mathsf{row}}|+|r_{\mathsf{col}}|-\Omega\left(\frac{|r_{\mathsf{row}}|}{\log(R/|r_{\mathsf{row}}|)}\right)} = T(n) \cdot 2^{O(m \log\log T(n)) - \log^{\Omega(\varepsilon)} T(n)} \le o(T(n)).$$

Note that the time to prepare the PCP system $\mathsf{VrecPCP}_z$ is bounded by $\mathrm{poly}(n, T(n)^{1/m}) \le o(T(n))$ for a sufficiently large m. It follows that $\mathcal{A}_{\mathsf{rectPCP}}$ runs in nondeterministic $o(T(n))$ time. Hence, for all large enough n, by Theorem 4.8, we can find in $n^{C^{O(1)}}$ time (with a SAT oracle) an input z of length $|z| \in [n, n^{C+1}]$ such that $\mathcal{A}_{\mathsf{RO}}^T(z) = 1$ while $\mathcal{A}_{\mathsf{rectPCP}}(z) = 0$.

Slightly abusing notation, we still let $N_1 = N_1(|z|)$ so that N_1^2 is the proof length of the $\mathsf{VrecPCP}_z$. Then it follows from Lemma 8.3 that there is an $H \in \mathbb{F}^{N_1 \times N_1}$ such that

$$\Pr_r[\mathsf{VrecPCP}_z^H(r) = 1] \ge 1.$$

Using a SAT oracle, the lexicographically first such H can be constructed in $\mathrm{poly}(N_1)$ time, given z. We claim that H is δ-far from any matrix of rank $R = 2^{\log^{1-\varepsilon/2} N_1}$. In fact, let $H' \in \mathbb{F}^{N_1 \times N_1}$ be any matrix of rank at most R with low rank decomposition $H' = A \cdot B$. Suppose the Hamming distance between H and H' is at most $\delta \cdot {N_1}^2$. Then

$$\begin{aligned}
\Pr_r[\mathsf{VrecPCP}_z^{H'}(r) = 1] &\ge \Pr_r[\mathsf{VrecPCP}_z^{H}(r) = 1] - O(\delta) \quad \text{(The Smoothness Condition in Lemma 8.3)} \\
&\ge 1 - O(\delta) \\
&\ge \frac{1+s}{2}.
\end{aligned}$$

This shows that (A, B) is a correct witness for $\mathcal{A}_{\mathsf{rectPCP}}(z)$, and $\mathcal{A}_{\mathsf{rectPCP}}(z) = 1$, which is a contradiction.

Note that $N_1 \in [n, n^{(C+1)^2}]$, since $|z| \in [n, n^{C+1}]$. Applying Lemma 8.4, we can pad the size of H to be $N_{\max} = n^{(C+1)^2}$, letting the new matrix be $H_{\max}$. Then

$$\mathcal{R}_{H_{\max}}(2^{\log^{1-\varepsilon/2} n}) \ge \delta \cdot {N_{\max}}^2.$$

Finally, observe that $2^{\log^{1-\varepsilon/2} n} \ge 2^{C^{-3} \cdot \log^{1-\varepsilon/2} N_{\max}} \ge 2^{\log^{1-\varepsilon} N_{\max}}$.

Therefore, our final P^{NP} algorithm can be described as follows: on input 1^m, set $n = m^{1/(C+1)^2}$ so that $N_{\max} = m$, and output the matrix $H_{\max}$. The whole construction takes at most $n^{C^3} \le \mathrm{poly}(m)$ time with a SAT oracle, which completes the proof.

□

[21] This reduces to the $\#\mathsf{OV}_{n,d}$-$\mathbb{F}_2$ instance between row vectors of $Q_{i,1}$ and column vectors of $Q_{i,2}$.

8.2 Almost-Everywhere Probabilistic Degree Lower Bounds

Next we improve Viola's $\Omega(n/\log^2 n)$ probabilistic degree lower bound [Vio20] to an almost-everywhere one. We first recall the definition of probabilistic degree.

Reminder of Definition 1.6. *The ε-error degree of a function $f\colon \{0,1\}^n \to \{0,1\}$ is the minimum d such that there is a distribution $\mathcal{D}$ on polynomials over $\mathbb{F}_2$ of degree d such that $\Pr_{P\sim\mathcal{D}}[P(x) \neq f(x)] \leq \varepsilon$.*

We need the following fast evaluation algorithm for low degree polynomial.

Lemma 8.5 ([Wil18b, Theorem 26])**.** *Let $p\colon \{0,1\}^n \to \{0,1\}$ be an $\mathbb{F}_2$-polynomial of degree d, we can evaluate $\sum_{x\in\{0,1\}^n} p(x)$ in $2^{n-\Omega(n/d)+O(\log n)}$ time.*

The main theorem in [Vio20] can be adapted to the almost-everywhere setting.

