Electronic Colloquium on Computational Complexity, Report No. 183 (2022)

New Lower Bounds and Derandomization for ACC, and a Derandomization-centric View on the Algorithmic Method

Lijie Chen*

December 19, 2022

Abstract

In this paper, we obtain several new results on lower bounds and derandomization for ACC^0 circuits (constant-depth circuits consisting of $\mathsf{AND}/\mathsf{OR}/\mathsf{MOD}_m$ gates for a fixed constant m, a frontier class in circuit complexity):

1. We prove that any polynomial-time Merlin-Arthur proof system with an ACC^0 verifier (denoted by $\mathsf{MA}_{\mathsf{ACC}^0}$) can be simulated by a nondeterministic proof system with quasi-polynomial running time and polynomial proof length, on infinitely many input lengths. This improves the previous simulation by [Chen, Lyu, and Williams, FOCS 2020], which requires both quasi-polynomial running time and proof length.
2. We show that $\mathsf{MA}_{\mathsf{ACC}^0}$ cannot be computed by fixed-polynomial-size ACC^0 circuits, and our hard languages are hard on a sufficiently dense set of input lengths.
3. We show that NEXP (nondeterministic exponential-time) does not have ACC^0 circuits of *sub-half-exponential* size, improving the previous *sub-third-exponential* size lower bound for NEXP against ACC^0 by [Williams, J. ACM 2014].

Combining our first and second results gives a conceptually simpler and derandomization-centric proof of the recent breakthrough result $\mathsf{NQP} \coloneqq \mathsf{NTIME}[2^{\mathrm{polylog}(n)}] \not\subset \mathsf{ACC}^0$ by [Murray and Williams, SICOMP 2020]: Instead of going through an easy witness lemma as they did, we first prove an ACC^0 lower bound for a subclass of MA, and then derandomize that subclass into NQP, while retaining its hardness against ACC^0.

Moreover, since our derandomization of $\mathsf{MA}_{\mathsf{ACC}^0}$ achieves a polynomial proof length, we indeed prove that nondeterministic quasi-polynomial-time with $n^{\omega(1)}$ nondeterminism bits (denoted as $\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(n)}, n^{\omega(1)}]$) has no $\mathrm{poly}(n)$-size ACC^0 circuits, giving a new proof of a result by Vyas. Combining with a win-win argument based on *randomized encodings* from [Chen and Ren, STOC 2020], we also prove that $\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(n)}, n^{\omega(1)}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size ACC^0 circuits, improving the recent strongly average-case lower bounds for NQP against ACC^0 by [Chen and Ren, STOC 2020].

One interesting technical ingredient behind our second result is the construction of a PSPACE-complete language that is paddable, downward self-reducible, same-length checkable, and weakly error correctable. Moreover, all its reducibility properties have corresponding $\mathsf{AC}^0[2]$ non-adaptive oracle circuits. Our construction builds and improves upon similar constructions from [Trevisan and Vadhan, Complexity 2007] and [Chen, FOCS 2019], which all require at least TC^0 oracle circuits for implementing these properties.

*Miller Institute for Basic Research in Science at University of California, Berkeley, CA. Email: wjmzbmr@gmail.com

ISSN 1433-8092

Contents

1 Introduction

Background. 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 holy grails in complexity theory and theoretical computer science. As the first step toward lower bounds for general circuits, constant-depth circuits received a lot of attention in the 1980s, with classical works culminating in exponential lower bounds against AC^0 [Ajt83, FSS84, Yao85, Hås89] (constant-depth circuits consisting of unbounded fan-in AND/OR gates) and $\mathsf{AC}^0[q]$ (AC^0 circuits with MOD_q gates) for prime power q [Raz87, Smo87].

Unfortunately, after the 1980s, the initial successes met an obstacle: lower bounds for $\mathsf{AC}^0[m]$ have been extremely difficult to establish for composite m, despite the conjecture that $\mathsf{AC}^0[m]$ cannot compute the majority function. In fact, it had been a notorious open question whether NEXP (nondeterministic exponential time) has polynomial-size ACC^0 circuits (ACC^0 denotes the union of $\mathsf{AC}^0[m]$ for all constants m), until a decade ago Williams [Wil11, Wil14] finally proved such a lower bound, via an *algorithmic* approach to circuit lower bounds [Wil10, Wil13a]. Combining many classical results from complexity theory, such as the nondeterministic time hierarchy theorem [SFM78, Žák83], hardness vs. randomness [NW94], and the PCP theorem [ALM+98, AS98], Williams' work shows how non-trivial circuit-analysis algorithms can be generically applied to prove circuit lower bounds.

In 2018, Murray and Williams [MW18] significantly improved Williams' lower bound by showing $\mathsf{NQP} := \mathsf{NTIME}[2^{\mathrm{polylog}(n)}]$ is not contained in ACC^0. A line of recent work [COS18, CW19, Che19, CR20] generalized the algorithmic approach to the average-case setting, culminating in the result that NQP cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by $\mathrm{poly}(n)$-size ACC^0 circuits [CR20].[1]

Motivation: making the algorithmic method direct. Most of the proofs following the algorithmic method are indirect and make heavy use of *proof by contradiction* or *win-win analysis*.[2] These proofs are considered by some to be conceptually hard to understand. A natural question is whether we can give a more direct (*i.e.*, no win-win analysis or proof by contradiction) proof of the separation $\mathsf{NQP} \not\subset \mathsf{ACC}^0$ [MW20], which might be (hopefully) easier to understand.

Inspired by [Wil13b, Wil16] and motivated by the goal of proving average-case lower bounds for NQP against ACC^0, Chen [Che19] proposed a derandomization-centric perspective of the algorithmic method, which consists of the following two steps:

1. Assuming that the desired lower bound is false (*e.g.*, $\mathsf{NQP} \subseteq \mathsf{ACC}^0$ or NQP is average-case easy for ACC^0), [Che19] gave a derandomization of a certain sub-class of MA into NQP.
2. [Che19] then proved that this sub-class of MA contains a language that is hard against ACC^0, and its hardness is retained after the aforementioned derandomization into NQP, from which the desired lower bound for NQP against ACC^0 follows immediately.

Following this perspective from [Che19], [CR21] managed to refine both of the two steps above to prove the strongly average-case lower bounds for NQP against ACC^0. Still, the proofs from [Che19] and [CR21] are not direct, since the first step involves a proof by contradiction.[3] Later, [CLW20] managed to give a derandomization of the sub-class of MA with ACC^0 verifier (*i.e.*, $\mathsf{MA}_{\mathsf{ACC}^0}$; see Definition 3.4 for a formal definition.)

[1] The journal versions of [MW18] and [CR20] are [MW20] and [CR21], respectively.

[2] A win-win analysis often goes as follows: for some classes $\mathcal{A}$ and $\mathcal{B}$, if $\mathcal{A} \subseteq \mathcal{B}$, then our desired lower bound follows in one way, and if $\mathcal{A} \not\subseteq \mathcal{B}$, then our desired lower bound follows in another way.

[3] It can be interpreted as a "conditional derandomization", since we can derandomize MA assuming the lower bound is false.

One may hope that we can replace the derandomization in the first step of [Che19] by a direct application of the new derandomization by [CLW20], but this does not work for the following technical reason: The specific sub-class of MA considered in [Che19] is $\mathsf{MA}_{\mathsf{NC}}$, meaning that the verifier of the Merlin-Arthur protocol has polylog(n)-depth circuits. This is (much) stronger than the class $\mathsf{MA}_{\mathsf{ACC}^0}$ that [CLW20] can derandomize. So to make [Che19]'s derandomization-centric perspective completely direct, we will need a lower bound for $\mathsf{MA}_{\mathsf{ACC}^0}$ against ACC^0, so that the derandomization of [CLW20] can be applied to the corresponding hard language.

1.1 Our Results

Let $\mathscr{C}$ be a circuit class. We use $\mathsf{MA}_{\mathscr{C}}$ to denote the sub-class of MA whose verifier can be implemented by $\mathscr{C}$ circuits. More formally, we say that $L \in \mathsf{MA}$ if there is a polynomial-time algorithm $V(x, y, z)$ with $|x| = n$ and $|y|, |z| \leq \mathrm{poly}(n)$ such that (1) $x \in L$ implies that there exists y satisfying $\Pr_z[V(x, y, z) = 1] = 1$ and (2) $x \notin L$ implies that for all y it holds that $\Pr_z[V(x, y, z) = 1] \leq 1/3$. And we say that $L \in \mathsf{MA}_{\mathscr{C}}$ if V can be computed by polynomial-size $\mathscr{C}$ circuits.[4]

1.1.1 A direct proof of $\mathsf{NQP} \not\subset \mathsf{ACC}^0$

Our first result is an affirmative answer to the question above by proving the following results.

Theorem 1.1 (Informal). *For every $k, d_\star, m_\star \in \mathbb{N}$, there is a language $L \in \mathsf{MA}_{\mathsf{ACC}^0}$ and a constant $c \in \mathbb{N}$ such that for every sufficiently large $n \in \mathbb{N}$, there exists an input $m \in [n, n^c]$ such that L_m (the restriction of L on m-bit inputs) does not have m^k-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits.*[5]

Intuitively speaking, our hard language L is not only hard against m^k-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits, but its hard input lengths are not sparse in the sense that every interval $[n, n^c]$ contains a hard input length.[6] This stronger hardness condition is crucial for the hard language to remain hard after the infinitely often derandomization from [CLW20]. Combining Theorem 1.1 and the derandomization from [CLW20], we then have an alternative proof of $\mathsf{NQP} \not\subset \mathsf{ACC}^0$.[7]

A subtle caveat. Interestingly, the direct derandomization proof above indeed only proves

$$\text{(L1)} \ : \ \mathsf{NQP} \not\subset \mathsf{AC}^0_{d_\star}[m_\star] \text{ for every } d_\star, m_\star \in \mathbb{N},$$

which is different from

$$\text{(L2)} \ : \ \mathsf{NQP} \not\subset \mathsf{ACC}^0.$$[8]

This issue occurs in [MW20] as well, as a direct application of their easy witness lemma (which is the core technical result of [MW20]) also only proves (L1). Nonetheless, [MW20] resolved this issue

[4] Our formal definition of $\mathsf{MA}_{\mathscr{C}}$ is slightly more technical and contains more languages (see Definition 3.4), but for the sake of the introduction, it might be easier to think of this simpler definition.

[5] The hard language also needs one bit of advice per input length, we omit this technical issue in the introduction. We also indeed prove a weakly average-case lower bound against $\mathsf{AC}^0_{d_\star}[m_\star]$; see Section 6 for more detail.

[6] This is weaker than the "almost almost-everywhere" hardness notion in [MW20].

[7] We remark that there are previous papers [JMV15, BV14] that simplifies some parts of Williams' original proof [Wil11]. However, these works do not change the high-level structure of Williams' proof: they still argue by a proof of contradiction using an easy witness lemma. Our goal here is to eliminate the proof by contradiction completely.

[8] (L2) implies that there is *fixed* language $L \in \mathsf{NQP}$ such that $L \notin \mathsf{AC}^0_{d_\star}[m_\star]$ for every $d_\star, m_\star \in \mathbb{N}$, which *a priori* looks stronger than (L1).

by proving that (L1) implies (L2) via a simple win-win analysis.[9] Unfortunately, we do not know how to get rid of this particular win-win analysis and leave this as an intriguing open question. Still, we believe that a direct proof of (L1) is already sufficiently interesting.

In Section 2, we give a *detailed overview* of new derandomization-centric alternative proofs for the main results of both [MW20] and [CR21].

1.1.2 An improved derandomization of $\mathsf{MA}_{\mathsf{ACC}^0}$

Our next result is an improvement of the derandomization of $\mathsf{MA}_{\mathsf{ACC}^0}$ from [CLW20]. Below, $\mathsf{NTIMEGUESS}[T(n), G(n)]$ denotes the class of languages computable by nondeterministic algorithms with $T(n)$ time and $G(n)$ bits of nondeterminism; see Definition 3.2 for a formal definition.

Theorem 1.2.

$$\mathsf{MA}_{\mathsf{ACC}^0} \subset \textit{i.o.}\text{-}\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(n)}, \mathrm{poly}(n)].$$

Theorem 1.2 is most interesting when viewed as a derandomization of randomized proof systems. It says that we can derandomize Merlin Arthur proof systems with ACC^0 verifier into a nondeterministic proof system with roughly the same proof length, albeit the running time goes up to a quasi-polynomial of the original running time. We hope such derandomization might have some applications in the derandomization of some recently purposed algebraic proof systems whose verifiers are randomized (*i.e.*, they are MA proof systems), for example [GP18].

As a consequence of Theorem 1.2, we give another proof of a result by [Vya19].

Corollary 1.3 ([Vya19])**.** *For every $\alpha(n) \geq \omega(1)$,*

$$\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(n)}, n^{\alpha(n)}] \not\subset \mathsf{ACC}^0.$$

Combining the win-win argument from [CR21], we also strengthen Corollary 1.3 to the average-case, thus improving on [CR21].

Theorem 1.4. *There is a $\beta \in \mathbb{N}$ such that for every $\alpha(n) \geq \omega(1)$, $\mathsf{NTIMEGUESS}[2^{\log^\beta n}, n^{\alpha(n)}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by ACC^0.*

Indeed, similar to all previous works following the algorithmic method, we prove a generic connection between circuit analysis algorithm and $\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(n)}, n^{\omega(1)}]$ lower bounds; see Section 5 for formal statements.

1.1.3 An improved lower bound for NEXP against ACC^0

Our third result is an improvement over the original lower bound for NEXP against ACC^0 in [Wil14]. Roughly speaking, we say that a reasonable time-bound[10] function $g(n)$ is *sub-half-exponential* if for every $k \in \mathbb{N}$, $g(g(n)^k)^k \leq 2^{n^{o(1)}}$, and we say call g *sub-third-exponential* if $g(g(g(n)^k)^k)^k \leq 2^{n^{o(1)}}$ for every $k \in \mathbb{N}$. In [Wil14], it was proved that NEXP has no sub-third-exponential-size ACC^0 circuits. We improve that size bound to be sub-half-exponential.

Theorem 1.5. *For every sub-half-exponential reasonable time-bound function $g(n)$, NE has no $g(n)$-size ACC^0 circuits.*[11]

[9] If $\mathsf{P} \not\subset \mathsf{ACC}^0$, clearly (L2) is true. If $\mathsf{P} \subset \mathsf{ACC}^0$, then $\mathsf{P}_{/\mathrm{poly}}$ collapsed to $\mathsf{AC}^0_{d_\star}[m_\star]$ for some fixed $d_\star, m_\star \in \mathbb{N}_{\geq 1}$, hence (L1) implies $\mathsf{NQP} \not\subset \mathsf{P}_{/\mathrm{poly}}$.

[10] see Section 3 for a formal definition.

[11] Here $\mathsf{NE} := \mathsf{NTIME}[2^{O(n)}]$.

Although our proof of Theorem 1.5 is still an indirect argument via a win-win analysis that is similar to and inspired by [Wil16], we managed to nicely abstract out the indirect part by the following lemma. Recall that $\mathsf{E} := \mathsf{TIME}[2^{O(n)}]$ denotes the class of languages computable by single exponential-time deterministic algorithms.

Lemma 1.6 (Informal). *Let $g(n)$ be a sub-half-exponential reasonable time-bound function. Assume that E has $g(n)$-size ACC^0 circuits. Then*

$$\mathsf{MATIME}_{\mathsf{ACC}^0}\left[2^{n^{o(1)}}\right] \not\subset \textit{i.o.-}\mathsf{SIZE}[g(n)].$$

Theorem 1.5 then follows immediately by the following win-win argument: If E has no $g(n)$-size ACC^0, then we are done; otherwise, we apply our quasi-polynomial-time derandomization to show

$$\mathsf{MATIME}_{\mathsf{ACC}^0}[2^{n^{o(1)}}] \subseteq \text{i.o.-}\mathsf{NTIME}[2^n],$$

which immediately implies that $\mathsf{NTIME}[2^n] \not\subset \mathsf{SIZE}[g(n)]$.

1.1.4 An improved construction of PSPACE-complete language

Finally, as a key technical ingredient behind the proof of Theorem 1.1, we construct a PSPACE-complete language with several nice reducibility properties that can all be implemented by non-adaptive $\mathsf{AC}^0[2]$ circuits; see Section 3.5 for formal definitions of these properties.

Theorem 1.7. *There is a PSPACE-complete language L^{PSPACE} that is paddable, non-adaptive $\mathsf{AC}^0[2]$ downward self-reducible, non-adaptive $\mathsf{AC}^0[2]$ same-length checkable and non-adaptive $\mathsf{AC}^0[2]$ weakly error correctable.*

Theorem 1.7 improved upon a similar construction from [Che19, CR21] (following [TV07, San09]), which gave a PSPACE-complete language that is paddable, non-adaptive TC^0 downward self-reducible, non-adaptive TC^0 same-length checkable and non-adaptive TC^0 weakly error correctable.

1.2 Intuition

Let us briefly discuss the intuition behind our results.

Lower bounds for $\mathsf{MA}_{\mathsf{ACC}^0}$ and construction of PSPACE-complete languages. We will first discuss the intuition and technical challenges behind the proof of Theorem 1.1 and Theorem 1.7. Our proof of Theorem 1.1 follows the proof of a similar statement from [Che19].[12] In the following, we will ignore all minor details and only focus on the part relevant to us. In the framework of [Che19], to prove a lower bound against $\mathscr{C}$ circuits, the verifier of the constructed hard MA language first guesses a $\mathscr{C}$ circuit C of a certain size and then runs the instance-checker of the PSPACE-complete language from [Che19]. Thus, because the PSPACE-complete language from [Che19] only admits a TC^0 instance checker, the complexity of the verifier above is at least TC^0, even if we only aim to prove lower bounds against weaker classes such as ACC^0.

Since our new Theorem 1.7 improves the instance-checker complexity of the PSPACE-complete language to $\mathsf{AC}^0[2]$, we managed to prove Theorem 1.1 by plugging this new ingredient into the old framework of [Che19]. Along the way, we need several clever technical manipulations.[13]

[12] Roughly speaking, [Che19] proved that $\mathsf{MATIME}_{\mathsf{NC}}[2^{\mathrm{polylog}(n)}]$ has no $\log^d n$-depth circuits for any fixed d.

[13] *e.g.*, Lemma 6.2 and the choice of working with a specific sub-class of ACC^0 called $\widetilde{\mathsf{AC}}^0_d[m]$

Next, we explain the ideas and technical challenges behind our proof of Theorem 1.7. It is instructive to recap some relevant details from the construction of the PSPACE-complete language L^{Che19} from [Che19]. Roughly speaking, on input length n, L^{Che19} computes a polynomial $P_n \colon \mathbb{F}_n^{m(n)} \to \mathbb{F}_n$, where $\mathbb{F}_n = \mathsf{GF}(2^{r(n)})$ for some $r(n) \in \mathbb{N}_{\geq 1}$. Omitting many details,[14] the computational bottleneck in implementing these reducibility properties (say, instance-checkability) is indeed *polynomial interpolation* on a single variable. That is, for example, for some fixed $x_2, \ldots, x_{m(n)} \in \mathbb{F}_n$, we wish to interpolate a polynomial $p \colon \mathbb{F}_n \to \mathbb{F}_n$ such that $p(x) = P_n(x, x_2, \ldots, x_n)$ for every $x \in \mathbb{F}_n$, given oracle access to P_n.

A direct algorithm for the task above first queries P_n to obtain $P_n(z_\ell, x_2, \ldots, x_n)$ for $\ell \in [d+1]$, where z_ℓ is the ℓ-th element in $\mathbb{F}_n$ and d is the degree of P_n, and then uses Lagrange interpolation to obtain the description of p. Unfortunately, the Lagrange interpolation (as well as other interpolation methods) requires multiplying at least d elements from $\mathbb{F}_n$ together. Since $d \geq n$ in [Che19], we will need a TC^0 circuit [HAB02, HV06] to compute the interpolated polynomial p.

To improve the complexity of computing p to $\mathsf{AC}^0[2]$, we observe that for the argument above to work, d only has to be greater than the *maximum individual degree* of P_n as opposed to the total degree.[15] Moreover, interpolation over $\mathbb{F}_n$ for a constant-degree polynomial can be done in $\mathsf{AC}^0[2]$ (see Corollary 7.2, which supports up to $\log n$ degree). Hence, we made several careful non-trivial modifications to the language L^{Che19} so that the corresponding polynomials always have a constant maximum individual degree while preserving the required $\mathsf{AC}^0[2]$ downward self-reducibility. These modifications allow us to implement the instance checker in $\mathsf{AC}^0[2]$; see Section 7 for more detail.

Improved derandomization of $\mathsf{MA}_{\mathsf{ACC}^0}$. Next we discuss the intuition behind the proof of Theorem 1.2. Fix $d_\star, m_\star \in \mathbb{N}_{\geq 1}$, our goal is to derandomize $\mathsf{MA}_{\mathsf{AC}^0_{d_\star}[m_\star]}$ into $\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(n)}, \mathrm{poly}(n)]$. Let $\mathscr{C} = \mathsf{AC}^0_{d_\star}[m_\star]$.

For the sake of gaining intuition, let us assume that we have a *magical PRG* construction G, such that when given a function $f \colon \{0,1\}^n \to \{0,1\}$ that is *worst-case* hard against $S(n)$-size $\mathscr{C}$ circuits, G^f takes $O(n)$ bits of seeds and fools all $S(n)$-size $\mathscr{C}$ circuits.[16] Then we can start from the worst-case witness lower bound against $\mathscr{C}$ from [Wil16]. Roughly speaking, [Wil16] proved that there exists $\varepsilon \in (0,1)$ and a linear time algorithm $V_{\mathsf{tt}} \colon \{0,1\}^* \to \{0,1\}$, such that for infinitely many $n \in \mathbb{N}$, $V_{\mathsf{tt}}(\mathsf{tt}(f)) = 1$ for some $f \colon \{0,1\}^n \to \{0,1\}$,[17] and for every $f \colon \{0,1\}^n \to \{0,1\}$ such that $V_{\mathsf{tt}}(\mathsf{tt}(f)) = 1$, we know that f has no 2^{n^ε}-size $\mathscr{C}$ circuits. Again, for the sake of intuition, we assume that the condition above holds for all $n \in \mathbb{N}$, instead of only for infinitely many $n \in \mathbb{N}$.

Making these two unrealistic assumptions, we can easily derandomize $\mathsf{MATIME}_{\mathscr{C}}[n]$. Let $L \in \mathsf{MATIME}_{\mathscr{C}}[n]$ and $V(x,y,z)$ be its verifier, such that $x \in L$ implies that there exists $y \in \{0,1\}^n$ such that $\Pr_z[V(x,y,z) = 1] \geq 2/3$, and $x \notin L$ implies that for all $y \in \{0,1\}^n$ we have $\Pr_z[V(x,y,z) = 1] \leq 1/3$. We can construct the following new verifier $V'(x,(y,f))$, where $2^{\log^\varepsilon |f|} = n$ (*i.e.*, $|f| = 2^{\log^{1/\varepsilon} n}$.) V' first verifies that $V_{\mathsf{tt}}(f) = 1$, and then verifies that $\Pr_{s \in O(\log |f|)}[V(x,y,G^f(s)) = 1] \geq 1/2$. We can see that V' runs in quasi-polynomial time, and indeed V' puts $L \in \mathsf{NQP}$.

However, the derandomization above requires that the whole truth-table f is given as the

[14] These polynomials are derived from the proof of $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92], following [TV07].

[15] The individual degree of a polynomial p with respect to a variable x_i, is the largest power of x_i appearing in a monomial of p.

[16] This PRG is too-good-to-be-true for two reasons: it starts from worst-case hardness instead of average-case hardness, and the circuit size it fools has no loss compared to its hardness. But we only use it to highlight the key intuition behind our proof.

[17] For a function $f \colon \{0,1\}^n \to \{0,1\}$, $\mathsf{tt}(f)$ denotes the 2^n-length string that represents the truth-table of f.

witness, which has length $|f| = 2^{\mathrm{polylog}(n)}$. To improve this, we consider a thought experiment: letting $\ell = \log |f| = \log^{1/\varepsilon} n$, what if for *some* $f\colon \{0,1\}^\ell \to \{0,1\}$ such that $V_{\mathrm{tt}}(\mathrm{tt}(f)) = 1$, f indeed has a $2^{2\cdot\ell^\varepsilon}$-size $\mathscr{C}$ circuit? Assuming this is true, then we observe that we do not have to guess the whole truth-table f in our verifier V' anymore, and can instead just guess a $2^{2\cdot\log^\varepsilon |f|} = \mathrm{poly}(n)$-size circuit $C\colon \{0,1\}^{\log^{1/\varepsilon} n} \to \{0,1\}$ and use its truth-table as f! Hence, in this thought experiment, we dramatically reduce our witness length from $2^{\mathrm{polylog}(n)}$ to $\mathrm{poly}(n)$.

Of course, what if the hypothesis in our thought experiment is not true? Namely, what if for *every* $f\colon \{0,1\}^\ell \to \{0,1\}$ such that $V(\mathrm{tt}(f)) = 1$, f has no $2^{2\cdot\ell^\varepsilon}$-size $\mathscr{C}$ circuits as well? Staring at this for a moment, one can realize that now V certifies not only 2^{ℓ^ε} hardness, but indeed $2^{2\cdot\ell^\varepsilon}$ hardness! This essentially means that we can keep doing this argument and eventually obtain a quasi-polynomial-time derandomization of $\mathsf{MA}_{\mathscr{C}}$ with only $\mathrm{poly}(n)$ witnesses.

Of course, the above is just an idealized setting that captures our key ideas; see Section 4 for detailed proofs of how we managed to get rid of the "magic PRG assumption" using machinery developed in [CLW20]. In a sense, Case (1) in the proof of Theorem 4.1 is an implementation of what we described above.

2 Overview of the Derandomization-centric Perspective on the Algorithmic Method

This section aims to give a detailed overview of the derandomization-centric perspective on the algorithmic method.

In Section 2.1, we will first provide an overview of the original proofs from [Wil14] and [MW20]. We will give a somewhat different presentation from that of [Wil14, MW20]. Our presentation is centered around the concept of *easy witness lemmas (EWLs)*, which converts *witness lower bounds* into *circuit lower bounds for nondeterministic time classes*. Thus, we can decompose the proof into two parts: first, we prove a witness lower bound; second, we apply an easy witness lemma to convert the obtained witness lower bound into a circuit lower bound for nondeterministic time classes.