Reminder of Theorem 1.7. *There is a language $L\colon \{0,1\}^\star \to \{0,1\}$ in E^{NP} such that L_n has $1/3$-error probabilistic degree at least $\Omega(n/\log^2 n)$, for every sufficiently large n.*

Proof. Let $C > 0$ be a large enough constant and let $T(n) = n^C$. Recall the algorithm $\mathcal{A}^T_{\mathsf{FS}}$ from Definition 4.2, and define the following algorithm $\mathcal{A}_{\mathsf{polyPCP}}$ which tries to speed up $\mathcal{A}^T_{\mathsf{FS}}$:

1. On an input z of length n, it applies the PCP reduction from Lemma 3.10 to $\mathcal{A}^T_{\mathsf{FS}}(z)$. Let $\ell = C\log n + O(\log\log n)$ be the length of the random string in the PCP, also let VPCP_z be the oracle circuit.

2. Next, $\mathcal{A}_{\mathsf{polyPCP}}$ guesses an $\mathbb{F}_2$-polynomial $P\colon \{0,1\}^\ell \to \{0,1\}$ of degree at most $\frac{\ell}{2C\log\ell}$, which can be described by $\ell^{\ell/(2C\log\ell)}\cdot\ell \leq 2^{\ell/(2C)}\cdot\ell \leq n/10$ bits. (P will be treated as the oracle in the PCP.)

3. Consider VPCP^P_z. We know that each query $q_i\colon \{0,1\}^\ell \to \{0,1\}^\ell$ to P is a projection, which naturally induces a polynomial $Q_i(x) = P(q_i(x))$, where $\deg(Q_i) = \deg(P)$. The output of VPCP_x is a 3-CNF F over $\{Q_i\}$. Suppose F has $m = \mathrm{poly}(\ell) \leq \ell^{O(\sqrt{C})}$ clauses. We write the j-th clause of F as $(L^{(P)}_{j,1}, L^{(P)}_{j,2}, L^{(P)}_{j,3})$, where each $L^{(P)}_{j,b}$ is an element of $\{Q_i\}$ or its negation.

   Note that for each $j\in[m]$, $L^{(P)}_{j,1}(x)\vee L^{(P)}_{j,2}(x)\vee L^{(P)}_{j,3}(x)$ can be written as an $\mathbb{F}_2$-polynomial of degree $O\left(\frac{\ell}{2C\log\ell}\right)$. By Lemma 8.5, we can evaluate

$$\mathop{\mathbb{E}}_{j\in[m]}\ \mathop{\mathbb{E}}_{x\in\{0,1\}^\ell}(L^{(P)}_{j,1}(x)\vee L^{(P)}_{j,2}(x)\vee L^{(P)}_{j,3}(x)) \tag{23}$$

   in $m\cdot 2^{\ell-\Omega(C\log\ell)+O(\log\ell)} = 2^{\ell-\Omega(C\log\ell)}$ time.

4. Finally, we accept if and only if the expectation (23) is larger than $1-\ell^{-5C}$. This completes the construction of $\mathcal{A}_{\mathsf{polyPCP}}$.

Note that $\mathcal{A}_{\mathsf{polyPCP}}$ can be implemented to run in $2^{\ell-\Omega(C\log\ell)} \leq o(T(n))$ time. Since $\mathcal{A}_{\mathsf{polyPCP}}$ runs in $o(T(n))$ time while guessing at most $n/10$ bits of witness, we can apply the refuter in Theorem 4.6. We show the following two lemmas:

Lemma 8.6. *For every sufficiently large C and $T(n) = n^C$, it holds for all but finitely many $z\in\{0,1\}^*$ that $\mathcal{A}(z) \leq \mathcal{A}^T_{\mathsf{FS}}(z)$.*

Proof. The lemma statement is equivalent to saying that $\mathcal{A}^T_{\mathsf{FS}}(z) = 0$ implies $\mathcal{A}(z) = 0$. Suppose $\mathcal{A}^T_{\mathsf{FS}}(z) = 0$. Then for any oracle $\mathcal{O}$, at most an $n^{-10} \leq \ell^{-10C}$ fraction of $\mathsf{VPCP}^{\mathcal{O}}_z$'s assignments are satisfying. Therefore, for any guessed polynomial P, we have:

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell}\left[\bigwedge_{j\in[m]}\left(L^{(P)}_{j,1}(x)\vee L^{(P)}_{j,2}(x)\vee L^{(P)}_{j,3}(x)\right)\right]\leq \ell^{-10C}.$$

It follows that

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell}\mathop{\mathbb{E}}_{j\in[m]}(L^{(P)}_{j,1}(x)\vee L^{(P)}_{j,2}(x)\vee L^{(P)}_{j,3}(x))\leq \ell^{-10C}+\frac{m-1}{m}<1-\ell^{-5C}.$$

Therefore, $\mathcal{A}(z)$ rejects every possible witness. This completes the proof. □

Lemma 8.7. *For every sufficiently large C and $T(n) = n^C$, for all large enough n and input $z \in \{0,1\}^n$, if $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$ and $\mathcal{A}_{\mathsf{polyPCP}}(z) = 0$, then the correct oracle for VPCP_z has $1/3$-error degree $\Omega(\ell/\log^2\ell)$.*

Proof. Suppose on the contrary that some the correct oracle $\mathcal{O}$ for $V_{\mathsf{PCP}}(x)$ has $1/3$-error degree $d \leq o(\ell/\log^2\ell)$. Let $\mathcal{F}$ be the corresponding probabilistic polynomial. Next, we reduce the error of the probabilistic $\mathbb{F}_2$-polynomial by composing a degree $O(\log\ell)$ $\mathbb{F}_2$-poly for majority with $O(\log\ell)$ many independently samples from $\mathcal{F}$ naturally. We obtain a $(1-\ell^{-10C})$-error probabilistic polynomial $\mathcal{F}'$ of degree $O(d\log\ell)$. Now, if we take one random polynomial P from the distribution $\mathcal{F}'$, it follows that

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell}\mathop{\mathbb{E}}_{j\in[m]}\mathop{\mathbb{E}}_{P\sim\mathcal{F}'}(L^{(P)}_{j,1}(x)\vee L^{(P)}_{j,2}(x)\vee L^{(P)}_{j,3}(x))\geq 1-\ell^{-10C}.$$

By the averaging principle, we can fix a polynomial P such that:

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^\ell}\mathop{\mathbb{E}}_{j\in[m]}(L^{(P)}_{j,1}(x)\vee L^{(P)}_{j,2}(x)\vee L^{(P)}_{j,3}(x))\geq 1-\ell^{-10C}.$$

Note that the degree of P is $O(d\log\ell)\leq o(\ell/\log\ell)$. Therefore, P is an acceptable witness for $\mathcal{A}_{\mathsf{polyPCP}}(x)$ and $\mathcal{A}_{\mathsf{polyPCP}}(x) = 1$. This is a contradiction.

□

Given Lemma 8.6 and 8.7, the rest of the proof is analogous to the proof of Theorem 1.1.

□

8.3 An Improved Degree-Error Trade-off

Using our improved XOR Lemma, we are also able to prove a better degree-error trade-off for correlation bounds against $\mathbb{F}_2$-polynomials.

Reminder of Theorem 1.8. *For all $\delta > 0$, there is a language $L\colon \{0,1\}^\star \to \{0,1\}$ in E^{NP} such that L_n cannot be $(1/2 + 2^{-n^{\Omega(1)}})$-approximated by $n^{1/2-\delta}$ degree $\mathbb{F}_2$-polynomials, for every sufficiently large n.*

Proof Sketch. Note that low-degree $\mathbb{F}_2$-polynomials are just a special case of $\mathsf{AC}^0[\oplus]$, which admits a fast #SAT algorithm (Lemma 8.5). In the following we will mimic the proof of Theorem 1.2.

Let $C > 0$ to be a large enough constant and $T(n) = n^C$. Consider $\mathcal{A}^T_{\mathsf{FS}}$ and the following algorithm $\mathcal{A}$ which tries to speed up $\mathcal{A}^T_{\mathsf{FS}}$.

1. On an input z of length n, $\mathcal{A}$ applies the PCP of Lemma 3.10 first, obtaining the oracle circuit VPCP_z. Recall that VPCP_z and its oracle take inputs of length $\ell = \ell(n) = C\log n + O(\log\log n)$. Let $d = d(\ell) = \ell^{1/2-\delta/3}$ be the degree parameter.

2. Next, $\mathcal{A}$ guesses a general circuit C of size at most $2^{\sqrt{\ell}}$. Note that we need a total of $2^{\widetilde{O}(\sqrt{\ell})} \le o(n)$ bits to specify C.

3. Feeding C into VPCP_z, we obtain VPCP_z^C, which can be written as a general circuit of size at most $\mathrm{poly}(|C|)$.

4. Applying the PCPP from Lemma 3.11, we obtain a 2-SAT instance with $m_1 \le \mathrm{poly}(|C|) \le 2^{O(\sqrt{\ell})}$ clauses, over the set of variables $\mathcal{Y} \cup \mathcal{Z}$. We guess a list of Sum of $m_2 = 2^{\ell^\delta}$ polynomials of degree d, as proof functions. Note that we need at most

$$\mathrm{poly}(|C|) \cdot m_2 \cdot 2^{O(d\log\ell)} \le 2^{\widetilde{O}(\sqrt{\ell})} \le o(n)$$

   bits to specify these polynomials.