Next, in Section 2.2, we give an overview of our results on circuit lower bounds for nondeterministic time classes, which follows a *derandomization-centric perspective*. Roughly speaking, our proofs are centered around *derandomization of Merlin-Arthur classes*. Our proofs for lower bounds for nondeterministic time classes can also be naturally decomposed into two parts: first, we prove a circuit lower bound for a certain subclass of Merlin-Arthur protocols; second, we derandomize the same subclass into a nondeterministic time class. To obtain average-case circuit lower bounds for nondeterministic time classes, it amounts to start from average-case lower bounds for Merlin-Arthur classes and apply a careful win-win analysis (adopted from [CR21]).

We first recall the definitions of the following standard derandomization problems.

- CAPP **(Circuit Acceptance Probability Problem) with error δ (denoted CAPP_δ):** Given a circuit C on n inputs, estimate $\Pr_{x\in_R\{0,1\}^n}[C(x) = 1]$ within an *additive error* of δ. When not explicitly stated, δ is set to be $1/3$ by default.

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

Notation. We use $\mathbb{N}$ to denote all non-negative integers and $\mathbb{N}_{\geq 1}$ to denote all positive integers. We say a circuit class $\mathscr{C}$ is **concrete** if we can talk about the $\mathscr{C}$ complexity of a single function

$f\colon \{0,1\}^n \to \{0,1\}$ (as opposed to a family of functions $\{f_n\}_{n\in\mathbb{N}_{\geq 1}}$). For example, for fixed $d, m \in \mathbb{N}_{\geq 1}$, $\mathsf{AC}^0_d[m]$ is a concrete circuit class, but AC^0 is not (because the depth can vary). We say a concrete circuit class $\mathscr{C}$ is *typical* if it is closed under (1) taking the negation of the output, (2) taking the projections of the input, and (3) flipping input bits; see Section 3.1.1 for more formal definitions.

2.1 An Overview of Williams' EWL-centered Proofs

We begin by formally stating the generic connection between non-trivial circuit analysis algorithms and lower bounds from [Wil13a, Wil14, MW20].[18]

Theorem 2.1 ([Wil13a, Wil14]). *Let $\mathscr{C}$ be a typical concrete circuit class. If there is a $2^n/n^{\omega(1)}$-time algorithm for* CAPP *of* $\mathrm{poly}(n)$*-size n-input $\mathsf{AC}^0_2 \circ \mathscr{C}$ circuits, then* $\mathsf{NE} \not\subset \mathscr{C}$.

Theorem 2.2 ([MW20]). *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. If* CAPP *for 2^{n^ε}-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ can be solved in $2^n/n^{\omega(1)}$ time, then* $\mathsf{NQP} \not\subset \mathscr{C}$.

Notation. Let $s\colon \mathbb{N} \to \mathbb{N}$ and $\mathscr{C}$ be a concrete circuit class. We say that NE *does not admit $s(n)$-size $\mathscr{C}$ witnesses*, if there exists a verifier $V(x,y)$ that takes input $|x| = n$, $|y| = 2^n$ and runs in $2^{O(n)}$ time, such that for infinitely many $x \in \{0,1\}^*$, the following hold:

1. there exists $y \in \{0,1\}^{2^{|x|}}$ such that $V(x,y) = 1$;
2. for every $y \in \{0,1\}^{2^{|x|}}$ such that $V(x,y) = 1$, it follows that $\mathsf{func}(y)$ has no $s(n)$-size $\mathscr{C}$ circuit.[19]

Moreover, we say that *unary* NE *does not admit $s(n)$-size $\mathscr{C}$ witnesses*, if for some verifier V and for infinitely many $n \in \mathbb{N}_{\geq 1}$, the above two conditions hold for $x = 1^n$. This is a stronger statement and implies that NE does not admit $s(n)$-size $\mathscr{C}$ witnesses.

2.1.1 Overview for the proof of Theorem 2.1

The easy witness lemma of [IKW02] says the following:

Lemma 2.3 (EWL for NE [IKW02]). *Let $\mathscr{C}$ be a typical concrete circuit class. If* NE *does not admit* $\mathrm{poly}(n)$*-size $\mathscr{C}$ witnesses*[20]*, then* $\mathsf{NE} \not\subset \mathscr{C}$.

Indeed, the original statement says the contrapositive of Lemma 2.3: if $\mathsf{NE} \subseteq \mathscr{C}$, then NE admits polynomial-size $\mathscr{C}$ witnesses. Hence the name of easy witness lemma (*i.e.*, NE admits small circuit implies that NE admits *easy witnesses*).[21] We present it in this way since it is clear that witness lower bounds imply circuit lower bounds.

Williams then gave a way to prove witness lower bounds from non-trivial derandomization. The lemma below is from [Wil13b], but the proof ideas are similar to that from [Wil13a, Wil14].

[18] We remark that in Theorem 2.1 and Theorem 2.2, CAPP can be replaced by Gap-UNSAT: a circuit-analysis problem that is weaker than both CAPP and SAT (see [MW20] for details). We will work with CAPP for simplicity.

[19] Here we use $\mathsf{func}(y)$ to denote the $|x|$-bit Boolean function whose truth-table is y; see Section 3 for a formal definition.

[20] More precisely, NE does not admit n^k-size $\mathscr{C}$ witnesses for every $k \in \mathbb{N}_{\geq 1}$.

[21] [IKW02] indeed talks about NEXP instead of NE; we choose to work with NE since it simplifies the discussions.

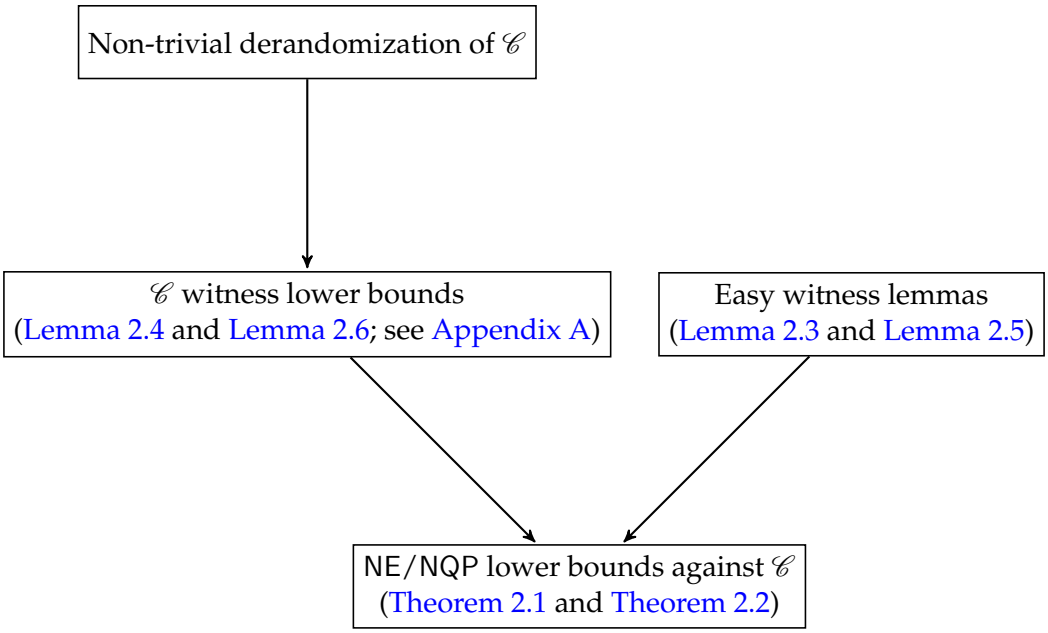

Figure 1: High-level structure of Williams' EWL-centered proofs

Lemma 2.4 ([Wil13b]). *Let $\mathscr{C}$ be a typical concrete circuit class. If* CAPP *for polynomial-size* $\mathsf{AC}^0_2 \circ \mathscr{C}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time, then unary* NE *does not admit* $\mathrm{poly}(n)$*-size* $\mathscr{C}$ *witnesses.*

Combining Lemma 2.3 and Lemma 2.4 immediately proves Theorem 2.1.

2.1.2 Overview for the Proof of Theorem 2.2

To obtain lower bounds for NQP, [MW20] proved the following easy witness lemma. Again, we state their lemma in the contrapositive.

Lemma 2.5 (EWL for NQP [MW20]). *Let $\mathscr{C}$ be a typical concrete circuit class. If* NE *does not admit* 2^{n^ε}*-size* $\mathscr{C}$ *witnesses for some* $\varepsilon \in (0,1)$*, then* $\mathsf{NQP} \not\subset \mathscr{C}$.

We note that the lemma above is weaker than the easy witness lemma for NQP in [MW20],[22] but we observe that it still suffices for circuit lower bounds for NQP. To obtain $\mathsf{NQP} \not\subset \mathscr{C}$, we need the following adaption of Lemma 2.4.

Lemma 2.6 ([Wil13b]). *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. If* CAPP *for* 2^{n^ε}*-size* $\mathsf{AC}^0_2 \circ \mathscr{C}$ *can be solved in* $2^n/n^{\omega(1)}$ *time, then unary* NE *does not admit* $2^{n^\varepsilon/2}$*-size* $\mathscr{C}$ *witnesses.*

We give a proof of Lemma 2.6 in Appendix A for completeness. Now, combining Lemma 2.5 and Lemma 2.6, Theorem 2.2 follows immediately.

2.2 NQP Lower Bounds via Derandomization of Merlin-Arthur Protocols

In the rest of this section, we give outlines of our alternative proofs of the following two results: (1) $\mathsf{NQP} \not\subset \mathsf{ACC}^0$ [MW20] and (2) there is a constant $\beta \in \mathbb{N}_{\geq 1}$ such that $\mathsf{NTIME}[2^{\log^\beta n}]$ cannot

[22] Its contrapositive says that if $\mathsf{NQP} \subset \mathscr{C}$, then NE admits 2^{n^ε}-size $\mathscr{C}$ witnesses for every $\varepsilon \in (0,1)$; this consequence is weaker than NQP admits polynomial-size $\mathscr{C}$ witnesses.

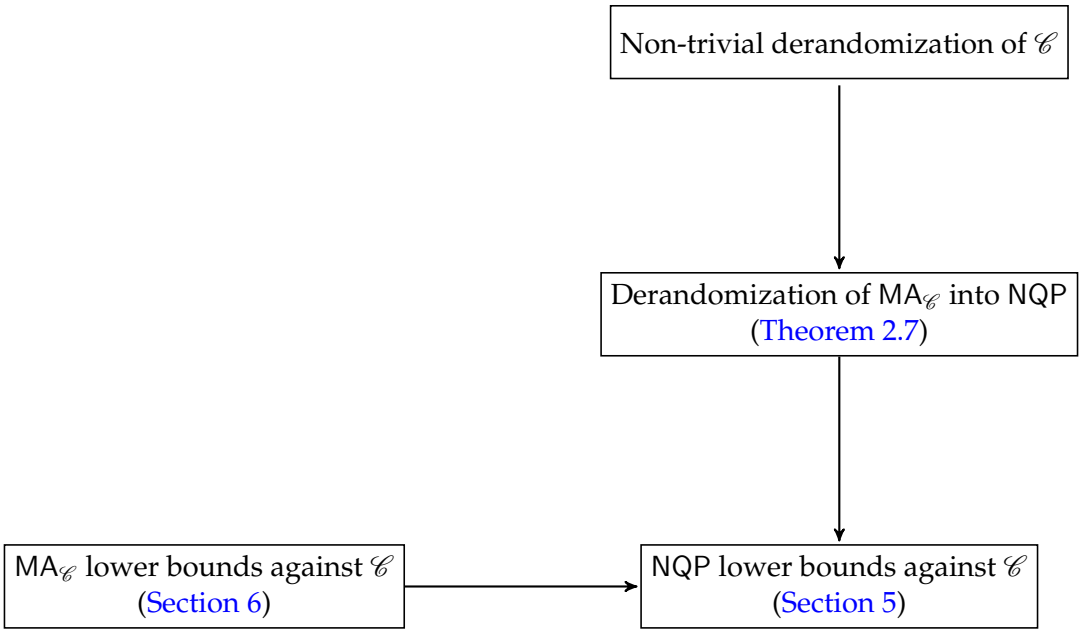

Figure 2: High-level structure of the new derandomization-centric perspective for proving NTIME lower bounds

be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size ACC^0 circuits [CR21].[23] See Theorem 2.12 and Theorem 2.16 for detailed statements. In Section 5, we prove stronger versions of Theorem 2.12 and Theorem 2.16 using our improved NPRG construction from Theorem 4.1, but the proof outlines are identical.

Derandomization of $\mathsf{MA}_{\mathscr{C}}$. Recall that $\mathsf{MA}_{\mathscr{C}}$ denotes the sub-class of Merlin-Arthur protocols whose verification can be simulated by $\mathscr{C}$ circuits for every possible witness, and NPRG is a weaker version of PRG that suffices to derandomize Merlin-Arthur protocols; see Section 3.1.3 and Section 3.2 for formal definitions.

The following theorem is from [CLW20].

Theorem 2.7 ([CLW20, Theorem 7.1])**.** *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in 2^{n-n^ε} time. Then there is a $\delta \in (0,1)$ and an infinity often NPRG for 2^{n^δ}-size $\mathscr{C}$ circuits with error 2^{-n^δ}, seed-length $\mathrm{poly}(n)$, and $2^{\mathrm{poly}(n)}$ running time. Consequently, $\mathsf{MA}_{\mathscr{C}} \subseteq$ i.o.-$\mathsf{NTIME}[2^{\log^\beta n}]$ for some $\beta \in \mathbb{N}_{\geq 1}$.*

2.2.1 NQP Lower Bounds via Derandomization

In order to apply Theorem 2.7 to get circuit lower bounds for NQP, (roughly speaking) we will prove an $\mathsf{MA}_{\mathscr{C}}$ lower bounds against $\mathscr{C}$. In more detail, we will prove the following theorem.

Theorem 2.8. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a universal constant $d_{\mathsf{v}} \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, it holds that*

$$\left(\mathsf{MA}_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2]\circ\mathscr{C}}\right)_{/1} \not\subset \mathscr{C}\text{-}\mathsf{SIZE}[n^a].$$

[23] We note that both of [MW20] and [CR21] indeed proved NQP lower bounds against $2^{\log^a n}$-size ACC^0 circuits for every $a \in \mathbb{N}_{\geq 1}$. We only consider lower bounds against all polynomial-size ACC^0 circuits in this paper for simplicity, but our proofs can be straightforwardly modified to prove the same lower bounds from [MW20, CR21].

It seems that assuming the required $\widetilde{\mathsf{CAPP}}$ algorithm for $\mathsf{AC}^0_{d_\mathsf{v}+1}[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ and applying Theorem 2.7,[24] we will be able to derandomize the hard $\left(\mathsf{MA}_{\mathsf{AC}^0_{d_\mathsf{v}}[2]\circ\mathscr{C}}\right)_{/1}$ language from Theorem 2.8 into $\mathsf{NTIME}[2^{\log^\beta n}]$ (we ignore the extra one bit of advice in the hard language for now), which *should* imply $\mathsf{NQP} \not\subset \mathscr{C}$. However, there is a huge caveat that we explain below.

Retaining the hardness after derandomization. The issue is that the $\mathsf{MA}_{/1}$ hard language L from Theorem 2.8 is only *infinitely often* hard, meaning that we only know for infinitely many input lengths $n \in \mathbb{N}_{\geq 1}$, L_n is hard against $\mathscr{C}$ circuits. Also, the derandomization of Theorem 2.7 works *infinitely often* too, in the sense that our new NQP language L' only agrees with the hard language L on infinitely many input lengths n. Let I_{hard} and I_{derand} be the input lengths that L_n is hard and $L'_n = L_n$, respectively. We see that it is possible that $I_{\mathsf{hard}} \cap I_{\mathsf{derand}} = \emptyset$, meaning that our new NQP language L' *does not retain any hardness of* L.

Following [MW20], the idea is to make both I_{hard} and I_{derand} larger so that they *must intersect at infinitely many input lengths*.

In more detail, we first strengthen Theorem 2.7 by showing the following theorem.

Theorem 2.9. *Let $\mathscr{C}$ be a typical concrete circuit class. Suppose that there is a constant $\varepsilon \in (0,1)$ and an infinity often NPRG for 2^{n^ε}-size $\mathscr{C}$ circuits with $\mathrm{poly}(n)$ seed-length and $2^{\mathrm{poly}(n)}$ running time.*

Then, there is a constant $\beta \in \mathbb{N}_{\geq 1}$ that only depends on ε such that for every $L \in (\mathsf{MA}_{\mathscr{C}})_{/1}$ and $c \in \mathbb{N}_{\geq 1}$, there is an $L' \in \mathsf{NTIME}[2^{\log^\beta n}]_{/O(\log\log n)}$ such that for infinitely many $n \in \mathbb{N}$, for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

Combining Theorem 2.7 and Theorem 2.9, we immediately have the following strengthening of Theorem 2.7.

Corollary 2.10. *Let $\mathscr{C}$ be a typical concrete circuit class and $\varepsilon \in (0,1)$. Assuming the hypothesized $\widetilde{\mathsf{CAPP}}$ algorithm from Theorem 2.7, the conclusion of Theorem 2.9 holds.*

Roughly speaking, Corollary 2.10 says that by allowing $O(\log\log n)$ bits of advice, we can enlarge I_{derand} from a set of infinitely many integers to a union of infinitely many segments of the from $[n, n^c]$, where c is a constant *of our choice*.

Next, we have the following strengthening of Theorem 2.8, which fits perfectly with the larger I_{derand} above.

Theorem 2.11. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a universal constant $d_\mathsf{v} \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, there is a constant $c \in \mathbb{N}_{\geq 1}$ and a language $L \in \left(\mathsf{MA}_{\mathsf{AC}^0_{d_\mathsf{v}}[2]\circ\mathscr{C}}\right)_{/1}$ such that, for all large enough $n \in \mathbb{N}_{\geq 1}$, there exists $m \in [n, n^c]$ such that L_m does not have m^a-size $\mathscr{C}$ circuits.*

Essentially, it says that we can enlarge I_{hard} to be a *hitting set* for all segment $[n, n^c]$: for every large enough $n \in \mathbb{N}_{\geq 1}$, $[n, n^c] \cap I_{\mathsf{hard}} \neq \emptyset$. This fits perfectly with the I_{derand} above, which consists of infinitely many segments of the form $[n, n^c]$. Hence, we have that $I_{\mathsf{hard}} \cap I_{\mathsf{derand}}$ is an infinite set.

Therefore, combining Corollary 2.10 and Theorem 2.11, we immediately have the following theorem.[25]

[24] We recommend reader to think of $\mathscr{C} = \mathsf{AC}^0_d[6]$ for some large constant $d \in \mathbb{N}_{\geq 1}$, then we only need non-trivial $\widetilde{\mathsf{CAPP}}$ for $\mathsf{AC}^0_{d+O(1)}[6]$, which follows from [Wil14, Wil18b].

[25] A direct application of Corollary 2.10 yields a hard language in $\mathsf{NQP}_{/O(\log\log n)}$ instead of just NQP. Those advice can nonetheless be removed via a straightforward *enumeration trick* (from [COS18]); see Section 5.5 for details.

Theorem 2.12. *Let $\mathscr{C}$ be a typical concrete circuit class, $\varepsilon \in (0,1)$, and $d_{\mathsf{v}} \in \mathbb{N}_{\geq 1}$ be the constant from Theorem 2.11. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AC}^0_{d_{\mathsf{v}}+1}[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in 2^{n-n^ε}-time. Then $\mathsf{NQP} \not\subset \mathscr{C}$.*

As a corollary from the theorem above and Williams' #SAT algorithm for ACC^0 [Wil14, Wil18b], we immediately have that $\mathsf{NQP} \not\subset \mathsf{AC}^0_{d_\star}[m_\star]$ for every $d_\star, m_\star \in \mathbb{N}$, which implies $\mathsf{NQP} \not\subset$, as discussed in Section 1.1.1.

2.2.2 Average-case Lower Bounds for NQP via Randomized Encodings

Finally, we strengthen Theorem 2.12 to give average-case lower bounds as well. Given the discussions above, it seems that we can simply strengthen the $\mathsf{MA}_{\mathscr{C}}$ lower bounds of Theorem 2.11 to average-case, and our derandomization from Corollary 2.10 would immediately imply average-case lower bounds for NQP.

We are indeed able to strengthen Theorem 2.11 to an average-case lower bound, but only with very weak inapproximability.

Theorem 2.13. *Let $\mathscr{C}$ be a typical concrete circuit class. There are universal constants $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, there is a constant $c \in \mathbb{N}_{\geq 1}$ and a language $L \in \left(\mathsf{MA}_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2]\circ\mathscr{C}}\right)_{/1}$ such that, for all large enough $n \in \mathbb{N}_{\geq 1}$, there exists $m \in [n, n^c]$ such that L_m cannot be $(1-m^{-\tau})$-approximated by m^a-size $\mathscr{C}$ circuits.*[26]

Combining with Corollary 2.10, we immediately have the following theorem.

Theorem 2.14. *Let $\mathscr{C}$ be a typical concrete circuit class, $\varepsilon \in (0,1)$, and $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ be the constants from Theorem 2.13. Suppose that $\widetilde{\mathsf{CAPP}}$ of 2^{n^ε}-size $\mathsf{AC}^0_{d_{\mathsf{v}}+1}[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be solved in 2^{n-n^ε} time. Then NQP cannot be $(1-n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.*

To improve the inapproximability of Theorem 2.14 from $1-n^{-\tau}$ to $1/2+1/\mathrm{poly}(n)$, we wish to perform some *mild-to-strong average-case hardness amplification* (*e.g.*, an XOR Lemma; see Lemma 5.16). Unfortunately, we currently *do not* have such an amplification for weak circuit classes such as ACC^0 (and there are barriers against such possibilities, see, *e.g.*, [SV10, GSV18]).

Chen and Ren [CR21] overcame the issue above with a clever win-win argument based on *randomized encodings* [IK02, AIK06] and *approximate linear sums*. Recall that for a $\mathsf{Sum} \circ \mathscr{C}$ circuit $L = \sum_{i\in[m]} \alpha_i \cdot C_i$ (where each C_i is a $\mathscr{C}$ circuit; see Section 3.1.2 for a formal definition), the complexity of L is defined as

$$\max\left(\sum_{i\in[m]} |\alpha_i|, \sum_{i\in[m]} \mathsf{SIZE}(C_i)\right).$$

For a function $F\colon \{0,1\}^n \to \{0,1\}$, we say that F admits a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity S, if there exists a $\mathsf{Sum} \circ \mathscr{C}$ circuit L with complexity at most S, such that $|F(x) - L(x)| \leq \delta$ for every $x \in \{0,1\}^n$.

Using the techniques from randomized encodings, [CR21] proved the following win-win result.

Lemma 2.15 ([CR21])**.** *Let $\mathscr{C}$ be a typical concrete circuit class. There is a language $L \in \mathsf{P}$ such that one of the following holds:*

[26] We remark that in Section 5 we will indeed prove a stronger version where the hard language L is in $\left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2]\circ\mathscr{C}}\right)_{/1}$; see Theorem 5.3. We will discuss why this is needed at the end of this subsection.

1. *For every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*
2. *There is a constant $\gamma \in \mathbb{N}_{\geq 1}$ such that every S-size formula admits a $\widetilde{\mathsf{Sum}}_{0.01} \circ \mathscr{C}$ circuit of complexity S^γ.*

In other words, either (Item (1)) we already have strongly average-case lower bounds for P against $\mathscr{C}$, or (Item (2)) formulas can be simulated by $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ with a polynomial blow-up in size. The key observation now is that an NPRG that fools $\mathscr{C}$ circuits with *a small error* also fools functions admitting low-complexity $\widetilde{\mathsf{Sum}} \circ \mathscr{C}$ circuits. Hence, now we are able to perform the following win-win analysis:

- Suppose Item (1) of Lemma 2.15 holds. Then it immediately follows that NQP cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.
- Otherwise, Item (2) of Lemma 2.15 holds. Then under the condition of Theorem 2.7, we would have i.o. NPRG for *formulas* (note that the i.o. NPRG from Theorem 2.7 indeed has a small error). Applying Theorem 2.9, this implies that we can now derandomize $\mathsf{MA}_{\mathsf{Formula}}$ as follows:

  There is a constant $\beta \in \mathbb{N}_{\geq 1}$ such that for every $L \in (\mathsf{MA}_{\mathsf{Formula}})_{/1}$ and every $c \in \mathbb{N}_{\geq 1}$, there is an $L' \in \mathsf{NTIME}[2^{\log^\beta n}]_{/O(\log \log n)}$ such that for infinitely many $n \in \mathbb{N}$, for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

  Noting that formulas are closed under taking an $\mathsf{AC}^0[2]$ circuit at the top, we now can use the derandomization above together with Theorem 2.13 to obtain an NQP language that is $(1 - n^{-\tau})$-hard against polynomial-size formulas, which can then be amplified to $(1/2 + 1/\mathrm{poly}(n))$-hardness against formulas, using mild-to-average-case hardness amplification for formulas.

  If we further assume that $\mathscr{C}$ can be simulated by $\mathsf{Formula}$, then we also have that NQP cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

To summarize, we have the following theorem.

Theorem 2.16 (strong average-case lower bound for NQP via an additional win-win argument)**.** *Let $\mathscr{C}$ be a typical concrete circuit class that can be simulated by formulas. Suppose that for some $\eta \in (0, 1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in $2^{n - n^\eta}$ time. Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that $\mathsf{NTIME}[2^{\log^\beta n}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.*

Note that applying the theorem above directly only shows that for every $d_\star, m_\star \in \mathbb{N}_{\geq 1}$, there is $\beta \in \mathbb{N}_{\geq 1}$ that depends on $d_\star, m_\star$ such that $\mathsf{NTIME}[2^{\log^\beta n}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits. To swap the quantifiers before $d_\star, m_\star$ and β, we can apply another win-win analysis from [CR21]; see Section 5.6 for details.

Finally, we give one technical remark.