5. Using the fast evaluation algorithm of Lemma 8.5, we run a similar validity test and verification as in the proof of Theorem 1.2. The total running time is bounded by

$$\mathrm{poly}(|C|) \cdot \mathrm{poly}(m_2) \cdot 2^{\ell-\Omega(\ell/d)+O(\log\ell)} \le 2^{\ell-\Omega(\ell^{1/2+\delta/3})} \le o(T(n)).$$

   This completes the construction of algorithm $\mathcal{A}$.

By construction, $\mathcal{A}$ is an $\mathsf{NTIMEGUESS}[o(T(n)), n/10]$ algorithm. The analysis of $\mathcal{A}$ is similar to that for Theorem 1.2: We first prove $\mathcal{A}(z) \le \mathcal{A}^T_{\mathsf{FS}}(z)$ for all but finitely many z. Then, for all large enough n, we can use the refuter $\mathcal{R}^T$ from Theorem 4.6 to find an input $z \in \{0,1\}^n$ such that $\mathcal{A}(z) = 0$ and $\mathcal{A}^T_{\mathsf{FS}}(z) = 1$. Using the SAT oracle, we can construct the first correct oracle $\mathcal{O}$ for VPCP_z. Depending on whether there is a succinct circuit computing $\mathcal{O}$ or not, we consider following two cases:

1. **The oracle $\mathcal{O}$ cannot be computed by circuits of size at most $2^{\sqrt{\ell}}$.** Then, by Lemma 3.9, from the oracle $\mathcal{O}$ we can construct a function $f\colon \{0,1\}^{O(\ell)} \to \{0,1\}$, which cannot be $(1/2 + 2^{-\ell^{\Omega(1)}})$-approximated by circuits of size $2^{\Omega(\sqrt{\ell})}$. This in turn implies that f cannot be $(1/2 + 2^{-\ell^{\Omega(1)}})$-approximated by polynomials of degree $d(\ell) = \ell^{1/2-\delta/3}$.

2. **The oracle $\mathcal{O}$ can be computed by some circuit of size at most $2^{\sqrt{\ell}}$.** Let C be the lexicographically first such circuit. (Note that C can be constructed with access to a SAT oracle.) Feeding C into VPCP_z we obtain VPCP_z^C.

   Now consider the PCPP reduction (step 4) in $\mathcal{A}$, we can first find the lexicographically first correct list of proof functions $\mathcal{Y}_s, \mathcal{Z}_t\colon \{0,1\}^\ell \to \{0,1\}$. Based on these functions, we use similar construction as in the proof of Theorem 1.2. It gives us a function f of input length $\ell + O(\log m_2) \le 2\ell$, which is δ-far from every $[0,1]\mathsf{Sum} \circ (\text{degree-}d\text{-poly})$. Using the hardness amplification in Lemma 3.8 with parameter $k = \frac{1}{2}\ell^{\delta/3}$, we obtain a function of input length $\ell^{1+\delta/3}$ which cannot be $(1/2 + 2^{-\ell^{\Omega(1)}})$-approximated by degree $d(\ell) = \ell^{1/2-\delta/2}$ polynomials.

For both cases, we pad the input length to $r = \ell^{1+\delta/3}$. The final functions cannot be $(1/2 + 2^{-r^{\Omega(1)}})$-approximated by polynomials of degree $d(\ell) = \ell^{1/2-\delta/3} \ge r^{1/2-\delta}$. This completes the construction of the hard function. The rest of the proof is analogous to the proof of Theorem 1.1. □

Acknowledgments. The first author wants to thank Shuichi Hirahara for pointing out that Levin's proof of the XOR Lemma has a close connection with approximate linear sum (which was later pointed out again to the first author by Ronen Shaltiel). The authors are grateful to Josh Alman, Hanlin Ren and Roei Tell for detailed comments on an early version of the draft, and helpful discussions. The authors want to thank Amey Bhangale, Prahladh Harsha, Orr Paradise and Avishay Tal for adding Remark 5.3 and 8.3 to their paper so that we can apply their construction of rectangular PCPs directly.

L. Chen and R. Williams were supported by NSF grants CCF-1909429 and CCF-1741615. L. Chen is also supported by an IBM Fellowship.

References

[Aar06] Scott Aaronson. Oracles are subtle but not malicious. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 340–354, 2006.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. 2009.

[AC19] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1034–1055, 2019.

[ADH97] Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM J. Comput.*, 26(5):1524–1540, 1997.

[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.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[Ats06] Albert Atserias. Distinguishing SAT from polynomial-size circuits, through black-box queries. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 88–95, 2006.

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, volume 2483 of *Lecture Notes in Computer Science*, pages 194–208, 2002.

[BCG+96] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996.