Mild-to-strong hardness amplification requires $(\mathsf{N} \cap \mathsf{coN})\mathsf{QP}$ lower bounds. Recall that we wish to apply an XOR Lemma (Lemma 5.16) to the mildly average-case hard NQP language L from Theorem 2.14. This causes a subtle issue: $L^{\oplus 2}(x, y) := L(x) \oplus L(y)$ may not be in NQP, since to certify $L^{\oplus 2}(x, y) = 1$, one needs to prove exactly one of $L(x)$ and $L(y)$ is 1 and the other one is 0; we cannot prove (say) $L(y) = 0$ since this requires that $L \in \mathsf{coNQP}$.

To resolve this issue, we wish to get a mildly average-case hard language L from $(\mathsf{N}\cap\mathsf{coN})\mathsf{QP}$ instead of NQP. It is easy to see that the derandomization from Corollary 2.10 also derandomize $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$ languages into $(\mathsf{N}\cap\mathsf{coN})\mathsf{QP}_{/O(\log\log n)}$ languages. Hence, we strengthen Theorem 2.13 so that the hard languages now belong to $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$; see Theorem 5.3 for details.

3 Preliminaries

We use $\mathbb{N}$ to denote all non-negative integers and $\mathbb{N}_{\geq 1}$ to denote all positive integers. We use $\mathsf{GF}(p^r)$ to denote the finite field of size p^r, where p is a prime and r is an integer. For a set X, we often use $x \in_{\mathsf{R}} X$ to denote that we pick an element x from X uniformly at random. We also use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$.

For $r, m \in \mathbb{N}$, we use $\mathcal{F}_{r,m}$ to denote the set of all functions from $\{0,1\}^r$ to $\{0,1\}^m$. For a language $L\colon \{0,1\}^* \to \{0,1\}$, we use L_n to denote its restriction on n-bit inputs. For a function $f\colon \{0,1\}^n \to \{0,1\}$, we use $\mathrm{tt}(f)$ to denote the truth-table of f (*i.e.*, $\mathrm{tt}(f)$ is a string of length 2^n such that $\mathrm{tt}(f)_i$ is the output of f on the i-th string from $\{0,1\}^n$ in the lexicographical order). For a string $Z \in \{0,1\}^{2^n}$, we use $\mathsf{func}(Z)$ to denote the unique function from $\mathcal{F}_{n,1}$ with the truth-table being Z.

Let Σ be an alphabet set. For two strings $x, y \in \Sigma^*$, we use $x \circ y$ to denote their concatenation. We sometimes use $\vec{x}$ ($\vec{y}, \vec{z}$, etc.) to emphasize that $\vec{x}$ is a vector. For $\vec{x} \in \Sigma^n$ for some $n \in \mathbb{N}$, we use $\vec{x}_{<i}$ and $\vec{x}_{\leq i}$ to denote its prefix $(x_1, \ldots, x_{i-1})$ and $(x_1, \ldots, x_i)$, respectively. We also define $\vec{x}_{>i}$ and $\vec{x}_{\geq i}$ in the same way.

We call $f\colon \mathbb{N} \to \mathbb{N}$ a *reasonable time-bound function*, if f is time-constructible and increasing, and $f(cn) \leq \mathrm{poly}(f(n))$ for every constant $c \in \mathbb{N}_{\geq 1}$.

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

3.1 Complexity Classes and Basic Definitions

3.1.1 Basic Circuit Families

Unless otherwise specified, all circuits appearing in this paper consist of fan-in 2 AND/OR gates and fan-in 1 NOT gates.

A *circuit family* is an infinite sequence of circuits $\{C_n\colon \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$. A *circuit class* is a collection of circuit families. The *size* of a circuit is the number of *non*-NOT *gates* in the circuit,[27] and the size of a circuit family is a function of the input length that upper bounds the size of circuits in the family. The *depth* of a circuit is the maximum number of wires on a path from an input gate to the output gate.

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

AC^0 is the class of circuit families of constant depth and polynomial size, with $\mathsf{AND}, \mathsf{OR}$ and NOT gates, where AND and OR gates have unbounded fan-in. For an integer m, the function $\mathsf{MOD}_m\colon \{0,1\}^* \to \{0,1\}$ is one if and only if the number of ones in the input is not divisible by m. The class $\mathsf{AC}^0[m]$ is the class of constant-depth polynomial-size circuit families consisting of unbounded fan-in AND, OR, and MOD_m gates, along with unary NOT gates. We denote $\mathsf{ACC}^0 =$

[27] We do not count the number of NOT gates here for a technical reason; see our definition of typical concrete circuit classes below for more explanations.

$\cup_{m\geq 2}\mathsf{AC}^0[m]$. We also use AC^0_d (resp. $\mathsf{AC}^0_d[m]$) to denote the subclass of AC^0 (resp. $\mathsf{AC}^0[m]$) with depth at most d.

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

For $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, we define $\mathsf{Approx\text{-}MAJ}_{n,\varepsilon}$ to be the function that outputs 1 (resp. 0) if at least a $(1-\varepsilon)$ fraction of the inputs are 1 (resp. 0), and is undefined otherwise. We also use $\mathsf{Approx\text{-}MAJ}_n$ to denote $\mathsf{Approx\text{-}MAJ}_{n,1/3}$ for simplicity.

The following standard construction for the approximate-majority in AC^0 will be useful for the proofs in this paper.

Lemma 3.1 ([ABO84, Ajt90, Vio09])**.** $\mathsf{Approx\text{-}MAJ}_n$ *can be computed by uniform* AC^0_3.

We say that a circuit family $\{C_n\}_{n\in\mathbb{N}}$ is uniform, if there is a deterministic algorithm A such that $A(1^n)$ runs in time polynomial of the description length of C_n, and outputs C_n.[28]

For a circuit class $\mathscr{C}$, we say that a circuit C is a $\mathscr{C}$ oracle circuit if C is also allowed to use a special oracle gate (which can occur multiple times in the circuit, but with the same fan-in), in addition to the usual gates allowed by $\mathscr{C}$ circuits. We say that an oracle circuit is *non-adaptive* if, on any path from an input gate to the output gate, there is at most one oracle gate.[29]

Typical concrete circuit class. We say that a circuit class $\mathscr{C}$ is *concrete*, if we can talk about a single $\mathscr{C}$ circuit $C\colon \{0,1\}^n \to \{0,1\}$ for a fixed input length $n \in \mathbb{N}$. For example, AC^0 is not a concrete circuit class while AC^0_d for any fixed $d \in \mathbb{N}$ is. For two concrete circuit classes $\mathscr{C}$ and $\mathscr{D}$, we say $\mathscr{C}$ can be simulated by $\mathscr{D}$, if there exists a constant $c \in \mathbb{N}_{\geq 1}$ such that for every $n, s \in \mathbb{N}_{\geq 1}$ satisfying $s \geq n$, an s-size $\mathscr{C}$ circuit on n-bit inputs has an equivalent s^c-size $\mathscr{D}$ circuit. We use Formula and Circuit to denote the concrete circuit classes of fan-in 2 De-Morgan formulas and fan-in 2 De-Morgan circuits (*i.e.*, general circuits), respectively.

For a concrete circuit class $\mathscr{C}$ and a function $f\colon \{0,1\}^n \to \{0,1\}$, we define $\mathscr{C}\text{-}\mathsf{SIZE}(f)$ to be the minimum size of a $\mathscr{C}$ circuit computing f exactly. We say that a circuit $C\colon \{0,1\}^n \to \{0,1\}$ γ-approximates a function $f\colon \{0,1\}^n \to \{0,1\}$, if $C(x) = f(x)$ for at least a γ fraction of inputs from $\{0,1\}^n$. For a parameter $\gamma \in (1/2, 1]$, we define $\mathsf{heur}_\gamma\text{-}\mathsf{SIZE}(f)$ to be the minimum size of a circuit γ-approximating f. Note that $\mathsf{heur}_1\text{-}\mathsf{SIZE}(f) = \mathsf{SIZE}(f)$.

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

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

all have $\mathscr{C}$ circuits of size s,[30] and their corresponding circuit descriptions can be constructed in $\mathrm{poly}(s)$ time. We also require that $\mathscr{C}$ can be simulated by Circuit. That is, $\mathscr{C}$ is typical if it is closed under both *negation* and *projection*, and general circuits can simulate it up to a polynomial blow-up in size.

For two concrete circuit classes $\mathscr{C}$ and $\mathscr{D}$, we use $\mathscr{C} \circ \mathscr{D}$ to denote the class of circuits consisting of a top $\mathscr{C}$ circuit whose inputs are the output gates of some bottom $\mathscr{D}$ circuits. Note that we are

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

[29] Note that the function computed by the oracle circuit C depends on the truth-table of the oracle.

[30] Note that the straightforward construction of these circuits above can have more NOT gates compared to the original circuit C. This is why we do not count NOT gates when defining the size of circuits.

overloading the notation $\circ$ (which also denotes the string concatenation). But the meaning of the symbol $\circ$ (concatenation of strings or composition of circuit classes) will always be clear from the context.

3.1.2 Linear Sums of Circuits

We recall the definitions of $\mathsf{Sum} \circ \mathscr{C}$ circuits and their variants, which are introduced in [Wil18a] and [CW19].

$\mathsf{Sum} \circ \mathscr{C}$ denotes the following 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 $S(n)$.

For a function $F \colon \{0,1\}^n \to \{0,1\}$, we say that F admits a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity S, if there exists a $\mathsf{Sum} \circ \mathscr{C}$ circuit L with complexity at most S, such that $|F(x) - L(x)| \leq \delta$ for every $x \in \{0,1\}^n$.

3.1.3 NTIME and MATIME

We first recall the definitions of $\mathsf{NTIME}[T(n)]$ and $\mathsf{NTIMEGUESS}[T(n), G(n)]$.

Definition 3.2. *Let $T, G \colon \mathbb{N} \to \mathbb{N}$ be two time-constructible functions. A language $L \in \mathsf{NTIME}[T(n)]$ if there is an $O(T(n))$-time algorithm $V(x,y)$ such that $|x| = n$ and $|y| = T(n)$ and*

$$x \in L \Leftrightarrow \exists y \in \{0,1\}^{T(|x|)} V(x,y) = 1.$$

We call V an $\mathsf{NTIME}[T(n)]$ verifier for L. Moreover, $L \in \mathsf{NTIMEGUESS}[T(n), G(n)]$ if V only takes $G(n)$ bits of witness (i.e., $|y| = G(n)$ instead of $|y| = T(n)$), and call V an $\mathsf{NTIMEGUESS}[T(n), G(n)]$ verifier for L.

In particular, $\mathsf{NP} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}[n^k]$, and $\mathsf{NQP} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}[2^{\log^k n}]$. Also, we will often use NTG as an abbreviation for $\mathsf{NTIMEGUESS}$ for notational convenience.

MA is a randomized version of NP. We now recall the definition of $\mathsf{MATIME}[T(n)]$.

Definition 3.3. *Let $T \colon \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A language $L \in \mathsf{MATIME}[T(n)]$ if there is an $O(T(n))$-time algorithm $V(x,y,r)$ such that $|x| = n$ and $|y| = |r| = T(n)$ and*

$$x \in L \Rightarrow \exists y \in \{0,1\}^{T(|x|)} \Pr_{r \in \{0,1\}^{T(|x|)}}[V(x,y,r) = 1] = 1,$$

and

$$x \in L \Rightarrow \forall y \in \{0,1\}^{T(|x|)} \Pr_{r \in \{0,1\}^{T(|x|)}}[V(x,y,r) = 1] \leq 1/3.$$

We call V an $\mathsf{MATIME}[T(n)]$ verifier for L.

In particular, $\mathsf{MA} = \bigcup_{k \in \mathbb{N}} \mathsf{MATIME}[n^k]$.

We will also pay attention to the complexity of the verifiers in $\mathsf{MATIME}[T(n)]$. We define $\mathsf{MATIME}_{\mathscr{C}}[T(n)]$ as follows.

Definition 3.4. *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a time-constructible function, and $\mathscr{C}$ be a concrete circuit class. A language $L \in \mathsf{MATIME}_{\mathscr{C}}[T(n)]$ if there is an $\mathsf{MATIME}[T(n)]$ verifier V for L that also satisfies the following additional condition:*

- *For every $x \in \{0,1\}^n$ and $y \in \{0,1\}^{T(n)}$, $V(x,y,\cdot)$ (the restriction of V to the randomness part) has a $T(n)$-size $\mathscr{C}$ circuit.*

We call V an $\mathsf{MATIME}_{\mathscr{C}}[T(n)]$ verifier *for L.*

In particular, $\mathsf{MA}_{\mathscr{C}} = \bigcup_{k\in\mathbb{N}} \mathsf{MATIME}_{\mathscr{C}}[n^k]$. We also define

$$\mathsf{MATIME}[T(n)]_{\mathsf{AC}^0} = \bigcup_{d\in\mathbb{N}} \mathsf{MATIME}[T(n)]_{\mathsf{AC}^0_d} \text{ and } \mathsf{MATIME}[T(n)]_{\mathsf{ACC}^0} = \bigcup_{d,m\in\mathbb{N}} \mathsf{MATIME}[T(n)]_{\mathsf{AC}^0_d[m]}.$$

$\mathsf{MA}_{\mathsf{AC}^0}$ and $\mathsf{MA}_{\mathsf{ACC}^0}$ are defined similarly.

We note that the additional condition in Definition 3.4 is weaker than requiring that V itself has a $T(n)$-size $\mathscr{C}$ circuit, so it applies to more languages.

3.1.4 $\mathsf{MA} \cap \mathsf{coMA}$ and $\mathsf{NP} \cap \mathsf{coNP}$ Algorithms

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

Definition 3.5. *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A language L is in $(\mathsf{MA}\cap\mathsf{coMA})\mathsf{TIME}[T(n)]$, if there is a deterministic algorithm $V(x,y,z)$ (V is called the predicate) such that:*

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

$$\Pr_z[V(x,y,z) = L(x)] = 1.$$

- *(Soundness) For every $x \in \{0,1\}^*$ and every y,*

$$\Pr_z[V(x,y,z) = 1 - L(x)] \leq 1/3.$$

Moreover, we say that $L \in (\mathsf{MA}\cap\mathsf{coMA})\mathsf{TIME}_{\mathscr{C}}[T(n)]$, if L further satisfies the following condition:

- *For every $x \in \{0,1\}^n$ and $y \in \{0,1\}^{T(n)}$, $V(x,y,\cdot)$ (the restriction of V to the randomness part) has a $T(n)$-size $\mathscr{C}$ circuit.*

Remark 3.6. *$(\mathsf{MA}\cap\mathsf{coMA})$ (resp. $(\mathsf{MA}\cap\mathsf{coMA})_{\mathscr{C}}$) languages with advice are defined similarly, with V being an algorithm with the corresponding advice.*

Definition 3.7. *Let $T, G\colon \mathbb{N} \to \mathbb{N}$ be two time-constructible functions. A language L is in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$ (resp. $(\mathsf{N}\cap\mathsf{coN})\mathsf{TG}[T(n),G(n)]$), if there is an algorithm $V(x,y)$ (which is called the predicate) such that:*

- *V takes two inputs x, y such that $|x| = n$, $|y| = T(n)$ (resp. $|y| = G(n)$), runs in $O(T(n))$ time, and outputs an element from $\{0,1,\perp\}$.*
- *(Completeness) For all $x \in \{0,1\}^*$, there exists a y such that*

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

- *(Soundness) For all $x \in \{0,1\}^*$ and all y,*

$$V(x,y) \neq 1 - L(x).$$

Remark 3.8. *$(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$ or $(\mathsf{N}\cap\mathsf{coN})\mathsf{TG}[T(n),G(n)]$ languages with advice are defined similarly, with V being an algorithm with the corresponding advice.*

3.2 Pseudorandom Generators

Throughout the paper, we will deal with different types of pseudorandom generators (PRG). In the following, we recall their definitions.

3.2.1 PRGs and NPRGs

Let $r, m \in \mathbb{N}$ and $\varepsilon \in (0,1)$, and let $\mathcal{H} \subseteq \mathcal{F}_{m,1}$ be a set of functions. We say $G \in \mathcal{F}_{r,m}$ is a PRG for $\mathcal{H}$ with error ε, if for every $D \in \mathcal{H}$

$$\left| \Pr_{z \in_R \{0,1\}^r}[D(G(z)) = 1] - \Pr_{z \in_R \{0,1\}^m}[D(z) = 1] \right| \leq \varepsilon.$$

We call r the *seed length* of G.

We also need the notion of *nondeterministic PRGs*, defined below.

Let $w \in \mathbb{N}$. We say a pair of function $G = (G^{\mathsf{P}}, G^{\mathsf{W}})$ such that $G^{\mathsf{P}} \in \{0,1\}^w \times \{0,1\}^r \to \{0,1\}^m$ and $G^{\mathsf{W}} \in \mathcal{F}_{w,1}$ is an NPRG for $\mathcal{H}$ with error ε, if the following hold:

1. For every $u \in \{0,1\}^w$, if $G^{\mathsf{W}}(u) = 1$, then $G^{\mathsf{P}}(u,\cdot)$ is a PRG for $\mathcal{H}$ with error ε.
2. There exists $u \in \{0,1\}^w$ such that $G^{\mathsf{W}}(u) = 1$.

Here, we call r the *seed length* of G and w the *witness length* of G.

Although an NPRG, in general, does not compute *the same PRG* for different witnesses u (*i.e.*, $G^{\mathsf{P}}(u_1,\cdot)$ and $G^{\mathsf{P}}(u_2,\cdot)$ can be two different PRGs for $\mathcal{H}$), it is still useful for the derandomization of MA. We remark that the concept of NPRG is already implicit in [IKW02]. Our definition is from the journal version of [Che19].[31]

3.2.2 Family of PRGs and NPRGs

Most of the time, we will be interested in a *family of PRGs (NPRGs)* $G = \{G_n\}$ that fools a family of sets of functions $\mathcal{H} = \{\mathcal{H}_n\}$. In this case, for seed length $r\colon \mathbb{N} \to \mathbb{N}$, error $\varepsilon\colon \mathbb{N} \to (0,1)$, output length $m\colon \mathbb{N} \to \mathbb{N}$ and witness length $w\colon \mathbb{N} \to \mathbb{N}$, we say $G = \{G_n\}$ is a PRG (resp. NPRG) family for $\mathcal{H} = \{\mathcal{H}_n\}$ if for every $n \in \mathbb{N}$, (1) $\mathcal{H}_n \subseteq \mathcal{F}_{m(n),1}$ (2) G_n is a PRG (resp. NPRG) for $\mathcal{H}_n$ with error $\varepsilon(n)$, seed length $r(n)$ (and witness length $w(n)$ for G being an NPRG).

Let $\mathcal{I} \subseteq \mathbb{N}_{\geq 1}$. We call G a PRG with range $\mathcal{I}$ (resp. NPRG with range $\mathcal{I}$) for $\mathcal{H}$ if the two conditions above hold for every $n \in \mathcal{I}$. When $\mathcal{I}$ is an infinite set, we also say that G is an i.o. PRG (resp. i.o. NPRG) family for $\mathcal{H}$.

We say that a PRG $G = \{G_n\}$ is computable in $T\colon \mathbb{N} \to \mathbb{N}$ time, if there is a uniform algorithm $A\colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}$ such that A_n (meaning the first input of A is fixed to n) computes G_n in $T(n)$ time. Similarly, we say an NPRG $G = \{G_n\}$ is computable in $T\colon \mathbb{N} \to \mathbb{N}$ time, if there are two uniform algorithms $A^{\mathsf{P}}\colon \mathbb{N} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ and $A^{\mathsf{W}}\colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}$

[31] See http://www.mit.edu/~lijieche/Che19-journal-version.pdf for the draft.

such that A_n^{P} computes G_n^{P} and A_n^{W} computes G_n^{W}, both in $T(n)$ time. Note that a $T(n)$-time computable NPRG G also has witness length at most $T(n)$. So if we do not specify the witness length parameter, it is the running time T by default.

We can similarly define PRG and NPRG computable with $a\colon \mathbb{N} \to \mathbb{N}$ bits of advice by allowing algorithm A_n (resp. A_n^{P} and A_n^{W}) to use $a(n)$ bits of advice. (Note that in the case of NPRG, the advice for A_n^{P} and A_n^{W} are the same.)

3.3 Derandomization of $\mathsf{MATIME}_{\mathscr{C}}[T(n)]$ from NPRGs for $\mathscr{C}$

Now we show that NPRGs are enough for the derandomization of $\mathsf{MA}_{\mathscr{C}}[T(n)]$.

Lemma 3.9. *Let $\mathcal{I} \subseteq \mathbb{N}_{\geq 1}$. Suppose there is an $s(n)$-seed-length, $w(n)$-witness-length NPRG G for $T(n)$-size $\mathscr{C}$ circuits with range $\mathcal{I}$, error $1/10$, and running time $T_G(n)$. Then, for every $L \in \mathsf{MATIME}_{\mathscr{C}}[T(n)]$, there is an $L' \in \mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$ such that L and L' agree on all n-bit inputs for every $n \in \mathcal{I}$.*

Proof. Let $L \in \mathsf{MATIME}_{\mathscr{C}}[T(n)]$ and $V(x, y, r)$ be the corresponding $\mathsf{MATIME}[T(n)]_{\mathscr{C}}$ verifier. We construct a new deterministic verifier V' as follows:

- V' takes both $y \in \{0,1\}^{T(n)}$ and $u \in \{0,1\}^{w(n)}$ as the witness. (*i.e.*, V' takes $T(n) + w(n)$ bits as the witness.)
- Accept if $G^{\mathsf{W}}(u) = 1$ and $\Pr_{r \in \{0,1\}^{s(n)}}[V(x, y, G^{\mathsf{P}}(u, r)) = 1] \geq 1/2$.

It is then easy to verify that V' is the desired $\mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$ verifier for L when the input length $n \in \mathcal{I}$, which completes the proof. ∎

There are two useful special cases of Lemma 3.9: (1) when $\mathcal{I} = \mathbb{N}_{\geq 1}$ (*i.e.*, G is an NPRG), then we have $\mathsf{MATIME}_{\mathscr{C}}[T(n)] \subseteq \mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$ and (2) when $\mathcal{I}$ is an infinite set, then we have $\mathsf{MATIME}_{\mathscr{C}}[T(n)] \subseteq \text{i.o.-}\mathsf{NTIMEGUESS}[2^{s(n)} \cdot T_G(n) \cdot T(n), T(n) + w(n)]$.

Remark 3.10. *If the NPRG mentioned in Lemma 3.9 requires $a(n)$ bits of advice to compute (for G_n). Then the resulting* NTIMEGUESS *simulations also need $a(n)$ bits of advice on n-bit inputs.*

3.4 Probabilistically Checkable Proofs (PCPs)

We need the following construction of PCPs by Ben-Sasson and Viola [BV14].

Lemma 3.11 ([BV14])**.** *Let M be an algorithm running in time $T = T(n) \geq n$ 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.*** *$2^r \leq T \cdot \mathrm{polylog} T$.*
- ***Completeness.*** *There is a polynomial-time algorithm E such that, for every $y \in \{0,1\}^{T(n)}$ such that $M(x, y)$ accepts, $E(x, y)$ outputs the truth-table of a map $\pi\colon \{0,1\}^r \to \{0,1\}$ such that for all $z \in \{0,1\}^r$, $R(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = Q(z)$.*
- ***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 $R(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = Q(z)$.*
- ***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 extremely efficient PCP: the 3CNF R and the projection Q collectively form the verifier for the PCP.

3.5 A PSPACE-complete Language with $\mathsf{AC}^0[2]$ Reducibility Properties

We first define the desired reducibility properties below.

Definition 3.12. *Let $L\colon \{0,1\}^* \to \{0,1\}$ be a language, we define the following properties:*

1. *L is $\mathscr{C}$* downward self-reducible *if there is a uniform $\mathscr{C}$ oracle circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every large enough $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$, $A^{L_{n-1}}(x) = L_n(x)$.*

2. *L is* paddable, *if there is a polynomial time computable projection Pad (i.e., each output bit is either a constant or only depends on 1 input bit), such that for all integers $1 \le n < m$ and $x \in \{0,1\}^n$, it holds that $x \in L$ if and only if $\mathsf{Pad}(x, 1^m) \in L$, where $\mathsf{Pad}(x, 1^m)$ always has length m.*

3. *L is* same-length checkable, *if there is a randomized oracle algorithm M with output in $\{0,1,\perp\}$ such that, for every input $x \in \{0,1\}^*$,*

   (a) M asks its oracle queries only of length $|x|$ and runs in $\mathrm{poly}(|x|)$ time.

   (b) M^{L_n} outputs $L_n(x)$ with probability 1.

   (c) M^O outputs an element in $\{L(x), \perp\}$ with probability at least $2/3$ for every oracle $O\colon \{0,1\}^n \to \{0,1\}$.

   We call M an instance checker *for L. Moreover, we say that L is $\mathscr{C}$ same-length checkable if there is an instance checker M that can be implemented by uniform $\mathscr{C}$ oracle circuits.*[32]

4. *L is $\mathscr{C}$* weakly error correctable, *if there is a constant $\tau_{\mathsf{wc2ac}} \ge 1$ such that for every large enough $n \in \mathbb{N}$ and for every function $\tilde{f}\colon \{0,1\}^n \to \{0,1\}$ such that $\tilde{f}$ $(1 - n^{-\tau_{\mathsf{wc2ac}}})$-approximates L_n, there exists an $n^{\tau_{\mathsf{wc2ac}}}$-size $\mathscr{C}$ oracle circuit C_n such that $C_n^{\tilde{f}}(x) = L_n(x)$ for every $x \in \{0,1\}^n$.*

Additionally, we say that L is non-adaptive $\mathscr{C}$ downward self-reducible, same-length checkable, or weakly error correctable if the corresponding $\mathscr{C}$ oracle circuits are non-adaptive.

The following PSPACE-complete language was given by [San09] (modifying a construction of Trevisan and Vadhan [TV07]).

Theorem 3.13 ([TV07, San09])**.** *There is a PSPACE-complete language L^{TV} that is paddable, TC^0 downward self-reducible, and same-length checkable.*[33]

Later, [Che19] gave a modification of the language L^{TV} that is also non-adaptive TC^0 same-length checkable, which is further modified by [CR21].