[BFS09] Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against advice. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 195–209, 2009.

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BHPT20] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular PCPs. 2020. To appear in the proceedings of FOCS 2020. Full version at ECCC-TR20-075.

[BTW10] Andrej Bogdanov, Kunal Talwar, and Andrew Wan. Hard instances for satisfiability and quasi-one-way functions. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 290–300, 2010.

[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.

[Cai07] Jin-yi Cai. S_2^P is contained in ZPP^{NP}. *J. Comput. Syst. Sci.*, 73(1):25–35, 2007.

[CCHO05] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005.

[Che19] Lijie Chen. Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1281–1304, 2019.

[CMMW19] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Relations and equivalences between circuit lower bounds and Karp-Lipton theorems. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, pages 30:1–30:21, 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.

[CR20] Lijie Chen and Hanlin Ren. Strong average-case lower bounds from non-trivial derandomization. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1327–1334, 2020.

[CW16] Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016.

[CW19] Lijie Chen and R. Ryan Williams. Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity. In *34th Computational Complexity Conference (CCC 2019)*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:43, Dagstuhl, Germany, 2019.

[FGHP99] Stephen A. Fenner, Frederic Green, Steven Homer, and Randall Pruim. Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy. *Electronic Colloquium on Computational Complexity (ECCC)*, 6(3), 1999.

[FLvMV05] Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM (JACM)*, 52(6):835–865, 2005.

[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.

[FS11] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 569–580, 2011.

[FS16] Lance Fortnow and Rahul Santhanam. New non-uniform lower bounds for uniform classes. In *31st Conference on Computational Complexity (CCC 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[FS17] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. *Inf. Comput.*, 256:149–159, 2017.

[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.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(50), 1995.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. 2008.

[GST07] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Comput. Complex.*, 16(4):412–441, 2007.

[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.

[Hås89] Johan Håstad. Almost optimal lower bounds for small depth circuits. *Advances in Computing Research*, 5:143–170, 1989.

[HRST17] Johan Håstad, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *J. ACM*, 64(5):35:1–35:27, 2017.

[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.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 538–545, 1995.

[Kab01] Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *J. Comput. Syst. Sci.*, 63(2):236–252, 2001.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982.

[Kli01] Adam R. Klivans. On the derandomization of constant depth circuits. In *4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2001 and 5th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2001*, volume 2129 of *Lecture Notes in Computer Science*, pages 249–260, 2001.

[KW98] Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998.

[Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[Lok09] Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009.

[MW20] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM Journal on Computing*, pages 300–322, 2020.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 665–677, 2017.

[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.

[Raz89] Alexander A. Razborov. On rigid matrices (in Russian), 1989. Steklov Mathematical Institute. Paper at http://people.cs.uchicago.edu/~razborov/files/rigid.pdf.

[RSS18] Ninad Rajgopal, Rahul Santhanam, and Srikanth Srinivasan. Deterministically counting satisfying assignments for constant-depth circuits with parity gates, with implications for lower bounds. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 78:1–78:15, 2018.

[SFM78] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978.

[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.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[SV10] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176, 1977.

[Vin05] N. V. Vinodchandran. A note on the circuit complexity of PP. *Theor. Comput. Sci.*, 347(1-2):415–418, 2005.

[Vio19] Emanuele Viola. Matching Smolensky's correlation bound with majority. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:175, 2019.

[Vio20] Emanuele Viola. New lower bounds for probabilistic degree and AC0 with parity gates. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:15, 2020.

[vMP06] Dieter van Melkebeek and Konstantin Pervyshev. A generic time hierarchy for semantic models with one bit of advice. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 129–144, 2006.

[VW20] Nikhil Vyas and R. Ryan Williams. Lower bounds against sparse symmetric functions of ACC circuits: Expanding the reach of #SAT algorithms. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 59:1–59:17, 2020.

[Wil07] Ryan Williams. *Algorithms and resource requirements for fundamental problems*. PhD thesis, Ph. D. Thesis, Carnegie Mellon University, CMU-CS-07-147, 2007.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[Wil14] 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.

[Wil18a] R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory Comput.*, 14(1):1–25, 2018.

[Wil18b] 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.

[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.

[Žák83] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983.

A A Hardness Amplification Lemma

In this section, we prove our non-standard XOR Lemma. We remark that the proof is already implicit in Levin's proof of XOR Lemma [Lev87, GNW95].