Theorem 3.14 ([Che19, CR21])**.** *There is a PSPACE-complete language L^{Che19} that is paddable, non-adaptive TC^0 downward self-reducible, non-adaptive TC^0 same-length checkable, and non-adaptive TC^0 weakly error correctable.*

In this work, we further improve the complexity of these reducibility properties to $\mathsf{AC}^0[2]$. For convenience, we sometimes allow an algorithm A to take some integers $\alpha_1, \ldots, \alpha_k$ as *input parameters*, and a Boolean string β with length at most $\mathrm{poly}(\sum_{i\in[k]} \alpha_i)$ as *input*. For simplicity we

[32] Formally, suppose M uses at most $p(n) \le \mathrm{poly}(n)$ bits of randomness on inputs of length n, and let $M(x;r)^O$ be the output of M given input x and randomness r. There is a polynomial-time algorithm A such that, for every $n \in \mathbb{N}$, $A(1^n)$ outputs a $\mathscr{C}$ oracle circuit C_n such that (1) C_n takes $n + p(n)$ bits as input and (2) $C_n(x;r)^O = M(x;r)^O$ for every $(x,r) \in \{0,1\}^n \times \{0,1\}^{p(n)}$ and oracle $O\colon \{0,1\}^n \to \{0,1\}$.

[33] [TV07] does not explicitly state the TC^0 downward self-reducible property, but it is evident from their proof.

require that $|\beta|$ only depends on $\alpha_1, \ldots, \alpha_k$. We will use $A_{\alpha_1,\ldots,\alpha_k}$ to denote the restriction of A when its input parameters are set to $\alpha_1, \ldots, \alpha_k$.

We say that A can be implemented by a uniform $\mathscr{C}$ circuit family, if there is an algorithm B such that for every $\alpha_1, \ldots, \alpha_k \in \mathbb{N}$, $B(\alpha_1, \ldots, \alpha_k)$ outputs a $\mathrm{poly}(\sum_{i\in[k]} \alpha_i)$-size $\mathscr{C}$ circuit that computes $A_{\alpha_1,\ldots,\alpha_k}$.

Theorem 3.15. *There is a* PSPACE*-complete language* L^{PSPACE} *that is paddable, non-adaptive* $\mathsf{AC}^0[2]$ *downward self-reducible, non-adaptive* $\mathsf{AC}^0[2]$ *same-length checkable and non-adaptive* $\mathsf{AC}^0[2]$ *weakly error correctable.*[34]

Moreover, there are two algorithms DSR *and* Aux *satisfying the following*[35]*:*

1. Aux *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and outputs a value from* $\{0,1\}$.
2. Aux *can be implemented by a uniform* $\mathsf{AC}^0[2]$ *circuit.*
3. DSR *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and functions* $h_1 \colon \{0,1\}^{n-1} \to \{0,1\}$ *and* $h_2 \colon \{0,1\}^n \to \{0,1\}$ *as oracles.*
4. *For every* $n \in \mathbb{N}_{\geq 1}$, $\mathsf{DSR}_n^{L_{n-1}^{\mathsf{PSPACE}},\mathsf{Aux}_n}$ *computes* L_n^{PSPACE}.
5. DSR *can be implemented by a uniform non-adaptive* $\mathsf{XOR} \circ \mathsf{AND}_3$ *oracle circuit family. In more detail,* DSR *first queries its oracles on some* projections *of the input* $\vec{x}$ *to obtain some intermediate values and then applies an* $\mathsf{XOR} \circ \mathsf{AND}_3$ *circuit on those intermediate values and the input* $\vec{x}$ *to obtain the output.*

See Section 7 for a proof of Theorem 3.15. The following corollary follows immediately from the paddable property of L^{PSPACE} in Theorem 3.15.

Corollary 3.16. *For any typical concrete circuit class* $\mathscr{C}$, $\mathscr{C}$-$\mathsf{SIZE}(L_n)$ *is non-decreasing.*

4 Nondeterministic PRGs with Short Witness Length from Non-trivial Derandomization

In this section we prove the following theorem.

Theorem 4.1. *For a typical concrete circuit class* $\mathscr{C}$*, if for some* $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ *of* 2^{n^η}*-size* $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}_2^0$ *circuits can be deterministically solved in* 2^{n-n^η} *time, then there is a constant* $\varepsilon \in (0,1)$*, size parameter* $S(n) = 2^{\Omega(n^\varepsilon)}$*, and an infinity often nondeterministic PRG for* $S(n)$*-size* $\mathscr{C}$ *circuits with* $S(n)^{-1}$*-error,* $2^{O(n^\varepsilon)}$*-witness-length,* $\mathrm{poly}(n)$*-seed-length, and* $2^{\mathrm{poly}(n)}$ *running time.*

4.1 Technical Ingredients

We begin by recalling some needed technical ingredients.

[34] When we say that L^{PSPACE} is non-adaptive $\mathsf{AC}^0[2]$ weakly error correctable, we mean it is non-adaptive $\mathsf{AC}_d^0[2]$ weakly error correctable for a universal constant $d \in \mathbb{N}$.

[35] The conditions below are stronger than only being $\mathsf{AC}^0[2]$ downward self-reducible, and will be useful in our proofs in Section 6; see Lemma 6.2.

4.1.1 Two standard PRGs

We will use two standard PRGs from the 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 4.2 ([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 concrete circuit class. There is a function $G^{\mathsf{NW}} \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ 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 any $\mathscr{C} \circ \mathsf{Junta}_a$ circuit whose top $\mathscr{C}$ circuit has at most size S, then $G^{\mathsf{NW}}(Y, \mathcal{U}_t)$ [36] ε-fools every $\mathscr{C}$ circuit of size S. That is, for any $\mathscr{C}$ circuit C of size S,*

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

Moreover, the function G^{NW} 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 PRG fooling circuits of roughly the same size.

Lemma 4.3 ([Uma03]). *There is a universal constant $g \in \mathbb{N}$ and a function $G^{\mathsf{Umans}} \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for every s and $Y \colon \{0,1\}^\ell \to \{0,1\}$, if Y cannot be computed by circuits of size s^g, then $G^{\mathsf{Umans}}(Y, \mathcal{U}_{g\ell})$ $1/s$-fools circuits of size s. That is, for all circuit C of size at most s, it holds:*

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

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

4.1.2 Hardness amplification with linear sums

We will need a technical tool from [CLW20]: an XOR Lemma based on approximate linear sums (see Section 3.1.2 for formal definitions). Now we are ready to state the needed XOR Lemma.

Definition 4.4. *Let $f \colon \{0,1\}^n \to \{0,1\}$ and $k \in \mathbb{N}_{\geq 1}$, define the function $f^{\oplus k} \colon \{0,1\}^{kn} \to \{0,1\}$ to be $f^{\oplus k}(x_1, \ldots, x_k) := \bigoplus_{i \in [k]} f(x_i)$, where $x_i \in \{0,1\}^n$ for every $i \in [k]$.*

Lemma 4.5 ([CLW20]). *Let $f \colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $\delta \in (0, \frac{1}{2})$, for any $k \in \mathbb{N}_{\geq 1}$, let $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right)$. If*

$$\mathop{\mathbb{E}}_{x \in_R \{0,1\}^n} |f(x) - H(x)| > \delta$$

for every $[0,1]\mathsf{Sum} \circ \mathscr{C}$ circuit H 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.

For notational convenience, if a function f satisfies the hardness condition in Lemma 4.5, we say f is *δ-far* from $[0,1]\mathsf{Sum} \circ \mathscr{C}\left[O\left(\frac{n \cdot s}{(\delta \cdot \varepsilon_k)^2}\right)\right]$.

[36] $\mathcal{U}_t$ denotes uniform distribution over $\{0,1\}^t$.

4.2 Proof of Theorem 4.1

Our starting point is the proof of [CLW20, Theorem 7.1]. We state the following lemma, which is implicit in their proof.

Lemma 4.6. *There is a universal constant $\delta \in (0,1)$ such that, for a typical concrete circuit class $\mathscr{C}$, if for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time, then there exists $\varepsilon \in (0,1)$ such that at least one of the following statements hold:*

1. *There is a polynomial-time algorithm $V\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that for infinitely many $n \in \mathbb{N}$, the following holds:*
   (a) *$V(1^n, w) = 1$ for some $w \in \{0,1\}^{2^n}$.*
   (b) *For every $w \in \{0,1\}^{2^n}$ such that $V(1^n, w) = 1$, it holds that $\mathsf{func}(w)$ has no 2^{n^ε}-size circuits.*
2. *There are two polynomial-time algorithms $V_{\mathsf{ckt}}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ and $H\colon \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that for infinitely many $n \in \mathbb{N}$, the following holds:*
   (a) *$V_{\mathsf{ckt}}(1^{2^n}, C) = 1$ for some 2^{n^ε}-size circuit $C\colon \{0,1\}^n \to \{0,1\}$. (Here, V_{ckt} takes the description of the circuit C as an input, which is of length $2^{O(n^\varepsilon)}$.)*
   (b) *For every 2^{n^ε}-size circuit $C\colon \{0,1\}^n \to \{0,1\}$ satisfying $V_{\mathsf{ckt}}(1^{2^n}, C) = 1$, it follows that $f_C \in \mathcal{F}_{2n,1}$, defined by $f_C(z) := H(1^n, C, z)$ for $z \in \{0,1\}^{2n}$, is δ-far from $[0,1]\mathsf{Sum} \circ \mathscr{C} \circ \mathsf{AC}^0_2[2^{n^\varepsilon}]$.*

Proof Sketch. Item (1) and Item (2) of the lemma correspond to the Case (1) and Case (2) of the proof of [CLW20, Theorem 7.1] directly. Item (1) follows immediately from the proof of [CLW20, Theorem 7.1].

To see Item (2), we observe that the lists of functions $\widetilde{Y}, \widetilde{Z}\colon \{0,1\}^\ell \to \{0,1\}$ in Case (2) of the proof of [CLW20, Theorem 7.1] do not have to be guessed, and instead their values on each input can be computed from the circuit C in $\mathrm{poly}(|C|)$ time, using [CLW20, Lemma 3.11]. ■

Now we are ready to prove Theorem 4.1.

Proof of Theorem 4.1. Let $\delta, \varepsilon \in (0,1)$ be the constants from Lemma 4.6. We analyze the following two cases separately.

Item (1) of Lemma 4.6 holds. Note that in this case, we can without loss of generality assume that $\varepsilon^{-1} \in \mathbb{N}$ simply by changing ε to $\lceil \varepsilon^{-1} \rceil^{-1}$.

Let $\mathcal{S}$ be the set of $n \in \mathbb{N}$ such that the conditions in Item (1) of Lemma 4.6 hold, and let V be the algorithm from Item (1) of Lemma 4.6. For every $n \in \mathcal{S}$, we define α_n as the minimum integer s such that there exists an s-size circuit $C\colon \{0,1\}^n \to \{0,1\}$ satisfying $V_{\mathsf{ckt}}(n, \mathsf{tt}(C)) = 1$. Note that from Item (1.a) of Lemma 4.6, α_n is well-defined for $n \in \mathcal{S}$, and from Item (1.b) of Lemma 4.6, $\alpha_n > 2^{n^\varepsilon}$.

Let $g \in \mathbb{N}$ be the constant in Lemma 4.3. Now we construct the following NPRG family $G = \{(G^{\mathsf{P}}_n, G^{\mathsf{W}}_n)\}$, as follows:

- Given the parameter $n \in \mathbb{N}$. Let m be the largest integer such that $m^{\varepsilon^{-1}} \le n$ and $\ell = n - m^{\varepsilon^{-1}}$.
- If $\ell \notin \{m, m+1, \ldots, m^{\varepsilon^{-1}}\}$, define G^{P}_n and G^{W}_n be trivial functions that always output 0, with appropriate input lengths. (*i.e.*, we give up this parameter n.)
- G^{W}_n takes the description of a circuit $C\colon \{0,1\}^\ell \to \{0,1\}$ with size at most 2^{m+1}, and outputs 1 if and only if $V(1^\ell, \mathsf{tt}(C)) = 1$.

- G_n^{P} takes the description of a circuit $C\colon \{0,1\}^\ell \to \{0,1\}$ with size at most 2^{m+1}, together with a seed r of length $g \cdot \ell$, and outputs $G^{\mathsf{Umans}}(\mathsf{tt}(C), r)$.

From the description above, we note that G has seed length $O(\ell) = O(m^{\varepsilon^{-1}}) = O(n)$, witness length $\mathrm{poly}(2^{m+1}) \le 2^{O(n^\varepsilon)}$, and running time $2^{O(\ell)} \le 2^{O(n)}$. Next, we will show G is an i.o. NPRG for $2^{\Omega(n^\varepsilon)}$-size circuits (this is stronger than the required i.o. NPRG for $\mathscr{C}$ circuits).

For every sufficiently large $\ell \in \mathcal{S}$, let m be the unique integer such that $\alpha_\ell \in [2^m, 2^{m+1})$. Note that we have $m \in [\ell^\varepsilon, \ell)$. Let $n = m^{\varepsilon^{-1}} + \ell$. We claim that for some $S(m) = 2^{\Omega(m)}$, $G_n = (G_n^{\mathsf{P}}, G_n^{\mathsf{W}})$ is an NPRG for $S(m)$-size with error $S(m)^{-1}$, which completes the proof for this case since $\Omega(m) = \Omega(n^\varepsilon)$.

To see this claim, on this particular parameter n, we have $\alpha_\ell \in [2^m, 2^{m+1}]$. In particular, it means there exists a circuit $C\colon \{0,1\}^\ell \to \{0,1\}$ of size at most 2^{m+1} such that $V(1^\ell, \mathsf{tt}(C)) = 1$. Also, for every such circuit C, we know that C, as a Boolean function, has no $(2^m - 1)$-size circuits. Hence, by Lemma 4.3, it follows that G_n is an NPRG for $S(m)$-size circuits with error $S(m)^{-1}$, for some $S(m) = 2^{\Omega(m)}$.

Item (2) of Lemma 4.6 holds. Let $\mathcal{S}$ be the set of $n \in \mathbb{N}$ such that the conditions in Item (2) of Lemma 4.6 hold, and let V_{ckt} and H be the algorithms from Item (2) of Lemma 4.6.

In the following, we construct the required NPRG family $G = \{(G_n^{\mathsf{P}}, G_n^{\mathsf{W}})\}$. For $n \in \mathbb{N}$, G_n^{W} takes as input a 2^{n^ε}-size circuit $C\colon \{0,1\}^n \to \{0,1\}$ and outputs $V_{\mathsf{ckt}}(1^{2^n}, C)$.

Now, fix a sufficiently large $n \in \mathcal{S}$. For every 2^{n^ε}-size circuit $C\colon \{0,1\}^n \to \{0,1\}$ such that $G_n^{\mathsf{W}}(C) = 1$, we define $f_C\colon \{0,1\}^{2n} \to \{0,1\}$ as in the Item (2) of Lemma 4.6. By Item (2) of Lemma 4.6, f_C is δ-far from $[0,1]\mathsf{Sum} \circ \mathscr{C} \circ \mathsf{AC}^0_2[2^{n^\varepsilon}]$.

We set $k = \Theta(n^\varepsilon)$ so that $\varepsilon_k = (1-\delta)^{k-1} \cdot (1/2 - \delta) = 2^{-n^\varepsilon/10}$. By Lemma 4.5, it follows that $f_C^{\oplus k}$ cannot be $(1/2 + \varepsilon_k)$-approximated by any $\mathscr{C} \circ \mathsf{Junta}_{n^\varepsilon/10}$ circuits whose top $\mathscr{C}$ circuit has size at most $2^{n^\varepsilon/10}$.[37] Plugging $f_C^{\oplus k}$ into Lemma 4.2, we obtain a PRG fooling $S(n)$-size $\mathscr{C}$ circuits with error $S(n)^{-1}$ seed length $\ell(n) \le \mathrm{poly}(n)$, for some $S(n) = 2^{\Omega(n^\varepsilon)}$.

Finally, we define G_n^{P} so that it takes a 2^{n^ε}-size circuit $C\colon \{0,1\}^n \to \{0,1\}$ and a seed $r \in \{0,1\}^{\ell(n)}$ as the input, and outputs $G^{\mathsf{NW}}(\mathsf{tt}(f_C^{\oplus k}), r)$.

By the above discussions, we have that G is an i.o. NPRG for $S(n)$-size $\mathscr{C}$ circuits with error $S(n)^{-1}$. Now we analyze the parameters of G. From its description, the witness length and seed length are bounded by $2^{O(n^\varepsilon)}$ and $\mathrm{poly}(n)$, respectively. The running time is dominated by the running time of G^{NW}, which can be bounded by $2^{\mathrm{poly}(n)}$. This completes the proof of the whole theorem. ∎

4.3 Application: Derandomization of $\mathsf{MA}_{\mathsf{ACC}^0}$

We say a function $T\colon \mathbb{N} \to \mathbb{N}$ is *nice for* $\varepsilon \in (0,1)$, if for every $c \in (0,1)$, for all large enough $\ell \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that $T(n) \in (2^{c\cdot(\ell-1)^\varepsilon}, 2^{c\cdot\ell^\varepsilon}]$. It is easy to verify that functions such as n^k and $2^{\log^k n}$ are *nice* for every constant $\varepsilon \in (0,1)$. Applying the #SAT algorithm for ACC^0 from [Wil14, Wil18b], we immediately have the following corollary of Theorem 4.1.

Corollary 4.7. *Let $d_\star, m_\star \in \mathbb{N}_{\ge 1}$. There is a constant $\varepsilon \in (0,1)$, a function $S(n) = 2^{\Omega(n^\varepsilon)}$, and an infinity often nondeterministic PRG for $S(n)$-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits with $S(n)^{-1}$-error, $2^{O(n^\varepsilon)}$-witness-length, $\mathrm{poly}(n)$-seed-length, and $2^{\mathrm{poly}(n)}$ running time. Moreover, for every time-constructible function $T\colon \mathbb{N} \to \mathbb{N}$ that is nice for ε, we have*

$$\mathsf{MATIME}_{\mathsf{AC}^0_{d_\star}[m_\star]}[T(n)] \subseteq \textit{i.o.}\text{-}\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(T(n))}, \mathrm{poly}(T(n))].$$

[37] Here we use the fact that a Junta_a can be simulated by 2^a-size AC^0_2 circuits.

Proof. The desired i.o.-NPRG follows directly from the 2^{n-n^η}-time #SAT algorithm for $\mathsf{AC}^0_{d_\star}[m_\star]$ from [Wil14, Wil18b].

To show the "moreover" part, let $L \in \mathsf{MATIME}_{\mathsf{ACC}^0}[T(n)]$. By definition, there are $d_\star, m_\star \in \mathbb{N}$ such that $L \in \mathsf{MATIME}_{\mathsf{AC}^0_{d_\star}[m_\star]}[T(n)]$. Let $c_0, \varepsilon \in (0,1)$ and $G = \{G_n\}$ be an i.o. NPRG for $2^{c_0 \cdot n^\varepsilon}$-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits with $2^{O(n^\varepsilon)}$-witness-length, $\mathrm{poly}(n)$-seed-length, and $2^{\mathrm{poly}(n)}$ running time.

We now define another NPRG $\widetilde{G} = \{\widetilde{G}_n\}_{n\in\mathbb{N}}$ as follows: for each $n \in \mathbb{N}$, we let $\ell_n \in \mathbb{N}$ be the smallest integer such that $2^{c_0 \cdot \ell_n^\varepsilon} \geq T(n)$, and set $\widetilde{G}_n = G_{\ell_n}$. We claim that $\widetilde{G}$ is an i.o.-NPRG for $T(n)$-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits with $\mathrm{poly}(T(n))$-witness-length, $\mathrm{polylog}(T(n))$-seed-length, and $2^{\mathrm{polylog}(T(n))}$ running time. The parameters of $\widetilde{G}$ follow from its construction and the parameters of G. To see that $\widetilde{G}_n$ works for infinitely many input length n, note that if G_ℓ works and $T(n) \in (2^{c_0 \cdot (\ell_n - 1)^\varepsilon}, 2^{c_0 \cdot \ell_n^\varepsilon}]$ then $\widetilde{G}_n$ works as well (here we use the assumption that $T(n)$ is nice for ε).

Finally, $L \in \text{i.o.-}\mathsf{NTIMEGUESS}[2^{\mathrm{polylog}(T(n))}, \mathrm{poly}(T(n))]$ follows from Lemma 3.9, which completes the proof. ■

The following corollary will be useful for our proof of Theorem 1.5.

Corollary 4.8. *Let $T(n) \leq 2^{n^{o(1)}}$ be time-constructible. It holds that*

$$\mathsf{MATIME}_{\mathsf{ACC}^0}[T(n)] \subseteq \textit{i.o.-}\mathsf{NE}.$$

Proof. Let $L \in \mathsf{MATIME}_{\mathsf{ACC}^0}[T(n)]$. That is, there are constants $d_\star, m_\star \in \mathbb{N}_{\geq 1}$ such that $L \in \mathsf{MATIME}_{\mathsf{AC}_{d_\star}[m_\star]}[T(n)]$. Let $\varepsilon \in (0,1)$ be the constant from Corollary 4.7 corresponding to $d_\star, m_\star$, and $\tau \in \mathbb{N}_{\geq 1}$ be a large enough constant. Note that $\widetilde{T}(n) = 2^{n^{\varepsilon/\tau}}$ is nice for ε.

From Corollary 4.7, it follows that

$$L \in \mathsf{MATIME}_{\mathsf{AC}_{d_\star}[m_\star]}[\widetilde{T}(n)] \subseteq \text{i.o.-}\mathsf{NTIME}[2^{\mathrm{polylog}(\widetilde{T}(n))}] \subseteq \text{i.o.-}\mathsf{NE},$$

the last containment follows from the fact that τ is large enough. ■

5 Circuit Lower Bounds via Derandomization

We say a function $\alpha\colon \mathbb{N} \to \mathbb{N}$ is a nice unbounded function if α satisfies the following: (1) $\alpha(n) \geq \omega(1)$; (2) α is non-decreasing; (3) $\alpha(n)$ is computable in $O(n)$ time.[38]

In this section, we prove the following two theorems.

Theorem 5.1 (Weakly average-case lower bound for NQP via direct derandomization; a stronger version of Theorem 2.14)**.** *Let $\mathscr{C}$ be a typical concrete circuit class, and $\alpha(n)$ be a nice unbounded function. There are two universal constants $d, \tau \in \mathbb{N}_{\geq 1}$ such that the following holds. Suppose that for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AC}^0_d[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time. Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that neither*

$$\mathsf{NTG}[2^{\log^\beta n}, n^{\alpha(n)}] \quad \textit{nor} \quad (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/1}$$

can be $(1 - n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

[38] Conditions (2) and (3) may not be necessary; we include them for technical convenience.

Theorem 5.2 (Strongly average-case lower bound for NQP via an additional win-win argument; a stronger version of Theorem 2.16)**.** *Let $\mathscr{C}$ be a typical concrete circuit class that can be simulated by* Formula *and $\alpha(n)$ be a nice unbounded function. Suppose that for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time. Then, there is $\beta \in \mathbb{N}_{\geq 1}$ such that neither*

$$\mathsf{NTG}[2^{\log^\beta n}, n^{\alpha(n)}] \text{ nor } (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/1}$$

can be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

We remark that while Theorem 5.2 is stronger than Theorem 5.1 in the sense that it requires a $\widetilde{\mathsf{CAPP}}$ algorithm for a weaker circuit class ($\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ vs. $\mathsf{AC}^0_d[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$) and gives stronger average-case lower bounds ($1/2 + 1/\mathrm{poly}(n)$ vs. $1/2 + n^{-\tau}$),[39] the proof of Theorem 5.2 needs an additional win-win argument. In contrast, the proof of Theorem 5.1 is a straightforward direct derandomization, which we believe is easier to understand.

5.1 Technical Ingredients

We will need several technical ingredients, some already discussed in Section 2.

Theorem 5.3 (Weakly average-case lower bounds for $\mathsf{MA} \cap \mathsf{coMA}$; a stronger version of Theorem 2.13)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. There are universal constants $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ such that for all $a \in \mathbb{N}_{\geq 1}$, there is a constant $c \in \mathbb{N}_{\geq 1}$ and a language*

$$L \in \left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2] \circ \mathscr{C}}\right)_{/1}$$

such that, for all large enough $n \in \mathbb{N}_{\geq 1}$, there exists $m \in [n, n^c]$ such that L_m cannot be $(1 - m^{-\tau})$-approximated by m^a-size $\mathscr{C}$ circuits.

Reminder of Lemma 2.15. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a language $L \in \mathsf{P}$ such that one of the following holds:*

1. *For every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*
2. *There is a constant $\gamma \in \mathbb{N}_{\geq 1}$ such that every S-size formula admits a $\widetilde{\mathsf{Sum}}_{0.01} \circ \mathscr{C}$ circuit of complexity S^γ.*

Theorem 5.4 (A variant of Theorem 2.9)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. Suppose that there is a constant $\varepsilon \in (0,1)$, size parameter $S(n) = 2^{\Omega(n^\varepsilon)}$, and an infinity often NPRG for $S(n)$-size $\mathscr{C}$ circuits with $S(n)^{-1}$-error, $2^{O(n^\varepsilon)}$-witness-length, $\mathrm{poly}(n)$-seed-length, and $2^{\mathrm{poly}(n)}$ running time.*

Then, there are constants $\beta, \lambda \in \mathbb{N}_{\geq 1}$ that only depends on ε such that for every $L \in (\mathsf{MA} \cap \mathsf{coMA}_{\mathscr{C}})_{/1}$ and every $c \in \mathbb{N}_{\geq 1}$, there is an $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, \mathrm{poly}(n)]_{/\lambda \log\log n}$ such that for infinitely many $n \in \mathbb{N}$, for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

[39] We remark that Theorem 5.2 also requires $\mathscr{C}$ to be weaker than Formula, while Theorem 5.1 do not. So, strictly speaking, they are incomparable.

5.2 The Derandomization Condition and the MA Lower Bound Condition

To simplify our presentation, we define the following two conditions, capturing the derandomization consequences of Theorem 5.4 and the lower bounds from Theorem 5.3, respectively.

Definition 5.5 (The derandomization condition for $\mathscr{C}$)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. We say that the* derandomization condition *holds for $\mathscr{C}$, if the following holds:*

- *There are universal constants $\beta, \lambda \in \mathbb{N}_{\geq 1}$ such that for every $L \in ((\mathsf{MA} \cap \mathsf{coMA})_{\mathscr{C}})_{/1}$ and every $c \in \mathbb{N}_{\geq 1}$, there is an*
$$L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{\beta} n}, \mathrm{poly}(n)]_{/\lambda \log\log n}$$
such that for infinitely many $n \in \mathbb{N}$ and for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

The following is a direct corollary of Theorem 4.1 and Theorem 5.4.

Corollary 5.6. *Let $\mathscr{C}$ be a typical concrete circuit class. If for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time, then the* derandomization condition *holds for $\mathscr{C}$.*

Definition 5.7 (The MA lower bound condition for $\mathscr{C}$ and $\mathscr{D}$)**.** *Let $\mathscr{C}$ and $\mathscr{D}$ be two typical concrete circuit classes. We say that the* MA lower bound condition *holds for $\mathscr{C}$ and $\mathscr{D}$, if the following holds:*

- *Let $\tau \in \mathbb{N}_{\geq 1}$ be a universal constant. For all $a \in \mathbb{N}_{\geq 1}$, there is constant $c \in \mathbb{N}_{\geq 1}$ and a language*
$$L \in ((\mathsf{MA} \cap \mathsf{coMA})_{\mathscr{D}})_{/1}$$
such that for every $n \in \mathbb{N}$, there exists $m \in [n, n^c]$ such that $\mathsf{heur}_{(1-m^{-\tau})}$-$\mathscr{C}$-$\mathsf{SIZE}(L_m) > m^a$.

For simplicity, when $\mathscr{C} = \mathscr{D}$, we simply say that the MA lower bound condition *holds for $\mathscr{C}$.*

The following is a direct corollary of Theorem 5.3.

Corollary 5.8. *Let $\mathscr{C}$ be a typical concrete circuit class and d_{v} be the constant from Theorem 5.3. The* MA lower bound condition *holds for $\mathscr{C}$ and $\mathsf{AC}^0_{d_{\mathsf{v}}}[2] \circ \mathscr{C}$.*

5.3 Weakly Average-case Lower Bounds for NQP via Direct Derandomization

We will first prove the following weaker version of Theorem 5.1, which allows for $O(\log\log n)$ bits of advice. Later, these advice bits will be eliminated or reduced in Section 5.5.

Theorem 5.9. *Let $\mathscr{C}$ be a typical concrete circuit class and $\alpha(n)$ be a nice unbounded function. There are two universal constants $d, \tau \in \mathbb{N}_{\geq 1}$ such that the following holds. Suppose that for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AC}^0_d[2] \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time.*

Then, there are constants $\beta, \lambda \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{\beta} n}]_{/\lambda \log\log n}$ cannot be $(1 - n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

We will prove a more general result, showing that weakly average-case lower bounds for $\mathscr{C}$ follow from appropriate derandomization condition and MA lower bound condition.

Lemma 5.10. *Let $\mathscr{C}$ and $\mathscr{D}$ be two typical concrete circuit classes and $\alpha(n)$ be a nice unbounded function. Suppose that the* derandomization condition *holds for $\mathscr{D}$, and the* MA lower bound condition *holds for $\mathscr{C}$ and $\mathscr{D}$.*

Then, there are constants $\beta, \tau, \lambda \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^{\beta} n}, n^{\alpha(n)}]_{/\lambda \log\log n}$ cannot be $(1 - n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

Proof. Let $a \in \mathbb{N}_{\geq 1}$. Let $c \in \mathbb{N}_{\geq 1}$ and $L \in ((\mathsf{MA} \cap \mathsf{coMA})_{\mathscr{D}})_{/1}$ be the constant and hard language guaranteed by the MA lower bound condition for $\mathscr{C}$ and $\mathscr{D}$. Next, from the derandomization condition for $\mathscr{D}$, there are universal constants $\beta, \lambda \in \mathbb{N}_{\geq 1}$ and a language

$$L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^{\beta} n}, \mathrm{poly}(n)]_{/\lambda \log\log n}$$

such that for infinitely many n and for every $m \in [n, n^c]$, L and L' agree on all m-bit inputs.

Now, from the MA lower bound condition for $\mathscr{C}$ and $\mathscr{D}$, for every $n \in \mathbb{N}_{\geq 1}$, there exists $m \in [n, n^c]$ such that $\mathsf{heur}_{(1-m^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_m) > m^a$. Putting together with our guarantee on L', it follows that for infinitely many $m \in \mathbb{N}_{\geq 1}$, $\mathsf{heur}_{(1-m^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L'_m) > m^a$.

Since a is arbitrary and $\alpha(n)$ is a nice unbounded function, it follows that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^{\beta} n}, n^{\alpha(n)}]_{/\lambda \log\log n}$ cannot be $(1 - n^{-\tau})$-approximated by n^a-size $\mathscr{C}$ circuits for every $a \in \mathbb{N}_{\geq 1}$. ∎

Now, Theorem 5.9 follows directly from Corollary 5.6, Corollary 5.8 and Lemma 5.10.

5.4 Strongly Average-case Lower Bounds for NQP via a Win-win Argument

Next, we move to prove strongly average-case lower bounds. We will first prove the following weaker version of Theorem 5.2, which again allows for $O(\log\log n)$ bits of advice.

Theorem 5.11. *Let $\mathscr{C}$ be a typical concrete circuit class that can be simulated by* Formula *and $\alpha(n)$ be a nice unbounded function. Suppose that for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time.*

Then, there are constants $\beta, \lambda \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^{\beta}(n)}, n^{\alpha(n)}]_{/\lambda \log\log n}$ cannot be $(1/2 + \mathrm{poly}(n))$-approximated by $\mathrm{poly}(n)$-size $\mathscr{C}$ circuits.

We need the following lemma first.

Lemma 5.12 (Approximate linear sums preserve PRGs)**.** *Let $\mathscr{C}$ be a typical concrete circuit class, $s \in \mathbb{N}_{\geq 1}$, and $\delta \in (0,1)$. Let $\mathcal{H} \subseteq \mathcal{F}_{s,1}$ be the set of all functions that admit $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits with complexity at most $\sqrt{s}$. If G is a PRG for s-size $\mathscr{C}$ circuits with error s^{-1}, then G is also a PRG for $\mathcal{H}$ with error $1/\sqrt{s} + 2\delta$.*

Proof. Let $r \in \mathbb{N}$ be the seed length of G, and let $f \in \mathcal{H}$. From the definition of $\mathcal{H}$, there are $m \leq \sqrt{s}$ many $\mathscr{C}$ circuits $\{C_i\}_{i\in[m]}$, together with m coefficients $\{\alpha_i\}_{i\in[m]}$ such that for every $x \in \{0,1\}^s$, we have

$$\left| f(x) - \sum_{i\in[m]} \alpha_i \cdot C_i(x) \right| \leq \delta.$$

We also have $|C_i| \leq s$ for all $i \in [m]$ and $\sum_{i\in[m]} |\alpha_i| \leq \sqrt{s}$. We let $L(x) = \sum_{i\in[m]} \alpha_i \cdot C_i(x)$ for notational convenience.

Then, we show G is also a PRG fooling f with the desired error as follows:

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{z\in_R\{0,1\}^s}[f(z)] - \mathop{\mathbb{E}}_{r\in_R\{0,1\}^s}[f(G(r))]\right| \\
\leq &\left|\mathop{\mathbb{E}}_{z\in_R\{0,1\}^s}[L(z)] - \mathop{\mathbb{E}}_{r\in_R\{0,1\}^s}[L(G(r))]\right| + 2\delta && (\|f-L\|_\infty \leq \delta)\\
\leq &\sum_{i\in[m]} \alpha_i \left|\mathop{\mathbb{E}}_{z\in_R\{0,1\}^s}[C_i(z)] - \mathop{\mathbb{E}}_{r\in_R\{0,1\}^s}[C_i(G(r))]\right| + 2\delta && (\text{the definition of } L)\\
\leq &\sum_{i\in[m]} |\alpha_i| \cdot s^{-1} + 2\delta && (C_i \text{ has size at most } s)\\
\leq &1/\sqrt{s} + 2\delta. && (\textstyle\sum_{i\in[m]} |\alpha_i| \leq \sqrt{s})
\end{aligned}$$

Hence, G is a PRG fooling $\mathcal{H}$ with error $1/\sqrt{s} + 2\delta$ as well. ■

The following corollary follows from Lemma 5.12 immediately.

Corollary 5.13. *Let $\mathscr{C}$ be a typical concrete circuit class, $s \in \mathbb{N}_{\geq 1}$ and $\delta \in (0,1)$. Let $\mathcal{H} \subseteq \mathcal{F}_{s,1}$ be the set of all functions that admit $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuits with complexity at most $\sqrt{s}$. If $G = (G^{\mathsf{P}}, G^{\mathsf{W}})$ is an NPRG for s-size $\mathscr{C}$ circuits with error s^{-1}, then G is also an NPRG for $\mathcal{H}$ with error $1/\sqrt{s} + 2\delta$.*

We need the following lemma, a simple corollary of Lemma 2.15 and Corollary 5.13.

Lemma 5.14. *Let $\mathscr{C}$ be a typical concrete circuit class that can be simulated by* Formula. *There is a language $L \in \mathsf{P}$ such that one of the following holds:*

1. *For every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*
2. *If for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time, then the* derandomization condition *holds for* Formula.

Proof. We first assume that Item (1) does not hold, then by Lemma 2.15, every S-size formula admits a $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$ circuit of complexity at most S^γ for some $\gamma \in \mathbb{N}_{\geq 1}$, where $\delta = 0.01$.

From the assumed algorithm for $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits from Item (2) and Theorem 4.1, it follows that there is a constant $\varepsilon \in (0,1)$, size parameter $S(n) = 2^{\Omega(n^\varepsilon)}$, and an infinity often NPRG for $S(n)$-size $\mathscr{C}$ circuits with $S(n)^{-1}$-error, $2^{O(n^\varepsilon)}$-witness-length, $\mathrm{poly}(n)$-seed-length, and $2^{\mathrm{poly}(n)}$ running time.

Now, combining Corollary 5.13, the i.o. NPRG for $\mathscr{C}$, and the simulation of formulas by $\widetilde{\mathsf{Sum}}_\delta \circ \mathscr{C}$, it follows that there is another size parameter $S_0(n) = 2^{\Omega(n^\varepsilon)}$ and an infinitely often NPRG for $S_0(n)$-size $\mathscr{C}$ circuits with $2^{O(n^\varepsilon)}$-witness-length, $\mathrm{poly}(n)$-seed-length, and $2^{\mathrm{poly}(n)}$ running time. The second item then follows from Theorem 5.4. ■

Since $\mathsf{AC}^0_{d_v}[2] \circ$ Formula can be simulated by Formula, we have that the MA lower bound condition holds for Formula from Corollary 5.8. Hence, the following lemma follows immediately from Lemma 5.14 and Lemma 5.10.

Lemma 5.15. *Let $\mathscr{C}$ be a typical concrete circuit class that can be simulated by* Formula *and $\alpha(n)$ be a nice unbounded function. There is a language $L \in \mathsf{P}$ such that one of the following holds:*

- *For every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*

- *If for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in 2^{n-n^η} time, then there are constants $\beta, \lambda \in \mathbb{N}_{\geq 1}$ such that $(\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/\lambda \log\log n}$ cannot be $(1-n^{-\tau})$-approximated by $\mathrm{poly}(n)$-size formulas.*

Finally, we need to perform some mild-to-strong hardness amplification to prove Theorem 5.11. The following lemma follows from a careful analysis of Levin's proof of Yao's XOR Lemma [Lev87, GNW11].[40]

Lemma 5.16. *Let $\mathscr{C}$ be a typical concrete circuit class. There is a universal constant $c \geq 1$ such that, for every $n \in \mathbb{N}$, $f \in \mathcal{F}_{n,1}$, $\delta \in (0, 0.01)$, $k \in \mathbb{N}$, $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2}-\delta\right)$ and $\ell = c \cdot \frac{\log \delta^{-1}}{\varepsilon_k^2}$, if f cannot be $(1-5\delta)$-approximated by $\mathsf{MAJ}_\ell \circ \mathscr{C}$ circuits of size $s \cdot \ell + 1$, then $f^{\oplus k}$ cannot be $(\frac{1}{2}+\varepsilon_k)$-approximated by $\mathscr{C}$ circuits of size s.*[41]

Lemma 5.17. *Let $\mathscr{C}$ be a typical concrete circuit class, $\tau \in \mathbb{N}_{\geq 1}$, and $\alpha(n)$ be a nice unbounded function. For every $\beta \geq 2$ and every language $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/O(\log\log n)}$, there is a language $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/O(\log\log n)}$ such that, for two nondecreasing unbounded functions $S, \ell \colon \mathbb{N} \to \mathbb{N}$ such that $n \leq \ell(n) \leq 2^{o(n)}$, the following holds:*

- *If L cannot be $(1-n^{-\tau})$-approximated by $O(\ell(n)S(n))$-size $\mathsf{MAJ}_{\ell(n)} \circ \mathscr{C}$ circuits, then L' cannot be $(1/2 + \ell(n^{1/(\tau+3)})^{-1/3})$-approximated by $S(n^{1/(\tau+3)})$-size $\mathscr{C}$ circuits.*

Proof. We first define L' as follows: Given an input $x \in \{0,1\}^n$ for some $n \in \mathbb{N}$. Letting m be the largest integer such that $m^{\tau+2} \leq n$, and $k = \min(n - m^{\tau+2}, m^{\tau+1})$, we set $L'(x) = L_m^{\oplus k}(x_{\leq km})$, where $x_{\leq km}$ denotes the first km bits of x. Using the straightforward algorithm for computing L', it follows that $L' \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/O(\log\log n)}$.[42]

Now, there are infinitely many $n \in \mathbb{N}_{\geq 1}$ such that L_n cannot be $(1-n^{-\tau})$-approximated by $\ell(n)S(n)$-size $\mathsf{MAJ}_{\ell(n)} \circ \mathscr{C}$ circuits. We say that these n are good.

For every sufficiently large good $n \in \mathbb{N}_{\geq 1}$, we set $\delta = n^{-\tau}/5$ and $k = k(n)$ be the smallest integer so that $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2}-\delta\right) \geq \ell(n)^{1/3}$. Let c_1 be the universal constant in Lemma 5.16. Since n is sufficiently large and $\ell(n) \geq n$, $\ell_0 = c_1 \frac{\log \delta^{-1}}{\varepsilon_k^2} < \ell(n)$. Now, by Lemma 5.16 and the fact that L_n cannot be $(1-5\delta)$-approximated by $\ell_0 \cdot S(n) + 1 \leq \ell(n) \cdot S(n)$-size $\mathsf{MAJ}_{\ell_0} \circ \mathscr{C}$ circuits, it follows that $(L_n)^{\oplus k}$ cannot be $(1/2 + \ell(n)^{-1/3})$-approximated (note that $\varepsilon_k \leq \ell(n)^{-1/3}$ from our choice) by $S(n)$-size $\mathscr{C}$ circuits.

From our definition of L', it follows that for infinitely many $n \in \mathbb{N}_{\geq 1}$, $L'_{n^{\tau+2}+k(n)}$ (from our choice of k and the assumption that $\ell(n) = 2^{o(n)}$, we have that $k \leq n^{\tau+2}$) cannot be $(1/2 + \ell(n)^{-1/3})$-approximated by $S(n)$-size $\mathscr{C}$ circuits, which completes the proof since both S and ℓ are nondecreasing. ∎

Now, Theorem 5.11 follows immediately from Lemma 5.15 and Lemma 5.17.

[40] See Appendix B of the journal version of [Che19] for a proof, which can be found in http://www.mit.edu/~lijieche/Che19-journal-version.pdf.

[41] See Definition 4.4 for a formal definition of $f^{\oplus k}$.

[42] We remark that this step uses the facts that (1) L is in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/O(\log\log n)}$ instead of $\mathsf{NTG}[2^{\log^b n}, n^{\alpha(n)}]_{/O(\log\log n)}$ and (2) $\alpha(n)$ is non-decreasing.

5.5 Eliminating or Reducing the Advice

Finally, we apply the following lemma to get rid of or reduce the advice in Theorem 5.9 and Theorem 5.11. The same trick was used in [COS18] as well.

Lemma 5.18. *Let $\alpha(n)$ be a nice unbounded function. For every $\beta \geq 2$ and every language $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/O(\log\log n)}$, there are $\tau \in \mathbb{N}_{\geq 1}$ and languages $L_1 \in \mathsf{NTG}[2^{\log^\beta n}, n^{\alpha(n)}]$ and $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/1}$ such that the following holds:*

- *For every typical circuit class $\mathscr{C}$, $S\colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \to (0, 1/2)$ such that S, if L cannot be $1/2 + \varepsilon(n)$-approximated by $S(n)$-size $\mathscr{C}$ circuits, then neither L_1 nor L_2 can be $1/2 + \varepsilon(m(n))$-approximated by $S(m(n))$-size $\mathscr{C}$ circuits, where $m(n)$ is the largest integer such that $m \cdot 2^{\tau \cdot \log\log m} \leq n$.*

Proof. Let $\tau \in \mathbb{N}_{\geq 1}$ be the constant such that $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/\tau \log\log n}$.

For every $n \in \mathbb{N}_{\geq 1}$, let $\ell_n = \tau \log\log n$, and $\{w_i^{(n)}\}_{i \in [2^{\ell_n}]}$ be an enumeration of the set $\{0,1\}^{\ell_n}$. We will prove the lemma for L_1 and L_2 separately.

NTG lower bounds. We first prove the case for $L_1 \in \mathsf{NTG}[2^{\log^\beta n}, n^{\alpha(n)}]$. We define $L_1 \in \mathsf{NTG}[2^{\log^\beta n}, n^{\alpha(n)}]$ by the following algorithm A_1: on an input of length n, let m be the largest integer such that $m \cdot 2^{\ell_m} \leq n$, and $k = n - m \cdot 2^{\ell_m} + 1$; A_1 simulates the nondeterministic algorithm for L'_m with the advice w_k on the first m bits of the input. (If $k > 2^{\ell_m}$, A_1 simply outputs 0.)

Since L cannot be $1/2 + \varepsilon(n)$-approximated by $S(n)$-size $\mathscr{C}$ circuits, there are infinitely many pairs $(m_i, a_i) \in \mathbb{N} \times \left[2^{\ell_{m_i}}\right]$ such that the nondeterministic algorithm for L_{m_i} with advice w_{a_i} computes a function that cannot be $(1/2 + \varepsilon(m_i))$-approximated by $S(m_i)$-size $\mathscr{C}$ circuits.

By the construction of L_1, for infinitely many $(m_i, a_i) \in \mathbb{N} \times \left[2^{\ell_{m_i}}\right]$, $(L_1)_{n_i}$ cannot be $(1/2 + \varepsilon(m_i))$-approximated by $S(m_i)$-size $\mathscr{C}$ circuits, where $n_i = m_i \cdot 2^{\ell_{m_i}} + a_i - 1$.

$(\mathsf{N} \cap \mathsf{coN})\mathsf{TG}_{/1}$ lower bounds. Now we define $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/1}$ by the following algorithm A_2: on an input of length n, let m be the largest integer such that $m \cdot 2^{\ell_m} \leq n$, and $k = n - m \cdot 2^{\ell_m} + 1$; we set the advice bit $\alpha_n = 1$ if and only if $k \leq 2^{\ell_m}$ and w_k is the correct advice for input length m of language L; when $\alpha_n = 1$, A_2 simulates L_m with the advice w_k on the first m bits of the input; otherwise, A_2 simply outputs 0. A similar argument as that of the previous case completes the proof. ∎

Finally, applying Lemma 5.18, Theorem 5.1 and Theorem 5.2 follow immediately from Theorem 5.9 and Theorem 5.11, respectively.

We also note that the proof above for Theorem 5.2 indeed proves the following.

Remark 5.19. *Let $\mathscr{C}$ be a typical concrete circuit class that can be simulated by $\mathsf{Formula}$ and $\alpha(n)$ be a nice unbounded function. Suppose that for some $\eta \in (0,1)$, $\widetilde{\mathsf{CAPP}}$ of 2^{n^η}-size $\mathsf{AND}_4 \circ \mathscr{C} \circ \mathsf{AC}^0_2$ circuits can be deterministically solved in $2^{n - n^\eta}$ time. There is a language $L \in \mathsf{P}$ such that one of the following holds:*

- *For every $k \in \mathbb{N}_{\geq 1}$, L cannot be $(1/2 + n^{-k})$-approximated by n^k-size $\mathscr{C}$ circuits.*
- *There is $\beta \in \mathbb{N}_{\geq 1}$ such that neither*

$$\mathsf{NTG}[2^{\log^\beta n}, n^{\alpha(n)}] \text{ nor } (\mathsf{N} \cap \mathsf{coN})\mathsf{TG}[2^{\log^\beta n}, n^{\alpha(n)}]_{/1}$$

can be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size formulas.

5.6 Proof of Theorem 1.4

Finally, we apply our generic connection Theorem 5.2 to prove Theorem 1.4.

Reminder of Theorem 1.4. *Let α be a nice unbounded function. There is a $\beta \in \mathbb{N}$ such that $\mathsf{NTIMEGUESS}[2^{\log^\beta n}, n^{\alpha(n)}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size ACC^0 circuits.*

Recall that [Wil14, Wil18b] proved that for every $d, m \in \mathbb{N}_{\geq 1}$, there is an $\varepsilon \in (0,1)$ (that depends on d, m) such that $\widetilde{\mathsf{CAPP}}$ for $\mathsf{AC}^0_d[m]$ circuits can be solved in 2^{n-n^ε} time. Combining this algorithm with Theorem 5.2, we immediately obtain the following weaker version of Theorem 1.4.

Corollary 5.20. *Let α be a nice unbounded function. For every $d, m \in \mathbb{N}_{\geq 1}$, there is a $\beta \in \mathbb{N}$ such that $\mathsf{NTIMEGUESS}[2^{\log^\beta n}, n^{\alpha(n)}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathsf{AC}^0_d[m]$.*

We note that Theorem 1.4 is stronger than the corollary above since it asserts the existence of a single constant $\beta \in \mathbb{N}_{\geq 1}$ such that $\mathsf{NTIMEGUESS}[2^{\log^\beta n}, n^{\alpha(n)}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size $\mathsf{AC}^0_d[m]$, for every $d, m \in \mathbb{N}_{\geq 1}$.

Now we prove Theorem 1.4 by a careful win-win argument, similar to [CR21].

Proof of Theorem 1.4. Let $L \in \mathsf{P}$ be the language from Lemma 2.15. If L cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size ACC^0 circuits, then we are done. Otherwise, there are constants $d_\star, m_\star, k \in \mathbb{N}_{\geq 1}$ such that L can be $(1/2 + n^{-k})$-approximated by n^k-size $\mathsf{AC}^0_{d_\star}[m_\star]$ circuits. Applying Remark 5.19 and the $\widetilde{\mathsf{CAPP}}$ algorithm for $\mathsf{AC}^0_{d_\star}[m_\star]$, there is a $\beta \in \mathbb{N}$ such that $\mathsf{NTIMEGUESS}[2^{\log^\beta n}, n^{\alpha(n)}]$ cannot be $1/2 + 1/\mathrm{poly}(n)$-approximated by $\mathrm{poly}(n)$-size formulas, which also implies the theorem (since ACC^0 can be simulated by $\mathsf{Formula}$). ∎

6 Circuit Lower Bounds for $(\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0[2] \circ \mathscr{C}}$ against $\mathscr{C}$

In this section, we prove Theorem 5.3; see Theorem 6.3 for a formal version.

6.1 Technical Ingredients

We begin with some technical ingredients.

Theorem 6.1. *Let $s \colon \mathbb{N} \to \mathbb{N}$ be a space-constructible function such that $s(n) \leq 2^{o(n)}$ and $s(n) \geq n$ for every $n \in \mathbb{N}_{\geq 1}$. There is a universal constant $c \in \mathbb{N}_{\geq 1}$ and a language $L \in \mathsf{SPACE}[s(n)^c]$ such that $\mathsf{heur}_{0.99}\text{-}\mathsf{SIZE}(L_n) > s(n)$ for all sufficiently large n.*

Proof. In the following, we always assume that n is large enough. Let $c_1 \in \mathbb{N}_{\geq 1}$ be a large enough constant and let $\ell = c_1 \log s(n)$. There are $2^{2^\ell} = 2^{s(n)^{c_1}}$ many distinct functions in $\mathcal{F}_{\ell,1}$. Also, there are at most $2^{s(n)^2}$ many ℓ-input $s(n)$-size circuits. We claim that there exists a function $f \in \mathcal{F}_{\ell,1}$ that cannot be 0.99-approximated by $s(n)$-size circuits.

To see the claim. Fix an ℓ-input $s(n)$-size circuit C. We draw a random function $f \in \mathcal{F}_{\ell,1}$. By a Chernoff bound, C 0.99-approximates f with probability at most $2^{-\Omega(2^\ell)} \leq 2^{-\Omega(s(n)^{c_1})} \leq 2^{-s(n)^3}$, the last inequality follows from the fact that c_1 and n are large enough. Our claim then follows from a union bound over all $2^{s(n)^2}$ many ℓ-input $s(n)$-size circuits.

Now, letting $c = 2c_1$, our algorithm computing L first enumerates all ℓ-bit functions to find the lexicographically first $f_0 \in \mathcal{F}_{\ell,1}$ that cannot be 0.99-approximated by all $s(n)$-size circuits. Note

that by our claim above, such f_0 exists for all sufficiently large n. Then our algorithm computes f_0 on the first ℓ bits of the input and ignores the rest. (Note that here we use the fact that $\ell \leq O(\log s(n)) \leq o(n)$.) This algorithm can be implemented in $s(n)^c$ space in a straightforward way, and the average-case hardness for L follows from our construction of f_0. ■

The concrete circuit class $\widetilde{\mathsf{AC}}^0_d[2]$. For $d \in \mathbb{N}_{\geq 1}$, we define $\widetilde{\mathsf{AC}}^0_d[2]$ as a sub-class of $\mathsf{AC}^0_d[2]$ such that the top-gate must be an XOR gate and the second layer (counting from the top layer) must all be AND gates. An S-size $\widetilde{\mathsf{AC}}^0_d[2]$ circuit is by definition also an S-size $\mathsf{AC}^0_d[2]$ circuit and an S-size $\mathsf{AC}^0_d[2]$ circuit has an equivalent $O(S)$-size $\widetilde{\mathsf{AC}}^0_{d+2}[2]$ circuit by adding two dummy layers at the top. (Note that for our proofs, we do not need $\widetilde{\mathsf{AC}}^0_d[2]$ to be typical.)