Reminder of Lemma 3.8. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a boolean function. Let $\delta < \frac{1}{2}$, For any $k \geq 1$, let $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2}-\delta\right)$. If f cannot be $(1-\delta)$-approximated in ℓ_1 distance by $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuits of complexity $O\left(\frac{n\cdot s}{(\delta\cdot\varepsilon_k)^2}\right)$, then $f^{\oplus k}$ cannot be $(\frac{1}{2}+\varepsilon_k)$-approximated by $\mathscr{C}$ circuits of size s.*

Proof. We prove the contrapositive, i.e., given a $\mathscr{C}$ circuit C of size s approximating $f^{\oplus k}$ on at least a $\left(\frac{1}{2}+\varepsilon_k\right)$-fraction of inputs, we show how to construct a $[0,1]\mathsf{Sum}\circ\mathscr{C}$ circuit Q approximating f with a much better guarantee.

For an input x to $f^{\oplus k}$, we write $x = yz$ such that $|y| = n, |z| = (k-1)n$. Our proof is by induction on k. The case $k = 1$ is clearly trivial. Assuming the hypothesis holds for $k-1$, we now consider the following two cases.

Case 1. Suppose for some $y \in \{0,1\}^n$, we have

$$\left|\Pr_z[f^{\oplus k}(y,z) = C(y,z)] - \frac{1}{2}\right| > \frac{\varepsilon_k}{1-\delta} = (1-\delta)^{k-2}\cdot\left(\frac{1}{2}-\delta\right) = \varepsilon_{k-1}.$$

Then, we can fix one such y, and note that either circuit $C'(z) := C(y,z)$ or $\neg C'(z)$ approximates $f^{\oplus(k-1)}$ well enough so that we can reduce it to the case of $k-1$.

Case 2. Otherwise, we have that for all $y \in \{0,1\}^n$:

$$\left|\Pr_z[f^{\oplus k}(y,z) = C(y,z)] - \frac{1}{2}\right| \leq \frac{\varepsilon_k}{1-\delta}.$$

We define

$$\begin{aligned} T(y) :=& \Pr_z[C(y,z) = f^{\oplus k}(y,z)] \\ =& \Pr_z[C(y,z) = f(y) \oplus f^{\oplus(k-1)}(z)]. \end{aligned}$$

From the definition, it follows directly that

$$\left|T(y) - \frac{1}{2}\right| \leq \frac{\varepsilon_k}{1-\delta}. \tag{24}$$

Also, since C approximates $f^{\oplus k}$ on at least $\frac{1}{2}+\varepsilon_k$ fraction of inputs, we have

$$\mathop{\mathbb{E}}_y[T(y)] \geq 1/2 + \varepsilon_k. \tag{25}$$

Now let $Z_1, Z_2, \ldots, Z_\ell$ be a sequence of i.i.d. random variables, where each Z_i is uniformly random from $\{0,1\}^{n(k-1)}$. We then define

$$\widetilde{T}(y) := \mathop{\mathbb{E}}_{i\leftarrow[\ell]}\left[C(y,Z_i) = f(y) \oplus f^{\oplus(k-1)}(Z_i)\right]. \tag{26}$$

Setting $\ell = O\left(\frac{n}{(\delta\varepsilon_k)^2}\right)$, and applying a Chernoff bound, we have

$$\Pr_{\{Z_i\}}\left[\left|T(y) - \widetilde{T}(y)\right| \geq \frac{\delta\varepsilon_k}{2(1-\delta)}\right] \leq 2^{-n-1}.$$

By a union bound, we can fix an assignment $Z_i = z_i$ for each of Z_i such that

$$\left|T(y) - \widetilde{T}(y)\right| \leq \frac{\delta\varepsilon_k}{2(1-\delta)} \tag{27}$$

holds, for all $y \in \{0,1\}^n$. Then from (25), (27), and (24) it follows directly that

$$\mathop{\mathbb{E}}_{y}[\widetilde{T}(y)] \geq \frac{1}{2} + \varepsilon_k - \frac{\delta\varepsilon_k}{2(1-\delta)}, \tag{28}$$

and for all y,

$$\left|\widetilde{T}(y) - \frac{1}{2}\right| \leq \frac{\varepsilon_k}{1-\delta} + \frac{\delta\varepsilon_k}{2(1-\delta)}.$$

Letting $r := \frac{2\varepsilon_k + \delta\varepsilon_k}{1-\delta}$, we define:

$$\widetilde{P}(y) := \left(\frac{\widetilde{T}(y) - \frac{1}{2}}{r} + \frac{1}{2}\right).$$

Note that $\widetilde{P}(y) \in [0,1]$ since $\left|\widetilde{T}(y) - \frac{1}{2}\right| \leq \frac{r}{2}$. From (28), we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{y}[\widetilde{P}(y)] &= \left(\frac{\mathbb{E}_y[\widetilde{T}(y)] - \frac{1}{2}}{r} + \frac{1}{2}\right) \\
&\geq \left(\frac{\varepsilon_k - \frac{\delta\varepsilon_k}{2(1-\delta)}}{r} + \frac{1}{2}\right) \\
&= \left(\frac{\varepsilon_k - \frac{\delta\varepsilon_k}{2(1-\delta)} + \frac{\varepsilon_k}{1-\delta} + \frac{\delta\varepsilon_k}{2(1-\delta)}}{r}\right) \\
&= \frac{\varepsilon_k\left(1 + \frac{1}{1-\delta}\right)}{\varepsilon_k \cdot \frac{2+\delta}{1-\delta}} = \frac{2-\delta}{2+\delta} = 1 - \frac{2\delta}{2+\delta} \geq 1 - \delta.
\end{aligned} \tag{29}$$