We work with $\widetilde{\mathsf{AC}}^0_d[2]$ instead of $\mathsf{AC}^0_d[2]$ because of the following lemma.

Lemma 6.2. *Let $\mathscr{C}$ be a typical concrete circuit class. There are two universal constants $d_0, c_0 \in \mathbb{N}$ such that the following holds. For every $d_\star \in \mathbb{N}$ such that $d_\star \geq d_0$, letting $\mathscr{D} = \widetilde{\mathsf{AC}}^0_{d_\star}[2] \circ \mathscr{C}$, for all sufficiently large $n \in \mathbb{N}_{\geq 1}$, we have*

$$\mathscr{D}\text{-}\mathsf{SIZE}(L^{\mathsf{PSPACE}}_n) \leq \left(\mathscr{D}\text{-}\mathsf{SIZE}(L^{\mathsf{PSPACE}}_{n-1}) + n^{c_0}\right)^{c_0},$$

where L^{PSPACE} is the PSPACE*-complete language from Theorem 3.15.*

Proof. Let DSR and Aux be the algorithms from Theorem 3.15. We set $d_0 \in \mathbb{N}$ so that Aux_n can be implemented by $\mathrm{poly}(n)$-size $\widetilde{\mathsf{AC}}^0_{d_0}[2]$ circuits. The lemma then follows from the properties of DSR and Aux, and the observation that a size-S $(\mathsf{XOR} \circ \mathsf{AND}_3) \circ \widetilde{\mathsf{AC}}_{d_0}[m_\star]$ circuit can be converted into an $\widetilde{\mathsf{AC}}_{d_0}[m_\star]$ circuit of size $\mathrm{poly}(S)$, since an $\mathsf{AND}_3 \circ \mathsf{XOR}_t$ circuit can be converted into an $\mathsf{XOR}_{O(t^3)} \circ \mathsf{AND}_3$ circuit. ■

6.2 Lower Bounds for $(\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0[2] \circ \mathscr{C}}$ against $\mathscr{C}$ Circuits

We will prove the following theorem, from which Theorem 5.3 follows immediately.

Theorem 6.3 (Formal version of Theorem 5.3)**.** *Let $\mathscr{C}$ be a typical concrete circuit class. There are universal constants $d_{\mathsf{v}}, \tau \in \mathbb{N}_{\geq 1}$ such that the following holds. For all $a \in \mathbb{N}_{\geq 1}$, there is a constant $c \in \mathbb{N}_{\geq 1}$ and a language*

$$L \in \left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_{\mathsf{v}}}[2] \circ \mathscr{C}}\right)_{/1}$$

such that the following hold:

1. *For all sufficiently large $\tau \in \mathbb{N}_{\geq 1}$ and $n = 2^\tau$, either*
   - $\mathsf{heur}_{(1-n^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) > n^a$, *or*
   - $\mathsf{heur}_{(1-m^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_m) > m^a$, *for some $m \in (n^c, 2 \cdot n^c) \cap \mathbb{N}$.*
2. *For every $n \in \mathbb{N}_{\geq 1}$, if the advice for L on n-bit inputs is 0, then L_n is the all-zero function.*

Proof of Theorem 6.3. Let L^{PSPACE} be the language from Theorem 3.15 and $\beta \in \mathbb{N}_{\geq 1}$ be the constant such that for every $n, s \in \mathbb{N}_{\geq 1}$ such that $s \geq n$, an s-size $\mathscr{C}$ circuit on n-bit inputs has an equivalent s^β-size general fan-in 2 Boolean circuit (such β exists since $\mathscr{C}$ is a typical concrete circuit class).

Applying Theorem 6.1 with size function $S(n) = n^{\beta \cdot a}$, there is a language $L^{\mathsf{diag}} \in \mathsf{PSPACE}$ such that $\mathsf{heur}_{0.99}\text{-}\mathsf{SIZE}(L_n^{\mathsf{diag}}) > n^{\beta \cdot a}$ for all sufficiently large $n \in \mathbb{N}$. Since L^{PSPACE} is PSPACE-complete and paddable, there is $c_1 \in \mathbb{N}$ such that L_n^{diag} can be reduced to L^{PSPACE} on input length n^{c_1} in $O(n^{c_1})$ time. Let d_0, c_0 be the constants from Lemma 6.2.

Let d_{wc2ac} be such that L^{PSPACE} is $\mathsf{AC}^0_{d_{\mathsf{wc2ac}}}[2]$ weakly error correctable, and let τ be the corresponding constant τ_{wc2ac}. We then set $c = c_1$ and also let $d_{\mathsf{g}} = d_0 + d_{\mathsf{wc2ac}} + 2$. We then set $\mathscr{D} = \widetilde{\mathsf{AC}}^0_{d_{\mathsf{g}}}[2] \circ \mathscr{C}$. Let $b \in \mathbb{N}_{\geq 1}$ be a sufficiently large constant to be chosen later.

The algorithm. Let $\tau \in \mathbb{N}_{\geq 1}$ be sufficiently large, $n = 2^\tau$, and $m = n^c$. We first provide an informal description of the $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm A_L that computes the hard language L. There are two cases:

1. When $\mathscr{D}\text{-}\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$. That is, when L_m^{PSPACE} is *easy*. In this case, on inputs of length n, we guess-and-verify a $\mathscr{D}$ circuit for L_m^{PSPACE} of size at most n^b, and use that to compute L_n^{diag}.

2. Otherwise, we know that L_m^{PSPACE} is *hard*. Let ℓ be the largest integer such that

$$\mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n^b.$$

   On inputs of length $m_1 = m + \ell$, we guess-and-verify a $\mathscr{D}$ circuit for L_ℓ^{PSPACE}, and compute L_ℓ^{PSPACE} on the first ℓ input bits. Note that by Corollary 3.16, we have $0 < \ell < m$ and therefore $m + \ell$ is not a power of 2.

Intuitively, A_L computes a hard function because either it computes the hard language L_n^{diag} on inputs of length n, or it computes the hard language L_ℓ^{PSPACE} on inputs of length m. A formal description of A_L is given in Algorithm 6.1, and the algorithm A_{adv} for setting the advice bits of A_L is given in Algorithm 6.2. Since $m + \ell$ at Line 9 is never a power of 2, α_n can only be set once in Algorithm 6.2.

Now we verify that A_L computes a language satisfying our requirements.

A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise. We first show that A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise (see Definition 3.5). The intuition is that it only tries to guess-and-verify a circuit for L^{PSPACE} when it exists, and the properties of the instance checker (see Definition 3.12) ensure that in this case A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise.

Formally, there are three cases:

1. $\alpha_n = 0$. In this case, A_L computes the all-zero function and satisfies the promise.

2. $\alpha_n = 1$ and n is a power of 2. In this case, from Algorithm 6.2, we have $\mathscr{D}\text{-}\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$ for $m = n^c$. Therefore, at least one guess of the circuit C is the correct circuit for L_m^{PSPACE}, and on that guess, A_L outputs $L_m^{\mathsf{PSPACE}}(z) = L_n^{\mathsf{diag}}(x)$ with probability 1, by the property of the instance checker (see Definition 3.12). Again by the property of the instance checker, on all guesses of C, A_L outputs $1 - L_m^{\mathsf{PSPACE}}(z) = 1 - L_n^{\mathsf{diag}}(x)$ with probability at most $1/3$.

3. $\alpha_n = 1$ and n is not a power of 2. In this case, from Algorithm 6.2, we have $\mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n_0^b$. Therefore, at least one guess of the circuit C is the correct circuit for L_ℓ^{PSPACE}, and on that guess, A_L outputs $L_\ell^{\mathsf{PSPACE}}(z)$ with probability 1, again by the property of the instance checker. Similar to the previous case, on all possible guesses of C, A_L outputs $1 - L_\ell^{\mathsf{PSPACE}}(z)$ with probability at most $1/3$.

Algorithm 6.1: The $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm A_L

Input: $x \in \{0,1\}^n$
Advice: $\alpha = \alpha_n \in \{0,1\}$
if $\alpha = 0$ **then**
  return 0
Let $m = n^c$;
Let $n_0 = n_0(n)$ be the largest integer such that $n_0^c \leq n$;
Let $m_0 = n_0^c$;
Let $\ell = n - m_0$;
if *n is a power of 2* **then**
  `/* We are in the case that` $\mathscr{D}$-$\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$. `*/`
  Compute $z \in \{0,1\}^m$ in $O(m)$ time such that $L_n^{\mathsf{diag}}(x) = L_m^{\mathsf{PSPACE}}(z)$;
  Guess a $\mathscr{D}$ circuit C of size at most n^b;
  Compute in $\mathrm{poly}(m) \leq \mathrm{poly}(n)$ time a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit IC_m that implements the instance checker for L_m^{PSPACE};
  Let $\mathsf{rnd}_m \leq \mathrm{poly}(m)$ be the number of random coins used by IC_m, draw $r \in_{\mathsf{R}} \{0,1\}^{\mathsf{rnd}_m}$;
  return $\mathsf{IC}_m^C(z,r)$;
else
  `/* We are in the case that` $\mathscr{D}$-$\mathsf{SIZE}(L_{m_0}^{\mathsf{PSPACE}}) > n_0^b$ `and` ℓ `is the largest integer such that` $\mathscr{D}$-$\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n_0^b$. `*/`
  Let z be the first ℓ bits of x;
  Guess a $\mathscr{D}$ circuit C of size at most n_0^b;
  Compute in $\mathrm{poly}(\ell) \leq \mathrm{poly}(n)$ time a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit IC_ℓ that implements the instance checker for L_ℓ^{PSPACE};
  Let $\mathsf{rnd}_\ell \leq \mathrm{poly}(\ell)$ be the number of random coins used by IC_ℓ, draw $r \in_{\mathsf{R}} \{0,1\}^{\mathsf{rnd}_\ell}$;
  return $\mathsf{IC}_\ell^C(z,r)$;

Algorithm 6.2: The algorithm A_{adv} for setting advice bits in Algorithm 6.1

All the α_n are set to 0 by default;
for $\tau = 1 \to \infty$ **do**
  Let $n = 2^\tau$;
  Let $m = n^c$;
  if $\mathscr{D}$-$\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$ **then**
    Set $\alpha_n = 1$;
  else
    Let $\ell = \max\{\ell : \mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n^b\}$;
    Set $\alpha_{m+\ell} = 1$;

The following claim summarizes what we have.

Claim 6.4. *A_L with advice set by A_{adv} is an $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm for a language L such that, for every $n \in \mathbb{N}_{\geq 1}$, L_n is defined as below:*

1. *If $\alpha_n = 0$, then L_n is the all-zero function.*
2. *If $\alpha_n = 1$ and n is a power of 2, then L_n is the same function as L_n^{diag}.*
3. *If $\alpha_n = 1$ and n is not a power of 2, then L_n is the n-bit function that computes L_ℓ^{PSPACE} on the first ℓ bits and ignores the rest of the input.*

Now, Item (2) of the theorem follows directly from Item (1) of Claim 6.4.

The verifier complexity of A_L. We also need to show that A_L is an $\left((\mathsf{MA} \cap \mathsf{coMA})_{\mathsf{AC}^0_{d_\star + d_{\mathsf{v}}}[m_\star]}\right)_{/1}$ algorithm. Let $d_{\mathsf{ic}} \in \mathbb{N}_{\geq 1}$ be such that the instance checker for L^{PSPACE} can be implemented by non-adaptive $\widetilde{\mathsf{AC}}^0_{d_{\mathsf{ic}}}[2]$ circuits. Then, we can see that for every possible guess C in Algorithm 6.1, $\mathsf{IC}^C_m(z,\cdot)$ (resp. $\mathsf{IC}^C_\ell(z,\cdot)$) is a polynomial-size $\widetilde{\mathsf{AC}}^0_{d_{\mathsf{g}}+d_{\mathsf{ic}}}[2] \circ \mathscr{C}$ circuit. We now set $d_{\mathsf{v}} = d_{\mathsf{g}} + d_{\mathsf{ic}}$.

A_L computes a hard language. Finally, we show that the algorithm A_L indeed computes a hard language as stated. Let τ be a sufficiently large integer, $n = 2^\tau$, and $m = n^c$. There are two cases:

1. $\mathscr{D}\text{-}\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$. In this case, we have $\alpha_n = 1$ by Algorithm 6.2. By Item (2) of Claim 6.4, we have that L_n is the same function as L_n^{diag}, and therefore $\mathsf{heur}_{0.99}\text{-}\mathsf{SIZE}(L_n) > n^{\beta \cdot a}$, which implies $\mathsf{heur}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) > n^a$, by the definition of β.
2. $\mathscr{D}\text{-}\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) > n^b$. Let ℓ be the largest integer such that $\mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n^b$. By Corollary 3.16, we have $0 < \ell < m$.

   Note that $\mathscr{D}\text{-}\mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}}) \leq ((\ell+1)^{c_0} + \mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}))^{c_0}$ from Lemma 6.2. Therefore,

   $$\mathscr{D}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \geq \left(\mathscr{D}\text{-}\mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}})\right)^{1/c_0} - (\ell+1)^{c_0} \geq n^{b/c_0} - m^{c_0} = n^{b/c_0} - n^{c \cdot c_0}.$$

   Furthermore, recall that $\mathscr{D} = \widetilde{\mathsf{AC}}^0_{d_0 + d_{\mathsf{wc2ac}}+2}[2]$ and L^{PSPACE} is $\mathsf{AC}^0_{d_{\mathsf{wc2ac}}}[2]$ weakly error correctable with constant $\tau_{\mathsf{wc2ac}} = \tau$.[43] We also have

   $$\mathsf{heur}_{(1-\ell^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \geq \left[n^{b/c_0} - n^{c \cdot c_0}\right] / O(\ell^\tau).$$

   Now, on inputs of length $m_1 = m + \ell$, we have $\alpha_{m_1} = 1$ by Algorithm 6.2 (note that $m_1 \in (m, 2m)$ as $\ell \in (0, m)$). We set b large enough so that

   $$\left[n^{b/c_0} - n^{c \cdot c_0}\right] / O(\ell^\tau) \geq (2n^c)^{a+1} = (2m)^{a+1} \geq m_1^{a+1},$$

   hence we also have $\mathsf{heur}_{(1-\ell^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) > m_1^a$.

   Then by Item (3) of Claim 6.4, we have that L_{m_1} is the m_1-input function that computes L_ℓ^{PSPACE} on the first ℓ bits and ignores the last m input bits. Hence, we have

   $$\mathsf{heur}_{(1-m_1^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_{m_1}) \geq \mathsf{heur}_{(1-\ell^{-\tau})}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_{m_1}) > m_1^a,$$

   which completes the proof.

■

[43] The $+2$ in the depth of $\mathscr{D}$ corresponds to the fact that $\mathsf{AC}^0_{d_{\mathsf{wc2ac}}}[2]$ can be simulated by $\widetilde{\mathsf{AC}}^0_{d_{\mathsf{wc2ac}}+2}[2]$ with a linear blow-up in size.

7 A PSPACE-complete Language with $\mathsf{AC}^0[2]$ Reducibility Properties

In this section, we prove Theorem 3.15, which is restated below.

Reminder of Theorem 3.15. *There is a* PSPACE*-complete language* L^{PSPACE} *that is paddable, non-adaptive* $\mathsf{AC}^0[2]$ *downward self-reducible, non-adaptive* $\mathsf{AC}^0[2]$ *same-length checkable and non-adaptive* $\mathsf{AC}^0[2]$ *weakly error correctable.*

Moreover, There are two algorithms DSR *and* Aux *satisfying the following:*

1. Aux *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and outputs a value from* $\{0,1\}$.
2. Aux *can be implemented by a uniform* $\mathsf{AC}^0[2]$ *circuit.*
3. DSR *takes* $n \in \mathbb{N}_{\geq 1}$ *as input parameter and* $\vec{x} \in \{0,1\}^n$ *as input, and functions* $h_1 \colon \{0,1\}^{n-1} \to \{0,1\}$ *and* $h_2 \colon \{0,1\}^n \to \{0,1\}$ *as oracles.*
4. *For every* $n \in \mathbb{N}_{\geq 1}$, $\mathsf{DSR}_n^{L^{\mathsf{PSPACE}}_{n-1},\mathsf{Aux}_n}$ *computes* L^{PSPACE}_n.
5. DSR *can be implemented by a uniform non-adaptive* $\mathsf{XOR} \circ \mathsf{AND}_3$ *oracle circuit family. In more detail,* DSR *first queries its oracles on some* projections *of the input* $\vec{x}$ *to obtain some intermediate values and then applies an* $\mathsf{XOR} \circ \mathsf{AND}_3$ *circuit on those intermediate values and the input* $\vec{x}$ *to obtain the output.*

7.1 Preliminaries

To avoid confusion, we often use bold letters (*e.g.*, $\mathbf{x}$ and $\mathbf{y}$) to emphasize that they are formal variables.

7.1.1 Finite Fields

Throughout this section, we will only consider finite fields of the form $\mathsf{GF}(2^{2\cdot 3^\ell})$ for some $\ell \in \mathbb{N}$, since they enjoy simple representations that will be useful for us. For every $\ell \in \mathbb{N}$, we set $\mathsf{pw}_\ell = 2 \cdot 3^\ell$ and use $\mathbb{F}^{(\ell)}$ to denote $\mathsf{GF}(2^{\mathsf{pw}_\ell})$.

Let $\ell \in \mathbb{N}$. We will always represent $\mathbb{F}^{(\ell)} = \mathsf{GF}(2^{\mathsf{pw}_\ell})$ as $\mathbb{F}_2[\mathbf{x}]/(\mathbf{x}^{\mathsf{pw}_\ell} + \mathbf{x}^{\mathsf{pw}_\ell/2} + 1)$.[44] That is, we identify an element of $\mathsf{GF}(2^{\mathsf{pw}_\ell})$ with an $\mathbb{F}_2[\mathbf{x}]$ polynomial with degree less than pw_ℓ. To avoid confusion, given a polynomial $P(\mathbf{x}) \in \mathbb{F}_2[\mathbf{x}]$ with degree less than pw_ℓ, we will use $(P(\mathbf{x}))_{\mathbb{F}^{(\ell)}}$ to denote the unique element in $\mathbb{F}^{(\ell)}$ identified with $P(\mathbf{x})$.

Let $\kappa^{(\ell)}$ be the natural bijection between $\{0,1\}^{\mathsf{pw}_\ell}$ and $\mathbb{F}^{(\ell)} = \mathsf{GF}(2^{\mathsf{pw}_\ell})$: for every $a \in \{0,1\}^{\mathsf{pw}_\ell}$, $\kappa^{(\ell)}(a) = \left(\sum_{i\in[\mathsf{pw}_\ell]} a_i \cdot \mathbf{x}^{i-1}\right)_{\mathbb{F}^{(\ell)}}$. We always use $\kappa^{(\ell)}$ to encode elements from $\mathbb{F}^{(\ell)}$ by Boolean strings. That is, whenever we say that an algorithm takes an input from $\mathbb{F}^{(\ell)}$, we mean it takes a string $x \in \{0,1\}^{\mathsf{pw}_\ell}$ and interprets it as an element of $\mathbb{F}^{(\ell)}$ via $\kappa^{(\ell)}$. Similarly, whenever we say that an algorithm outputs an element from $\mathbb{F}^{(\ell)}$, we mean it outputs a string $\{0,1\}^{\mathsf{pw}_\ell}$ encoding that element via $\kappa^{(\ell)}$. For simplicity, sometimes we use $(a)_{\mathbb{F}^{(\ell)}}$ to denote $\kappa^{(\ell)}(a)$. Also, we order elements from $\mathbb{F}^{(\ell)}$ by the lexicographical ordering of their corresponding Boolean strings from $\{0,1\}^{\mathsf{pw}_\ell}$ (*i.e.*, we use i-th element in $\mathbb{F}^{(\ell)}$ to denote the element in $\mathbb{F}^{(\ell)}$ encoded by the i-th lexicographically smallest Boolean string in $\{0,1\}^{\mathsf{pw}_\ell}$).

Finally, for each $n \in \mathbb{N}$, we set ℓ_n to be the smallest integer such that $\mathsf{pw}_{\ell_n} \geq n$. We also let $\mathsf{sz}_n = \mathsf{pw}_{\ell_n} = 2 \cdot 3^{\ell_n}$, $\mathbb{F}_n = \mathbb{F}^{(\ell_n)} = \mathsf{GF}(2^{\mathsf{sz}_n})$, and $\kappa_n = \kappa^{(\ell_n)}$. Note that $2^n \leq |\mathbb{F}_n| \leq 2^{3n}$.

[44] $\mathbf{x}^{2\cdot 3^\ell} + \mathbf{x}^{3^\ell} + 1 \in \mathbb{F}_2[x]$ is irreducible, see [VL99, Theorem 1.1.28].

7.1.2 Uniform $\mathsf{AC}^0[2]$ Circuits for Arithmetic Operations over $\mathbb{F}_n$

We will need the following uniform $\mathsf{AC}^0[2]$ circuits for arithmetic operations over $\mathbb{F}_n$ in [HAB02, HV06].

Lemma 7.1 ([HAB02, HV06]). *Let $n, t \in \mathbb{N}$ be input parameters. There are uniform $\mathsf{AC}^0[2]$ circuits for the following two tasks:*

1. Iterated addition*: given a list $a_1, \ldots, a_t \in \mathbb{F}_n$, compute $\sum_{i \in [t]} a_i$.*
2. Iterated multiplication*: given a list $a_1, \ldots, a_t \in \mathbb{F}_n$ such that $t \leq \log n$, compute $\prod_{i \in [t]} a_i$.*

Applying Lemma 7.1, we will give two algorithms for polynomial interpolation. Let $\alpha_1, \ldots, \alpha_t$ be the first t non-zero elements from $\mathbb{F}_n$ and $\beta_1, \ldots, \beta_t \in \mathbb{F}_n$ be the input. Let p be the unique degree-$(t-1)$ polynomial such that $p(\alpha_i) = \beta_i$ for every $i \in [t]$. The first algorithm allows us to compute $p(x)$ for any $x \in \mathbb{F}_n$ (given as an additional input) in uniform $\mathsf{AC}^0[2]$, but only works for $t \leq \log n$. The second algorithm only allows us to compute $p(0)$ in uniform $\mathsf{AC}^0[2]$, but works for any t. Jumping ahead, the first algorithm will be useful in our construction of instance checkers, and the second algorithm will be useful for establishing weakly error correctability.

Corollary 7.2 (Interpolation in $\mathsf{AC}^0[2]$). *There are two algorithms D^{intp} and D^{intp0} satisfying the following:*

1. *(**Input**) D^{intp} and D^{intp0} both take $n, t \in \mathbb{N}$ as input parameters, a list $\beta_1, \ldots, \beta_t \in \mathbb{F}_n$ as input, and outputs an element from $\mathbb{F}_n$. For D^{intp}, we also require that $t \leq \log n$ (i.e., D^{intp} aborts immediately if $t > \log n$) and it takes an additional $x \in \mathbb{F}_n$ as input.*
2. *(**Output**) Let $p(\mathbf{x}) \colon \mathbb{F}_n \to \mathbb{F}_n$ be the unique polynomial with degree at most $t-1$ such that $p(\alpha_i) = \beta_i$ for every $i \in [t]$, where α_i is the i-th non-zero element in $\mathbb{F}_n$. D^{intp} outputs $p(x)$ and D^{intp0} outputs $p(0)$.*
3. *(**Complexity**) Both of D^{intp} and D^{intp0} can be implemented by uniform $\mathsf{AC}^0[2]$ circuit families.*

Proof. For every $i \in [t]$, we define a polynomial $e_i(\mathbf{x}) \colon \mathbb{F}_n \to \mathbb{F}_n$ as follows:

$$e_i(\mathbf{x}) = \prod_{j \in [t] \setminus \{i\}} \frac{\mathbf{x} - \alpha_j}{\alpha_i - \alpha_j}.$$

We have that $p(\mathbf{x}) = \sum_{i \in [t]} e_i(\mathbf{x}) \cdot \beta_i$. Applying Lemma 7.1, $p(x)$ can be computed by a uniform $\mathsf{AC}^0[2]$ circuit given x and $\{\beta_i\}_{i \in [t]}$ as input (note that the α_i are constants) when $t \leq \log n$. Also, $p(0)$ can be computed by a uniform $\mathsf{AC}^0[2]$ circuit given $\{\beta_i\}_{i \in [t]}$ as input (since all of the $e_i(0)$ can be precomputed in polynomial time). ∎

7.2 An Adaption of the Construction from [TV07]

We need the following lemma, which builds on the proof of $\mathsf{IP} = \mathsf{PSPACE}$ theorem [LFKN92, Sha92]. The proof follows similar ideas from the proof of [TV07, Lemma 4.1] but requires several modifications.

Lemma 7.3. *There is a collection of polynomials $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n \in \mathbb{N}_{\geq 1},\, i \in [n]}$ with the following properties:*

1. *(**Term polynomial**) There is an algorithm* Term *satisfying the following:*
   (a) Term *takes* $n, i \in \mathbb{N}_{\geq 1}$ *such that* $i \in [n]$ *as input parameters, and* $\vec{x} \in \mathbb{F}_n$ *as input, and output an element from* $\mathbb{F}_n$.
   (b) Term *can be implemented by a uniform* $\mathsf{AC}^0[2]$ *circuit family.*
2. *(**Self-reducibility**) There is an algorithm* Red *satisfying the following:*
   (a) Red *takes* $n, i \in \mathbb{N}_{\geq 1}$ *such that* $i < n$ *as input parameters, and* $\vec{x} \in \mathbb{F}_n^n$ *as input, and functions* $h_1, h_2 \colon \mathbb{F}_n^n \to \mathbb{F}_n$ *as oracles.*
   (b) $\mathsf{Red}_{n,i}^{f_{n,i+1},\mathsf{Term}_{n,i}}$ *computes* $f_{n,i}$.
   (c) Red *can be implemented by a uniform non-adaptive* $\mathsf{XOR} \circ \mathsf{AND}_2$ *oracle circuit family. In more detail,* Red *first queries its oracle on some* projections *of the input* $\vec{x}$ *to obtain some intermediate values and then applies an* $\mathsf{XOR} \circ \mathsf{AND}_2$ *circuit on those intermediate values and the input* $\vec{x}$ *to obtain the output.*
3. *(**Base case**) For every* $n \in \mathbb{N}_{\geq 1}$, $f_{n,n}$ *is the constant-one polynomial.*
4. *(*PSPACE*-**hardness**) For every* $L \in \mathsf{PSPACE}$, *there is a pair of algorithm* $(A_L^{\mathsf{len}}, A_L^{\mathsf{red}})$ *satisfying the following:*
   (a) A_L^{len} *takes* $n \in \mathbb{N}_{\geq 1}$ *as input and outputs an integer in* $\mathrm{poly}(n)$ *time;* A_L^{red} *takes* $x \in \{0,1\}^*$ *as input, and outputs a vector* $\vec{z} \in \mathbb{F}_m^m$ *for* $m = A_L^{\mathsf{len}}(|x|)$.
   (b) *For every* $n \in \mathbb{N}_{\geq 1}$, $A_L^{\mathsf{len}}(n) \leq c_L \cdot n^{c_L}$ *for some constant* $c_L \in \mathbb{N}_{\geq 1}$ *that depends on* L, *and for every* $x \in \{0,1\}^n$, *it holds that* $L(x) = f_{m,1}(\vec{z})$, *where* $m = A_L^{\mathsf{len}}(|x|)$ *and* $\vec{z} = A_L^{\mathsf{red}}(x)$.[45]
5. *(**Low degree**) For every* $n \in \mathbb{N}_{\geq 1}$ *and* $i \in [n]$, $f_{n,i}$ *has individual degree at most* c_{deg}, *where* c_{deg} *is a universal constant.*
6. *(**Instance checker**) There is a randomized algorithm* IC *that takes* $n, i \in \mathbb{N}_{\geq 1}$ *such that* $i \in [n]$ *as input parameters, and* $\vec{x} \in \mathbb{F}_n$ *as input, and* $n - i + 1$ *functions* $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ *as oracles, and outputs an element in* $\mathbb{F}_n \cup \{\perp\}$. *The following properties hold for* IC:
   (a) *If* $\tilde{f}_j = f_{n,j}$ *for every* $j \in \{i, \ldots, n\}$, *then* $\mathsf{IC}_{n,i}^{\tilde{f}_i,\ldots,\tilde{f}_n}(\vec{x})$ *outputs* $f_{n,i}(\vec{x})$ *with probability* 1 *for every* $\vec{x} \in \mathbb{F}_n^n$.
   (b) *For every* $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ *and every* $\vec{x} \in \mathbb{F}_n^n$, $\mathsf{IC}_{n,i}^{\tilde{f}_i,\ldots,\tilde{f}_n}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$ *with probability* $1 - 1/2^n$, *over the internal randomness of* IC.
   (c) IC *can be implemented by a randomized uniform non-adaptive* $\mathsf{AC}^0[2]$ *circuit family.*

We prove Lemma 7.3 together with some additional properties of $\mathscr{F}^{\mathsf{TV}}$ below.

7.3 A PSPACE-complete Problem TQBF$^{\mathsf{u}}$

We first introduce a variant of the standard PSPACE-complete problem TQBF (True Quantified Boolean Formula), which we call TQBF$^{\mathsf{u}}$. For $n \in \mathbb{N}_{\geq 1}$, we let $\phi_n^{\mathsf{cl\text{-}idx}}$ be a bijection from $[8 \cdot \binom{n}{3}]$ to $\binom{[n]}{3} \times \{0,1\}^3$. Here, for a set S and an integer $m \in \mathbb{N}$, we use $\binom{S}{m}$ to denote the set of all size-m subsets of S.

[45] *i.e.*, $f_{m,1}(\vec{z}) = (L(x))_{\mathbb{F}_m}$.

Definition 7.4. *The* $\mathsf{TQBF}^{\mathsf{u}}$ *problem*[46] *takes a vector* $\vec{y} \in \{0,1\}^{8\cdot\binom{n}{3}}$ *as input, and the goal is to decide whether the following quantified Boolean formula holds:*

$$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \bigwedge_{u \in [8\cdot\binom{n}{3}]} [\neg y_u \vee \Phi_u(\vec{x})]\,. \tag{1}$$

where Q_i *equals* $\exists$ *for odd* i, $\forall$ *for even* i, *and* $\vec{x} = (x_1, \ldots, x_n)$.

For $u \in [8 \cdot \binom{n}{3}]$, *letting* $(S, \vec{\tau}) = \phi_n^{\mathsf{cl\text{-}idx}}(u)$, *where* $S = \{s_1, s_2, s_3\}$ *for* $s_1 < s_2 < s_3$, $\Phi_u(\vec{x})$ *is defined as*

$$\Phi_u(\vec{x}) = \bigvee_{i \in [3]} [(x_{s_i} \wedge \tau_i) \vee (\neg x_{s_i} \wedge \neg \tau_i)]\,.$$

We use $\mathsf{TQBF}^{\mathsf{u}}_n$ *to denote the* $\mathsf{TQBF}^{\mathsf{u}}$ *problem with parameter* n *(and input length* $8 \cdot \binom{n}{3}$*).*

We first show that $\mathsf{TQBF}^{\mathsf{u}}$ is still PSPACE-complete.

Lemma 7.5. $\mathsf{TQBF}^{\mathsf{u}}$ *is* PSPACE*-complete.*

Proof. Recall that the standard TQBF problem is defined as follows: given an n-variable m-clause 3-CNF $\phi(\vec{x})$ as input, the goal is to decide whether $Q_1 x_1 Q_2 \cdots Q_n x_n\, \phi(\vec{x})$ holds, where Q_i equals $\exists$ for odd i, and $\forall$ for even i. By adding dummy variables or dummy clauses, we can assume $n = m$.

Let $\vec{y} = 0^{8\cdot\binom{n}{3}}$ initially. For every $j \in [n]$, letting $C_j(\vec{x})$ be the j-th clause in $\phi(\vec{x})$, there exists an index $u \in [8 \cdot \binom{n}{3}]$ such that $C_j(\vec{x}) \equiv \Phi_u(\vec{x})$ and we set $y_u = 1$.

Now we can verify that $\mathsf{TQBF}^{\mathsf{u}}(\vec{y}) = \mathsf{TQBF}(\phi)$ from (1). This proves the PSPACE-hardness of $\mathsf{TQBF}^{\mathsf{u}}$ as TQBF is PSPACE-complete [SM73] (see also [AB09, Theorem 4.13]). From its definition, it is also clear that $\mathsf{TQBF}^{\mathsf{u}}$ is in PSPACE, which completes the proof. ∎

7.3.1 Construction of $\mathscr{F}^{\mathsf{TV}}$

Next, we will formally define the collection of polynomials $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n \in \mathbb{N}_{\geq 1},\, i \in [n]}$.

We begin with some notation. For a vector $\vec{x} \in \mathbb{F}_n^n$, $i \in [n]$, and $z \in \mathbb{F}_n$, we use $\vec{x}^{i \leftarrow z}$ to denote the vector obtained from $\vec{x}$ by changing x_i to z. For a polynomial $p(\vec{x}) \colon \mathbb{F}^n \to \mathbb{F}$ and $i \in [n]$, we use $\deg_{x_i}(p)$ to denote the maximum degree of x_i in p.

We also state the following lemma, which details how the self-reduction Red in Item (2) of Lemma 7.3 is implemented.

Lemma 7.6 (Self-reduction for $\mathscr{F}^{\mathsf{TV}}$)**.** *Let* $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n \in \mathbb{N}_{\geq 1},\, i \in [n]}$ *and* Term *be as in Lemma 7.3. For every* $n, i \in \mathbb{N}_{\geq 1}$ *such that* $i < n$, *one can compute an index* $J = J_{n,i} \in [n]$ *and a type* $Q = Q_{n,i} \in \{\exists, \forall, \mathsf{LIN}, \mathsf{MUL}\}$ *in* $\mathrm{poly}(n)$ *time such that the following hold for every vector* $\vec{x} \in \mathbb{F}_n^n$*:*

1. *If* $Q = \forall$, *then*
$$f_{n,i}(\vec{x}) = f_{n,i+1}(\vec{x}^{J \leftarrow 0}) \cdot f_{n,i+1}(\vec{x}^{J \leftarrow 1}).$$
2. *If* $Q = \exists$, *then*
$$f_{n,i}(\vec{x}) = 1 - (1 - f_{n,i+1}(\vec{x}^{J \leftarrow 0})) \cdot (1 - f_{n,i+1}(\vec{x}^{J \leftarrow 1})).$$
3. *If* $Q = \mathsf{LIN}$, *then*
$$f_{n,i}(\vec{x}) = x_J \cdot f_{n,i+1}(\vec{x}^{J \leftarrow 1}) + (1 - x_J) \cdot f_{n,i+1}(\vec{x}^{J \leftarrow 0}).$$

[46] u stands for universal, since in (1) we have a universal formula that can simulate every n-variable 3-CNF.

4. *If* $Q = \mathsf{MUL}$, *then*

$$f_{n,i}(\vec{x}) = \mathsf{Term}_{n,i}(\vec{x}) \cdot f_{n,i+1}(\vec{x}).$$

To simplify our presentation, we further define three polynomials $S_\exists, S_\forall, S_{\mathsf{LIN}}$ as

1. $S_\forall(x, y_0, y_1) = y_0 \cdot y_1$.
2. $S_\exists(x, y_0, y_1) = 1 - (1 - y_0) \cdot (1 - y_1)$.
3. $S_{\mathsf{LIN}}(x, y_0, y_1) = xy_1 + (1 - x)y_0$.

Now the first three cases in Lemma 7.6 can be succinctly written as

$$f_{n,i}(\vec{x}) = S_Q(x_j, f_{n,i+1}(\vec{x}^{j\leftarrow 0}), f_{n,i+1}(\vec{x}^{j\leftarrow 1})). \tag{2}$$

Construction of $\mathscr{F}^{\mathsf{TV}}$. Now we are ready to define $\mathscr{F}^{\mathsf{TV}}$. Let $n \in \mathbb{N}$ and m be the largest integer such that $20m^6 \le n$. We will use $\{f_{n,i}\}_{i\in[n]}$ to encode the problem $\mathsf{TQBF}^{\mathsf{u}}_m$. When $n < 20$, we set $f_{n,i}$ to be the constant-zero n-variate polynomial for all $i \in [n]$. So we can assume $m \ge 1$. We also let $m_y = 8 \cdot \binom{m}{3}$. (Note that $\mathsf{TQBF}^{\mathsf{u}}_m$ has m_y input bits.)

Recall that for $i \in [m]$, $Q_i = \exists$ for odd i and $Q_i = \forall$ for even i.

Letting $\lambda = m + m_y + 1$. For convenience, we first construct two other polynomial families $\{g^{(n)}_{i,j} : \mathbb{F}^n_n \to \mathbb{F}_n\}_{i,j\in[\lambda]}$ and $\{\mathsf{Term}^{(n)}_i : \mathbb{F}^n_n \to \mathbb{F}_n\}_i$ as follows:

1. $g^{(n)}_{\lambda,j}$ are all set to be constant-one polynomials for every $j \in [\lambda]$.
2. For every i from $m + m_y$ down to 1:
   (a) If $i \in [m]$, we set
   $$g^{(n)}_{i,\lambda}(\vec{x}) = S_{Q_i}(x_i, g^{(n)}_{i+1,1}(\vec{x}^{i\leftarrow 0}), g^{(n)}_{i+1,1}(\vec{x}^{i\leftarrow 1})) \tag{3}$$
   for every $\vec{x} \in \mathbb{F}^n_n$.
   (b) Otherwise $i > m$. Let $u = m + m_y + 1 - i$ and $(S, \vec{\tau}) = \phi^{\mathsf{cl\text{-}idx}}_m(u)$ such that $S = \{s_1, s_2, s_3\} \subseteq [m]$ where $s_1 < s_2 < s_3$. We define
   $$\mathsf{Term}^{(n)}_i(\vec{x}) = 1 - (1 - x_{m+u}) \cdot \prod_{\ell\in[3]} \Big[1 - [\tau_\ell \cdot x_{s_\ell} + (1 - \tau_\ell) \cdot (1 - x_{s_\ell})]\Big], \tag{4}$$
   and
   $$g^{(n)}_{i,\lambda}(\vec{x}) = g^{(n)}_{i+1,1}(\vec{x}) \cdot \mathsf{Term}^{(n)}_i(\vec{x}). \tag{5}$$
   (c) For every j from $m + m_y$ down to 1, we set
   $$g^{(n)}_{i,j}(\vec{x}) = S_{\mathsf{LIN}}(x_j, g^{(n)}_{i,j+1}(\vec{x}^{j\leftarrow 0}), g^{(n)}_{i,j+1}(\vec{x}^{j\leftarrow 1})). \tag{6}$$

Intuitively speaking, in the process above, we start from the base constant-one polynomial and keep multiplying the last polynomial with the arithmetization of a single term in (1), and then apply linearization to all variables to reduce the individual degrees of all variables. Then, we obtain the multi-linear extension of the base formula[47]

$$\bigwedge_{u\in[8\cdot\binom{n}{3}]} [\neg x_{m+u} \vee \Phi_u(\vec{x})]. \tag{7}$$

[47] Note that here we concatenate $\vec{x}$ and $\vec{y}$, so y_u corresponds to x_{m+u}.

After that, we apply the $\exists$ and $\forall$ quantifiers alternatively, each followed by LIN operators, just as in [TV07].

We will first prove some properties of the family $\{g_{i,j}^{(n)} : \mathbb{F}_n^n \to \mathbb{F}_n\}_{i,j\in[\lambda]}$, and then show how to construct $\mathscr{F}^{\mathsf{TV}}$ based on them.

Lemma 7.7. *For every $i \in [\lambda]$ the following hold:*

1. *The individual degree of $g_{i,1}^{(n)}$ is at most 1.*
2. *For every $j \in [\lambda]$, the individual degree of $g_{i,j}^{(n)}$ is at most 2.*
3. *$g_{i,1}^{(n)}$ agrees with $g_{i,\lambda}^{(n)}$ on all points from $\{0,1\}^n$.*

Proof. First, we observe that for every $i, j \in [\lambda]$, $g_{i,j}^{(n)}$ does not depend on any variable x_k with $k > m + m_y$ (*i.e.*, $\deg_{x_k}(g_{i,j}^{(n)}) = 0$). Next, by the definition of S_{LIN} together with (6), we can see that for every $i, j \in [\lambda - 1]$, we have (1) $\deg_{x_j}(g_{i,j}^{(n)}) \le 1$, (2) $\deg_{x_\ell}(g_{i,j}^{(n)}) \le \deg_{x_\ell}(g_{i,j+1}^{(n)})$ for $\ell \in [n] \setminus \{j\}$, and (3) $g_{i,j}^{(n)}$ and $g_{i,j+1}^{(n)}$ agree on all points from $\{0,1\}^n$.

From the discussions above, Item (1) and (3) follow immediately (the case of $i = \lambda$ follows from our definition). To see Item (2), again by the above discussions, it suffices to verify that the individual degree of $g_{i,\lambda}^{(n)}$ for every $i \in [\lambda - 1]$ are bounded by 2. When $i \in [m]$, this follows from the definition (3) and the fact that $g_{i+1,1}^{(n)}$ has individual degree at most 1. When $i > m$ it follows from the fact that $\mathsf{Term}_i^{(n)}$ has individual degree at most 1 and the definition (5). ∎

Lemma 7.8. *$g_{m+1,1}^{(n)}$ is the linear extension of (7).*[48]

Proof. We first note that for every $i \in [m + m_y] \setminus [m]$, letting $\mu_i = m + m_y + 1 - i$, from (4), $\mathsf{Term}_i^{(n)}$ is the linear extension of the clause

$$C_i := \neg x_{m+\mu_i} \vee \Phi_{\mu_i}(\vec{x}).$$

We will use a simple induction to prove the following claim, which easily implies this lemma (by setting $i = m + 1$).

Claim 7.9. *For every $i \in [m + m_y] \setminus [m]$, $g_{i,1}^{(n)}$ is the linear extension of*

$$\Psi_i := \bigwedge_{u\in[\mu_i]} \left[\neg x_{m+u} \vee \Phi_u(\vec{x})\right].$$

The base case $i = m + m_y$ can be established by the fact that $g_{i,\lambda}^{(n)} = g_{i+1,1}^{(n)} \cdot \mathsf{Term}_i^{(n)} = \mathsf{Term}_i^{(n)}$ and Item (3) of Lemma 7.7. Now, assuming that the claim holds for $i + 1 \in \{m + 2, \ldots, m + m_y\}$, we show it holds for i as well. Since $g_{i+1,1}^{(n)}$ is the linear extension of Ψ_{i+1}, $\Psi_i = \Psi_{i+1} \wedge C_i$, and $\mathsf{Term}_i^{(n)}$ is the linear extension of C_i, we know that $g_{i,\lambda}^{(n)} = g_{i+1,1}^{(n)} \cdot \mathsf{Term}_i^{(n)}$ agrees with Ψ_i on all points from $\{0,1\}^n$. Now, from Item (1) and Item (3) of Lemma 7.7, we know that $g_{i,1}^{(n)}$ is both multi-linear and agrees with Ψ_i on all points from $\{0,1\}^n$. Therefore, $g_{i,1}^{(n)}$ is the linear extension of Ψ_i. ∎

[48] We treat (7) as a Boolean function on n bits by adding $n - (m + m_y)$ dummy variables at the end.

Finally, we show that $g_{1,1}^{(n)}$ correctly encodes $\mathsf{TQBF}_m^{\mathsf{u}}$.

Lemma 7.10. *For every $\vec{x} \in \{0,1\}^m$, $\vec{z} \in \{0,1\}^{n-m-m_y}$, and $\vec{y} \in \{0,1\}^{m_y}$, $g_{1,1}^{(n)}(\vec{x},\vec{y},\vec{z}) = \mathsf{TQBF}_m^{\mathsf{u}}(\vec{y})$.*

Proof. For every $i \in [m+1]$, we define the following quantified formula

$$\Lambda_i(\vec{x},\vec{y}) := Q_i x_i Q_{i+1} x_{i+1} \cdots Q_m x_m \bigwedge_{u \in [8\cdot\binom{m}{3}]} [\neg y_u \vee \Phi_u(\vec{x})],$$

where $\vec{x} \in \{0,1\}^{i-1}$ and $\vec{y} \in \{0,1\}^{m_y}$.

In particular, $\Lambda_{m+1}(\vec{x},\vec{y}) = \bigwedge_{u \in [8\cdot\binom{m}{3}]} [\neg y_u \vee \Phi_u(\vec{x})]$ and $\Lambda_1(\vec{y}) = \mathsf{TQBF}_m^{\mathsf{u}}(\vec{y})$.

We will again use a simple induction to prove the following claim, which easily implies this lemma (by setting $i = 1$).

Claim 7.11. *For every $i \in [m+1]$, $\vec{x} \in \{0,1\}^{i-1}$, $\vec{w} \in \{0,1\}^{m-(i-1)}$, $\vec{z} \in \{0,1\}^{n-m-m_y}$, and $\vec{y} \in \{0,1\}^{m_y}$, $g_{i,1}^{(n)}(\vec{x},\vec{w},\vec{y},\vec{z}) = \Lambda_i(\vec{x},\vec{y})$.*

The base case $i = m+1$ is exactly Lemma 7.8. Now, assuming that the claim holds for $i+1 \in \{2,\ldots,m+1\}$, we show it holds for i as well. Note that $\Lambda_i(\vec{x},\vec{y}) = Q_i x_{i+1}\, \Lambda_{i+1}(\vec{x},x_{i+1},\vec{y})$, the claim then follows from the definition of $g_{i,\lambda}$ (see (3)) and Item (3) of Lemma 7.7. ■

Now, recall that $m_y = 8\cdot\binom{m}{3} \le 2m^3$ and $20m^6 \le n$. We have $\lambda^2 = (m+m_y+1)^2 \le 16\cdot m^6 < n$. We are now ready to define $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i}\}_{n\in\mathbb{N}_{\ge 1},\, i\in[n]}$ as follows: for every $(i,j) \in [\lambda]$, we set $f_{n,(i-1)\cdot\lambda+j} = g_{i,j}^{(n)}$ and for every $\ell \in [n] \setminus [\lambda^2]$, we set $f_{n,\ell}$ to be the constant-one polynomial.

Now we are ready to set $Q_{n,\ell}$, $J_{n,\ell}$, and $\mathsf{Term}_{n,\ell}$ for every $\ell \in [n]$ accordingly to prove Lemma 7.6.

Proof of Lemma 7.6. For every $\ell \in [n-1] \setminus [\lambda^2]$, we set $Q_{n,\ell} = \mathsf{LIN}$ and $J_{n,\ell} = 1$. We note that when $f_{n,\ell+1}$ is the constant-one polynomial, from (2), $f_{n,\ell}$ is also the constant-one polynomial. Hence, Lemma 7.6 holds for $\ell \in [n-1] \setminus [\lambda^2]$.

Next, for every $i,j \in [\lambda]$:

1. Let $\ell = (i-1)\cdot\lambda + j$.
2. If $j < \lambda$, we set $Q_{n,\ell} = \mathsf{LIN}$ and $J_{n,\ell} = j$.
3. Otherwise,
   (a) If $i > m$, we set $Q_{n,\ell} = \mathsf{MUL}$, $J_{n,\ell} = 1$, and $\mathsf{Term}_{n,\ell} = \mathsf{Term}_i^{(n)}$.
   (b) Otherwise $i \in [m]$, we set $Q_{n,\ell} = Q_i$ and $J_{n,\ell} = i$.

For every ℓ that $\mathsf{Term}_{n,\ell}$ is not defined above, we set $\mathsf{Term}_{n,\ell}$ to be the constant-zero function. Lemma 7.6 then follows from the definition of $g_{i,j}^{(n)}$ and $\mathsf{Term}_i^{(n)}$. ■

7.3.2 Proof of Lemma 7.3

Now we are ready to prove Lemma 7.3. We first prove every item except for Item (6).

Proof of Lemma 7.3, except for Item (6). First, Item (1) follows immediately from the definition of $\mathsf{Term}_i^{(n)}$ in (4), and Item (3) of Lemma 7.3 follows immediately from the definition of $f_{n,n}$.

Item (2.b) follows immediately from Lemma 7.6. To see Item (2.c), we note that according to Lemma 7.6, (1) $f_{n,i}(\vec{x})$ can be computed by a degree-2 polynomial over $\vec{x}$ and the outputs returned by queries to $f_{n,i+1}$ and $\mathsf{Term}_{n,i}$, and (2) the queries to $f_{n,i+1}$ or $\mathsf{Term}_{n,i}$ are projections on $\vec{x}$. Item (2.c) then follows from the observation that degree-2 polynomials over $\mathbb{F}_n$ can be computed by $\mathsf{XOR} \circ \mathsf{AND}_2$ circuits.

Item (4) follows from the fact that $f_{n,1} = g_{1,1}^{(n)}$, Lemma 7.10, and the PSPACE-completeness of $\mathsf{TQBF}^{\mathsf{u}}$ (Lemma 7.5); Item (5) follows immediately from the definition of $f_{n,i}$ and Lemma 7.7. ■

Finally, to prove Item (6) of Lemma 7.3, we give a detailed implementation of the instance checker IC in Algorithm 7.1 and show that it can indeed be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ circuit family.

Proof of Item (6) of Lemma 7.3. Fix $n, i \in \mathbb{N}_{\geq 1}$ such that $i \in [n]$. Let $\vec{x} \in \mathbb{F}_n^n$ be the input and $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ be the $n - i + 1$ oracle functions. We first note that throughout Algorithm 7.1, we have $r_j = \tilde{f}_j(\vec{\alpha}_j)$ for every $j \in \{i, \ldots, n\}$.

Completeness. We first establish Item (6.a) (*i.e.*, the completeness). Assuming that $\tilde{f}_j = f_{n,j}$ for every $j \in \{i, \ldots, n\}$, it follows that in Algorithm 7.1, $D_{n,t}^{\mathsf{intp}}(\mathcal{L}, \alpha)$ always equals to $f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow \alpha})$ for every $\alpha \in \mathbb{F}_n$.[49] Since we also have $r_j = \tilde{f}_j(\vec{\alpha}_j)$, by Lemma 7.6, we know that Algorithm 7.1 always passes the check on Line 11, Line 14, and Line 18. Finally, since $r_n = \tilde{f}_n(\vec{\alpha}_n) = f_{n,n}(\vec{\alpha}_n) = 1$ (by Item (3) of Lemma 7.3), we know that IC outputs $r_i = \tilde{f}_i(\vec{\alpha}_i) = f_{n,i}(\vec{x})$ with probability 1.

Soundness. Next we establish Item (6.b) (*i.e.*, the soundness). We will do so by establishing the following claim.

Claim 7.12. *For every $j \in \{i, \ldots, n\}$, if $r_j \neq f_{n,j}(\vec{\alpha}_j)$, then for every fixed randomness $z_i, \ldots, z_{j-1}$, with probability at least $1 - (n - j) \cdot c_{\mathsf{deg}}/2^n$ over the random choice of $z_j, \ldots, z_{n-1}$, we have $\mathsf{IC}_{n,i}^{\tilde{f}_i, \ldots, \tilde{f}_n}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$.*

Proof. We will prove the claim by induction. For the base case $j = n$, the claim immediately follows from the final check (Line 20) of Algorithm 7.1 since IC would always output $\perp$.

Now, assuming that the claim holds for $j + 1 \in \{i + 1, \ldots, n\}$, we will show it holds for j as well. The case for $Q = \mathsf{MUL}$ follows straightforwardly from the definition of $f_{n,j}$ and $f_{n,j+1}$, so in the following we assume $Q \in \{\exists, \forall, \mathsf{LIN}\}$.

Let $p \colon \mathbb{F}_n \to \mathbb{F}_n$ be the degree-c_{deg} polynomial such that $p(\alpha) = D_{n,t}^{\mathsf{intp}}(\mathcal{L}, \alpha)$ for every $\alpha \in \mathbb{F}_n$ (see Item (2) of Corollary 7.2), and $q \colon \mathbb{F}_n \to \mathbb{F}_n$ be the restriction of $f_{n,j+1}$ defined by $q(\alpha) = f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow \alpha})$. Note that q has degree at most c_{deg} by Item (5) of Lemma 7.3. We consider two cases separately, $p = q$ and $p \neq q$.