Finally, we use the samples $\{(z_i, f^{\oplus(k-1)}(z_i)\}$, to construct a $\mathsf{Sum} \circ \mathscr{C}$ circuit Q as follows:

$$Q(y) := \left(\frac{\Pr_i[f^{\oplus(k-1)}(z_i) \neq C(y, z_i)] - \frac{1}{2}}{r} + \frac{1}{2}\right).$$

Note that Q can be implemented as a sum of $\ell + 1$ $\mathscr{C}$ circuits (one for the constant function $\mathbf{1}$), as $f^{\oplus(k-1)}(z_i)$ can all be replaced by the corresponding constants. The size of Q is bounded by $O\left(\frac{n \cdot s}{(\delta\varepsilon_k)^2}\right)$. The sum of absolute values of coefficients in Q is bounded by $O(1/r) = O\left(\frac{n}{(\delta\varepsilon_k)^2}\right)$. Therefore, the complexity of Q is $O\left(\frac{n \cdot s}{(\delta\varepsilon_k)^2}\right)$.

Finally, note that

$$C(y, Z_i) = f(y) \oplus f^{\oplus(k-1)}(Z_i) \Leftrightarrow f(y) = C(y, Z_i) \oplus f^{\oplus(k-1)}(Z_i),$$

and therefore, from (26), it follows that

$$Q(y) = \begin{cases} \widetilde{P}(y) & f(y) = 1 \\ 1 - \widetilde{P}(y) & f(y) = 0 \end{cases}.$$

From the above, one can see that for all y, we have $Q(y) - f = \widetilde{P}(y) - 1$. Therefore, from (29) and note that $\widetilde{P}(y) \in [0,1]$, we have

$$\left\| Q - f \right\|_1 \leq \mathop{\mathbb{E}}_{y} \left[1 - \widetilde{P}(y) \right] \leq \delta.$$

Also, since $\widetilde{P}(y) \in [0,1]$ for all y, $Q(y) \in [0,1]$ for all y as well. Putting everything together, Q is the desired $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit and this completes the proof. □

B Refuter for Robustly Often Nondeterministic Time Hierarchy Theorem

In this section, we construct a refuter for the "robustly often" nondeterministic time hierarchy theorem. This refuter is used in the construction of rigid matrices (Theorem 1.5).

Reminder of Theorem 4.8. *For any time-constructible function $T(n)$ such that $n \leq T(n)$ and $T(n+1) \leq O(T(n))$. There is an $\mathsf{NTIME}[T(n)]$ machine $\mathcal{A}^T_{\mathsf{RO}}$ and an algorithm $\mathcal{R}^T_{\mathsf{RO}}$ such that:*

1. ***Input.*** *The input for $\mathcal{R}^T_{\mathsf{RO}}$ is a pair $(M, 1^n)$, with the promise that M is a nondeterministic Turing machine runs in $o(T(n))$ time.*
2. ***Output.*** *For any fixed M, for all large enough n, $\mathcal{R}^T_{\mathsf{RO}}(M, 1^n)$ outputs a string x of length $|x| \in [n, n + T(n)]$ such that $\mathcal{A}^T_{\mathsf{RO}}(x) \neq M(x)$.*
3. ***Complexity.*** *$\mathcal{R}^T_{\mathsf{RO}}$ is a deterministic algorithm, runs in $\mathrm{poly}(T(\mathrm{poly}(T(n))))$ time with adaptive access to a SAT oracle.*

Proof Sketch. Informally, the intuition behind the proof of Theorem 1.12 is as follows: We design the algorithm $\mathcal{A}^T_{\mathsf{FS}}$ such that, if an algorithm M accepts a particular string $x = \langle M, n, 0^{n/10} \rangle$, then by the design of $\mathcal{A}^T_{\mathsf{FS}}$ and the assumption that M and $\mathcal{A}^T_{\mathsf{FS}}$ agree on all n-bit inputs, $\mathcal{A}^T_{\mathsf{FS}}$ (implicitly) enumerates all possible witnesses for $M(x)$, and forces $M(x)$ to reject all of them. This implies that M rejects x, which is a contradiction. The case where M rejects the string x can be handled similarly as well.

If we no longer have the restriction that M only guesses $n/10$ bits as the witness, then $\mathcal{A}^T_{\mathsf{FS}}$ cannot enumerate all witnesses on the same input length. Instead, we can first pad the input length to be sufficiently long (i.e. $n + T(n)$), then follow the same approach: enumerate all possible witnesses, and force $M(x)$ to reject all of them.

Encoding. Recall that we fix a natural enumeration of all nondeterministic Turing machines, and associate the integer M with the M-th such machine. In the following, we use a slightly different encoding than that in Theorem 4.6, to deal with the fact that the witness could be longer than n. For $M, n \in \mathbb{N}$ such that $n \geq M + 2$ and $z \in \{0,1\}^*$, we encode them by

$$\langle M, n, z \rangle_{\mathsf{ro}} := 1^M 0 1^{n-M-2} 0 z.$$

Note that $|\langle M, n, z \rangle_{\mathsf{ro}}| = n + |z|$.

The Algorithm $\mathcal{A}^T_{\mathsf{RO}}$. $\mathcal{A}^T_{\mathsf{RO}}$ is defined as follows.

- Given an input x, we parse it as $x = \langle M, n, z\rangle_{\mathsf{ro}}$. We reject immediately if there is no valid parsing, or $|z| > T(n)$.
- If $|z| < T(n)$, $\mathcal{A}^T_{\mathsf{RO}}$ accepts x if both of the following hold:
  1. $z = 0^\ell$ for some ℓ.
  2. M accepts $x' = \langle M, n, 0^{|z|+1}\rangle_{\mathsf{ro}}$ in $T(|x'|)$ steps.
- Otherwise, $|z| = T(n)$, and $\mathcal{A}^T_{\mathsf{RO}}$ accepts x if both of the following hold:
  1. M rejects $\langle M, n, \varepsilon\rangle_{\mathsf{ro}}$ in $T(n)$ steps with witness z.[22] That is, $\mathcal{V}_M(\langle M, n, \varepsilon\rangle_{\mathsf{ro}}, T(n), z) = 0$.
  2. (This condition is only required when $z \neq 1^{T(n)}$.) M accepts $x' = \langle M, n, z+1\rangle_{\mathsf{ro}}$ in $T(|x'|)$ steps.

By the construction above, the assumption $T(n+1) \leq O(T(n))$, and the fact that nondeterministic algorithms can be simulated with only constant overhead in running time, $\mathcal{A}^T_{\mathsf{RO}}$ is a nondeterministic algorithm running in $O(T(n))$ time.

Construction of the Refuter. Now we design the refuter $\mathcal{R}^T_{\mathsf{RO}}$. Let M be an $o(T(n))$-time nondeterministic machine. For every sufficiently large n, we want to find an input x of length $|x| \in [n, n+T(n)]$ such that $M(x) \neq \mathcal{A}^T_{\mathsf{RO}}(x)$. Let $\mathcal{L}^{(n)}$ be the list consisting of strings of the form $\langle M, n, z\rangle_{\mathsf{ro}}$ for all $z = 0^\ell$ where $\ell \in \{0, 1, 2, \ldots, T(n)-1\}$ and $z \in \{0,1\}^{T(n)}$, sorted in lexicographical order.

The following lemma can be proved similarly as Lemma 4.3.

Lemma B.1. *If M runs in $o(T(n))$. For every sufficiently large n and integer $1 \leq i \leq |\mathcal{L}^{(n)}|$, if $\mathcal{L}^{(n)}_i = \langle M, n, z\rangle_{\mathsf{ro}}$, then*

$$\mathcal{A}^T_{\mathsf{RO}}(\mathcal{L}^{(n)}_i) = \begin{cases} M(\mathcal{L}^{(n)}_{i+1}) & |z| < T(n), \\ \left[\mathcal{V}_M(\mathcal{L}^{(n)}_1, T(n), z) = 0\right] \wedge M(\mathcal{L}^{(n)}_{i+1}) & |z| = T(n) \text{ and } i < |\mathcal{L}^{(n)}|, \\ \left[\mathcal{V}_M(\mathcal{L}^{(n)}_1, T(n), z) = 0\right] & i = |\mathcal{L}^{(n)}|. \end{cases}$$

Applying Lemma B.1, one can verify that the same algorithm in the proof of Theorem 4.6 also works given the list

$$\mathcal{L}^{(n)}_1, \mathcal{L}^{(n)}_2, \ldots, \mathcal{L}^{(n)}_{|\mathcal{L}^{(n)}|}.$$

Finally, note that $\mathcal{R}^T_{\mathsf{RO}}$ runs in

$$O(\log |\mathcal{L}^{(n)}|) \cdot \mathrm{poly}(T(\mathrm{poly}(T(n)))) = \mathrm{poly}(T(\mathrm{poly}(T(n))))$$

time, which completes the proof.

□

[22] We use ε to denote empty string.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il