First, suppose $p = q$. In this case, we have

$$\begin{aligned} S_Q((\vec{\alpha}_j)_J, D_{n,t}^{\mathsf{intp}}(\mathcal{L}, 0), D_{n,t}^{\mathsf{intp}}(\mathcal{L}, 1)) &= S_Q((\vec{\alpha}_j)_J, q(0), q(1)) \\ &= S_Q((\vec{\alpha}_j)_J, f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow 0}), f_{n,j+1}((\vec{\alpha}_j)^{J \leftarrow 1})) = f_{n,j}(\vec{\alpha}_j). \end{aligned}$$

[49] Here we crucially used the fact that the individual degree of $f_{n,i}$ is bounded by the constant c_{deg}, so that Corollary 7.2 can be applied.

Algorithm 7.1: The instance checker IC from Item (6) of Lemma 7.3

Given $n, i \in \mathbb{N}_{\geq 1}$ such that $i \in [n]$ as input parameters, and $\vec{x} \in \mathbb{F}_n$ as the input;
Given $n - i + 1$ functions $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_n \colon \mathbb{F}_n^n \to \mathbb{F}_n$ as the oracles;
Let $\vec{\alpha}_i = \vec{x}$ and $r_i = \tilde{f}_i(\vec{x})$; `// The initial goal is to verify the claim` $f_{n,i}(\vec{x}) = r_i$
for $j \in \{i, i+1, \ldots, n-1\}$ **do**
    `/* The goal at the` j`-th stage is to verify the claim` $f_{n,j}(\vec{\alpha}_j) = r_j$ `*/`
    Compute $J = J_{n,j}$ and $Q = Q_{n,j}$ from Lemma 7.6;
    Draw $z_j \in_{\mathsf{R}} \mathbb{F}_n$;
    if $Q \in \{\exists, \forall, \mathsf{LIN}\}$ **then**
        Let $t = c_{\mathsf{deg}} + 1$ and $w_1, \ldots, w_t$ be the first t non-zero elements in $\mathbb{F}_n$; `//` c_{deg} `is the constant from Item (5) of Lemma 7.3`
        Set $\beta_\ell = \tilde{f}_{j+1}((\vec{\alpha}_j)^{J \leftarrow w_\ell})$ for every $\ell \in [t]$;
        Let $\mathcal{L} = \{\beta_\ell\}_{\ell \in [t]}$;
        if $r_j \neq S_Q((\vec{\alpha}_j)_J, D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 0), D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 1))$ **then**
            return $\perp$;
        `/* Reduce the verification of` $f_{n,j}(\vec{\alpha}_j) = r_j$ `to the verification of` $f_{n,j}(\vec{\alpha}_{j+1}) = r_{j+1}$ `*/`
        Set $\vec{\alpha}_{j+1} = (\vec{\alpha}_j)^{J \leftarrow z_j}$ and $r_{j+1} = \tilde{f}_{j+1}(\vec{\alpha}_{j+1})$;
        if $r_{j+1} \neq D^{\mathsf{intp}}_{n,t}(\mathcal{L}, z_j)$ **then**
            return $\perp$;
    else
        Set $\vec{\alpha}_{j+1} = \vec{\alpha}_j$ and $r_{j+1} = \tilde{f}_{j+1}(\vec{\alpha}_{j+1})$;
        if $r_j \neq r_{j+1} \cdot \mathsf{Term}_{n,j}(\vec{\alpha}_j)$ **then**
            return $\perp$;
`/* Finally, check the final claim to see if it is correct. It should equal to 1 according to Item (3) of Lemma 7.3 */`
if $r_n = 1$ **then**
    return r_i;
else
    return $\perp$;

Hence, it follows that $r_j \neq S_Q((\vec{\alpha}_j)_J, D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 0), D^{\mathsf{intp}}_{n,t}(\mathcal{L}, 1))$ and IC does not pass the test on Line 11, and outputs $\perp$ immediately.

Next, suppose $p \neq q$. Since both polynomials have degree at most c_{deg} and note that z_j is independent of p and q (they both only depend on $z_i, \ldots, z_{j-1}$, which are fixed by the assumption), it follows that $p(z_j) = q(z_j)$ with probability at most $c_{\mathsf{deg}}/|\mathbb{F}_n| \leq c_{\mathsf{deg}}/2^n$. Conditioning on the event that $p(z_j) \neq q(z_j)$, we have that $r_{j+1} = p(z_j)$ (otherwise the check on Line 14 fails and IC outputs $\perp$ immediately) and $f_{n,j+1}(\vec{\alpha}_{j+1}) = q(z_j) \neq r_{j+1}$. Hence, by the induction hypothesis, it follows that $\mathsf{IC}^{\tilde{f}_i,\ldots,\tilde{f}_n}_{n,i}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$ happens with probability at least $1 - (n-j) \cdot c_{\mathsf{deg}}/2^n$ over the random choice of $z_j, \ldots, z_{n-1}$. ∎

Applying the claim above with $j = i$, we have that $\mathsf{IC}^{\tilde{f}_i,\ldots,\tilde{f}_n}_{n,i}(\vec{x}) \in \{f_{n,i}(\vec{x}), \perp\}$ with probability at least $1 - n \cdot c_{\mathsf{deg}}/2^n$. This error probability can be further amplified to at most $1/2^n$ (the stated error probability in Item (6) of Lemma 7.3) as follows: run the IC $t = O(1)$ times to obtain outputs $\alpha_1, \ldots, \alpha_t$; output $\perp$ if any of the α_i equals $\perp$ or they are not all identical (*i.e.*, there are $i < j$ such that $\alpha_i \neq \alpha_j$), and output α_1 otherwise. The amplification procedure can be implemented in AC^0, so it does not affect the complexity of IC, which is discussed below.

Complexity. Finally, we show that IC can be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ circuit family.

The crucial observation here is that we can first draw $z_i, \ldots, z_{n-1} \in_{\mathsf{R}} \mathbb{F}_n$ beforehand and run each iteration of the for loop in Algorithm 7.1 in parallel (and return $\perp$ if any of the checks on Line 11, Line 14, or Line 18 fails). Note that for each $j \in \{i, \ldots, n\}$ and $\ell \in [n]$, we have

$$(\vec{\alpha}_j)_\ell = \begin{cases} x_\ell & \text{there is no } j' < j \text{ such that } J_{n,j'} = \ell \text{ and } Q_{n,j'} \neq \mathsf{MUL} \\ z_{j_{\mathsf{max}}} & \text{otherwise,} \end{cases} \tag{8}$$

where j_{max} is the maximum $j' < j$ such that $J_{n,j'} = \ell$ and $Q_{n,j'} \neq \mathsf{MUL}$.

Using (8), for every $j \in \{i, \ldots, n\}$, we can compute $\vec{\alpha}_j$ by a uniform projection given $\vec{x}, z_i, \ldots, z_{n-1}$. It then follows from Algorithm 7.1, Corollary 7.2, and Lemma 7.1 that IC can be implemented by a randomized uniform non-adaptive AC^0_2 circuit family. ∎

7.4 Construction of the PSPACE-complete Language

In this section, we prove Theorem 3.15. We will first construct a PSPACE-complete language $L^{\mathsf{WH\text{-}TV}}$, and then prove it satisfies all the desired properties stated in Theorem 3.15 except for the paddability. Then we modify $L^{\mathsf{WH\text{-}TV}}$ into another PSPACE-complete language L^{PSPACE} that also satisfies the paddability.

7.4.1 The Language $L^{\mathsf{WH\text{-}TV}}$

To construct our PSPACE-complete language $L^{\mathsf{WH\text{-}TV}}$, we apply Walsh-Hadamard codes to turn the polynomials from Lemma 7.3 into Boolean functions. (Indeed, WH-TV stands for "Walsh-Hadamard version of $\mathscr{F}^{\mathsf{TV}}$".)

Let $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i} \colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n \in \mathbb{N}_{\geq 1},\, i \in [n]}$ be as in Lemma 7.3. First, we list all polynomials in $\mathscr{F}^{\mathsf{TV}}$ in the following order

$$f_{1,1}, f_{2,2}, \ldots, f_{2,1}, f_{3,3}, \ldots, f_{3,1}, \ldots, f_{n,n}, \ldots, f_{n,1}, \ldots. \tag{9}$$

For every $k \in \mathbb{N}$, we let g_k be the k-th polynomial in (9). We also set n_k and i_k so that $g_k = f_{n_k,i_k}$. When their meanings are clear from the context, we may write n and i instead of n_k and i_k. Recall that κ_n is the bijection from $\{0,1\}^{\mathsf{sz}_n}$ to $\mathbb{F}_n$ described in Section 7.1.

Construction of the interpolated polynomial G_k. We now define the following polynomial $G_k \colon \mathbb{F}_n^n \times \mathbb{F}_n^n \to \mathbb{F}_n$:

$$G_k(\vec{x}, \vec{y}) := \sum_{j=i_k}^{n_k} f_{n_k,j}(x) \cdot y_j, \tag{10}$$

where $\vec{x} \in \mathbb{F}_n^n$ and $\vec{y} \in \mathbb{F}_n^n$.

We define $F_k \colon \mathbb{F}_n^{2n} \times \{0,1\}^{\mathsf{sz}_n} \to \{0,1\}$ as

$$F_k(\vec{z}, \vec{r}) := \langle \kappa_n^{-1}(G_k(\vec{z})), \vec{r} \rangle, \tag{11}$$

where $\langle \kappa_n^{-1}(G_k(\vec{z})), \vec{r} \rangle$ denotes the inner product between the two vectors over $\mathsf{GF}(2)$.

F_k can be interpreted as a function from $\{0,1\}^{e_k}$ to $\{0,1\}$, where $e_k = (2 \cdot n_k + 1) \cdot \mathsf{sz}_{n_k}$. The following claim follows immediately from the definition of e_k.

Claim 7.13. *For every $k \in \mathbb{N}_{\geq 1}$, it holds that $e_k < e_{k+1}$.*

The language $L^{\mathsf{WH\text{-}TV}}$. Now we are ready to define $L^{\mathsf{WH\text{-}TV}}$ via the following algorithm.

Algorithm 7.2: Algorithm $A^{\mathsf{WH\text{-}TV}}$ for $L^{\mathsf{WH\text{-}TV}}$

```
Given an input x ∈ {0,1}^m for some m ∈ ℕ;
if m < e_1 then
    return 0
Let k be the largest integer such that e_k ≤ m;
return F_k(x_{≤ e_k});
```

From Claim 7.13 and Algorithm 7.2, the following claim is immediate.

Claim 7.14. *For every $k \in \mathbb{N}_{\geq 1}$, $L^{\mathsf{WH\text{-}TV}}_{e_k}$ equals F_k.*

7.4.2 Verifying Properties of $L^{\mathsf{WH\text{-}TV}}$

Next, we verify that $L^{\mathsf{WH\text{-}TV}}$ has all the desired properties stated in Theorem 3.15. First, we show $L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ same-length checkable.

Lemma 7.15. *$L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ same-length checkable.*

Proof. Let $m \in \mathbb{N}$ be an input length, and we can assume $m \geq e_1$ since otherwise, $A^{\mathsf{WH\text{-}TV}}_m$ computes the constant-zero function. Let k be the largest integer such that $e_k \leq m$. Note that it suffices to establish the instance checkability of F_k. In the following, we use n to denote n_k and i to denote i_k.

Instance checker for G_k. We first show how to establish an instance checker $\mathsf{G\text{-}IC}$ for G_k.

Recall that

$$G_k(\vec{x}, \vec{y}) := \sum_{j=i}^{n} f_{n,j}(x) \cdot y_j, \tag{12}$$

for every $\vec{x}, \vec{y} \in \mathbb{F}_n^n$.

Note that for every $j \in \{i, i+1, \ldots, n\}$, letting $\vec{r}_j$ be the vector from $\mathbb{F}_n^n$ with every entry being 0 except for the j-th entry being 1, we have

$$f_{n,j}(\vec{x}) = G_k(\vec{x}, \vec{r}_j) \quad \text{for every } \vec{x} \in \mathbb{F}_n^n, \tag{13}$$

meaning that the oracle access to $f_{n,j}$ can be simulated by the oracle access to G_k via fixing part of the input. G-IC works as follows:

1. Given $\vec{x} \in \mathbb{F}_n^n$ and $\vec{y} \in \mathbb{F}_n^n$ as input, and access to an oracle $\tilde{G} \colon \mathbb{F}_n^{2n} \to \mathbb{F}_n$ that is supposed to compute G_k.
2. For every $j \in \{i, i+1, \ldots, n\}$, G-IC runs $\mathsf{IC}_{n,j}$ (from Item (6) of Lemma 7.3) on input $\vec{x}$ with oracle access to $\tilde{f}_{j+1}, \ldots, \tilde{f}_n$ simulated by $\tilde{G}$ via (13) to obtain an output $u_j \in \mathbb{F}_n \cup \{\perp\}$.[50]
3. If any of the u_j equals $\perp$, we output $\perp$. Otherwise, we output $\sum_{j=i}^{n} u_j \cdot y_j$.

Since $\mathsf{IC}_{n,i}$ can be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ oracle circuit, so does G-IC. (Applying Lemma 7.1, we can use a uniform $\mathsf{AC}^0[2]$ circuit to implement the algorithm above.)

Now we show that when $\tilde{G} = G_k$, G-IC outputs $G_k(\vec{x}, \vec{y})$ with probability 1. Note that for every $j \in \{i, i+1, \ldots, n\}$, since $\tilde{G} = G_k$, we have $\tilde{f}_\ell = f_{n,\ell}$ for every $\ell \in \{j+1, \ldots, n\}$ from (13). Applying Item (6) of Lemma 7.3, it holds that with probability 1, $u_j = f_{n,j}(\vec{x})$ for every $j \in \{i, i+1, \ldots, n\}$. Therefore, with probability 1, G-IC outputs $\sum_{j=i}^{n} u_j \cdot y_j$, which equals $G_k(\vec{x}, \vec{y})$ by definition.

Next we show that for every oracle $\tilde{G}$, with probability at least $2/3$, $\text{G-IC}^{\tilde{G}}$ outputs either $G_k(\vec{x}, \vec{y})$ or $\perp$. We first note that by Item (6) of Lemma 7.3 and a union bound, with probability at least $2/3$, $u_j \in \{f_{n,j}(\vec{x}), \perp\}$ for every $j \in \{i, i+1, \ldots, n\}$, which implies that G-IC outputs either $G_k(\vec{x}, \vec{y})$ (when no u_j equals $\perp$) or $\perp$ (when some u_j equals $\perp$). This completes the construction of the instance checker G-IC for G_k.

Instance checker for F_k. Next we show how to construct the desired instance checker F-IC for F_k:

1. Given $\vec{x} \in \mathbb{F}_n^{2n}$ and $\vec{z} \in \{0,1\}^{\mathsf{sz}_n}$ as input, and access to an oracle $\tilde{F} \colon \mathbb{F}_n^{2n} \times \{0,1\}^{\mathsf{sz}_n} \to \{0,1\}$ that is supposed to compute F_k.
2. F-IC simulates G-IC on input $\vec{x}$ given oracle access to the function[51]

   $$\vec{x} \mapsto \tilde{F}(\vec{x}, \vec{e}_1) \circ \tilde{F}(\vec{x}, \vec{e}_2) \circ \cdots \circ \tilde{F}(\vec{x}, \vec{e}_{\mathsf{sz}_n}),$$

   to obtain an output $u \in \mathbb{F}_n \cup \{\perp\}$.
3. F-IC outputs $\perp$ if u equals $\perp$ and outputs $\langle \kappa_n^{-1}(u), \vec{z} \rangle$ (inner product is over $\mathsf{GF}(2)$) otherwise.

Since we encode an element of $\mathbb{F}_n$ via κ_n, when $\tilde{F} = F_k$, G-IC above indeed gets access to G_k, and hence F_k outputs $\langle \kappa_n^{-1}(G_k(\vec{x})), \vec{z} \rangle = F_k(\vec{x}, \vec{z})$. Also, for every oracle $\tilde{F}$, from the promise of G-IC, we know that G-IC outputs an element in $\{G_k(\vec{x}), \perp\}$ with probability at least $2/3$. This implies that F-IC outputs an element in $\{F_k(\vec{x}, \vec{z}), \perp\}$ with probability at least $2/3$ as well. Therefore, F-IC is an instance checker for F_k. Since G-IC can be implemented by a randomized uniform non-adaptive $\mathsf{AC}^0[2]$ oracle circuit, so does F-IC. ■

[50] That is, $\tilde{f}_\ell(\vec{x}) = \tilde{G}(\vec{x}, \vec{r}_\ell)$ for every $\ell \in \{j+1, \ldots, n\}$.

[51] Below $\vec{e}_\ell$ denotes the sz_n-bit vector with every entry being 0 except for the ℓ-th entry being 1

To prove that $L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ weakly error correctable, we need the following standard decoder for Reed-Muller codes. While the decoding algorithm is a standard interpolation, we analyze its complexity carefully using Lemma 7.1 and show that the decoder can be implemented by an $\mathsf{AC}^0[2]$ circuit.

Lemma 7.16 ($\mathsf{AC}^0[2]$ decoder for Reed-Muller code, [AB09, Section 19.3, 19.4])**.** *Let $n, m \in \mathbb{N}_{\geq 1}$ and $\mathbb{F} = \mathbb{F}_n$. Suppose there is a (hidden) degree-d m-variate polynomial P over $\mathbb{F}$, and $\delta \in \left(0, \frac{1}{3(d+1)}\right)$. For any oracle $O\colon \mathbb{F}^m \to \mathbb{F}$ such that*

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

there is a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit C of size $\mathrm{poly}(m, n)$, such that for every $\vec{x} \in \mathbb{F}^m$, $C^O(\vec{x}) = P(\vec{x})$.

Proof. Let $\vec{x} \in \mathbb{F}^m$ be the input. We will show a randomized non-adaptive $\mathsf{AC}^0[2]$ oracle circuit $\mathbf{C}^O$ (with the oracle set to O) of size $\mathrm{poly}(m, n)$ that computes $P(\vec{x})$ with probability at least $2/3$. Then by Adleman's argument, we can obtain a deterministic oracle $\mathsf{AC}^0[2]$ circuit of size $\mathrm{poly}(m, n)$ that correctly computes P by drawing $\mathrm{poly}(m, n)$ independent samples from $\mathbf{C}^O$ and applying approximate majorities to the output (which can be done in AC^0; see Lemma 3.1).

We choose a random vector $\vec{\mathbf{v}} \in_R \mathbb{F}^m$, and for every $\alpha \in \mathbb{F}$ we define $Q(\alpha) = P(\vec{x} + \alpha \cdot \vec{\mathbf{v}})$ (note that $Q\colon \mathbb{F} \to \mathbb{F}$ has degree at most d). Let $\alpha_1, \ldots, \alpha_{d+1}$ be the first $d+1$ non-zero elements from $\mathbb{F}$. We then query O to obtain $\beta_i = O(\vec{x} + \alpha_i \cdot \vec{\mathbf{v}})$ for every $i \in [d+1]$, and output $D^{\mathsf{intp0}}_{n,d+1}(\{\beta_i\}_{i \in [t+1]})$. This algorithm can be implemented in $\mathsf{AC}^0[2]$ by Corollary 7.2.

To show the correctness of the algorithm above. Let $\mathbf{z}$ denote the number of i's from $[d+1]$ such that $O(\vec{x} + \alpha_i \cdot \vec{\mathbf{v}}) \neq Q(\alpha_i)$, then $\mathbb{E}[\mathbf{z}] \leq \delta(d+1)$, and by Markov bound $\Pr[\mathbf{z} = 0] \geq 2/3$. If $\mathbf{z} = 0$, we know that $D^{\mathsf{intp0}}_{n,d+1}(\{\beta_i\}_{i \in [t+1]}) = Q(0) = P(\vec{x})$, which completes the proof. ∎

Lemma 7.17. *$L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ weakly error correctable.*

Proof. Let $m \in \mathbb{N}$ be an input length, and we can assume $m \geq e_1$ since otherwise $L^{\mathsf{WH\text{-}TV}}_m$ computes the constant-zero function. Let k be the largest integer such that $e_k \leq m$. Note that it suffices to establish the weakly error correctability of F_k. Again, in the following, we will also use n to denote n_k and i to denote i_k. Without loss of generality we can assume $m = e_k = (2 \cdot n + 1) \cdot \mathsf{sz}_n$. Let $\mu > 1$ be a sufficiently large universal constant.

Let $\tilde{f}\colon \{0,1\}^m \to \{0,1\}$ be a function that $(1 - 1/m^\mu)$-approximates $L^{\mathsf{WH\text{-}TV}}_m$. By Markov bound and recall that the definition of F_k from (11), for at least a $(1 - 1/m^{\mu-1})$ fraction of inputs $\vec{z} \in \mathbb{F}^{2n}$, we have

$$\Pr_{\vec{r} \in_R \{0,1\}^{\mathsf{sz}_n}}[\tilde{f}(\vec{z}, \vec{r}) = \langle \kappa_n^{-1}(G_k(\vec{z})), \vec{r} \rangle] \geq 1 - 1/m. \tag{14}$$

We say that an input $\vec{z}$ is *good* if (14) holds. If some $\vec{z} \in \mathbb{F}^{2n}$ is good, then for every $i \in [\mathsf{sz}_n]$, we can compute the i-th bit of $\kappa_n^{-1}(G_k(\vec{z}))$ with probability at least $1 - 2^{-4n \cdot \mathsf{sz}_n}$ by the following algorithm:

1. We pick $\vec{r} \in_R \{0,1\}^{\mathsf{sz}_n}$ and output $\tilde{f}(\vec{z}, \vec{r}) \oplus \tilde{f}(\vec{z}, \vec{r} \oplus \vec{e}_i)$, where $\vec{e}_i$ is the sz_n-bit string with 1 on the i-th bit and 0 everywhere else.

2. We repeat this procedure $\mathrm{poly}(n, \mathsf{sz}_n)$ times and take an approximate majority of the results (this can be done in AC^0, by Lemma 3.1).

By a union bound, we can fix the randomness used by the above algorithm and obtain a polynomial-size non-adaptive $\mathsf{AC}^0[2]$ oracle circuit C with $\tilde{f}$ oracle gates that correctly computes $\kappa_n^{-1}(G_k(\vec{z}))$ for every good $\vec{z}$. In other words, $C^{\tilde{f}}$ computes $\kappa_n^{-1}(G_k(\vec{z}))$ on an $(1 - 1/m^{\mu-1})$ fraction of inputs $\vec{z}$. Since $G_k \colon \mathbb{F}^{2n} \to \mathbb{F}$ is a degree-$O(m)$ polynomial and μ is sufficiently large, we can use Lemma 7.16 to compute G_k in the worst-case by a polynomial-size non-adaptive $\mathsf{AC}^0[2]$ oracle circuit D with $\tilde{f}$ oracle gates. From the definition of F_k in (11), we can convert $D^{\tilde{f}}$ into another polynomial-size non-adaptive $\mathsf{AC}^0[2]$ oracle circuit $E^{\tilde{f}}$ that computes F_k. This completes the proof. ■

Next, we prove the "Moreover part" in Theorem 3.15, which also implies that $L^{\mathsf{WH\text{-}TV}}$ is non-adaptive $\mathsf{AC}^0[2]$ downward self-reducible.

Lemma 7.18. *There are two algorithms* DSR *and* Aux *satisfying the following:*

1. Aux *takes $m \in \mathbb{N}_{\geq 1}$ as input parameter and $\vec{x} \in \{0,1\}^m$ as input, and outputs a value from $\{0,1\}$.*
2. Aux *can be implemented by a uniform $\mathsf{AC}^0[2]$ circuit.*
3. DSR *takes $m \in \mathbb{N}_{\geq 1}$ as input parameter and $\vec{x} \in \{0,1\}^m$ as input, and functions $h_1 \colon \{0,1\}^{m-1} \to \{0,1\}$ and $h_2 \colon \{0,1\}^m \to \{0,1\}$ as oracles.*
4. *For every $m \in \mathbb{N}$, $\mathsf{DSR}_m^{L_{m-1}^{\mathsf{WH\text{-}TV}}, \mathsf{Aux}_m}$ computes $L_m^{\mathsf{WH\text{-}TV}}$.*
5. DSR *can be implemented by a uniform non-adaptive $\mathsf{XOR} \circ \mathsf{AND}_3$ oracle circuit family. In more detail,* DSR *first queries its oracle on some* projections *of the input $\vec{x}$ to obtain some intermediate values and then applies an $\mathsf{XOR} \circ \mathsf{AND}_3$ circuit on those intermediate values and the input $\vec{x}$ to obtain the output.*

Proof. Let $m \in \mathbb{N}$ be the input length. We note that when $A_m^{\mathsf{WH\text{-}TV}}$ and $A_{m-1}^{\mathsf{WH\text{-}TV}}$ (we use $A_m^{\mathsf{WH\text{-}TV}}$ to denote the restriction of $A^{\mathsf{WH\text{-}TV}}$ on m-bit inputs; $A^{\mathsf{WH\text{-}TV}}$ is described in Algorithm 7.2) computes the same function on their prefixes, DSR_m and Aux_m can be constructed trivially.

Hence, from now on, we can assume that for some $k \in \mathbb{N}$, $A_{m-1}^{\mathsf{WH\text{-}TV}}$ computes F_{k-1} and $A_m^{\mathsf{WH\text{-}TV}}$ computes F_k (*i.e.*, $m = e_k$).[52] Let n_k and i_k be such that $g_k = f_{n_k, i_k}$. There are two different cases:

1. $i_k = n_k$. In this case we have $n_{k-1} = n_k - 1$ and $i_{k-1} = 1$.
2. $i_k < n_k$. In this case we have $n_{k-1} = n_k$, and $i_{k-1} = i_k + 1$.

For simplicity, in the following, we will use n to denote n_k and i to denote i_k. And our goal is to compute F_k given oracle access to F_{k-1} and Aux_m (we will define Aux_m later).

Case I: $i_k = n_k$. In this case, we know that $g_k = f_{n,n}$, and from (10), we have $G_k(\vec{x}, \vec{y}) = y_n \cdot f_{n,n}(\vec{x}) = y_n$ for every $\vec{x}, \vec{y} \in \mathbb{F}_n^n$ (note that $f_{n,n}$ is the constant-one polynomial by Item (3) of Lemma 7.3). Hence, $F_k(\vec{x}, \vec{z}) = \langle \kappa_n^{-1}(G_k(\vec{x})), \vec{z} \rangle$ can be computed by an $\mathsf{XOR} \circ \mathsf{AND}_2$ circuit without querying Aux_m or F_{k-1}, from which we can construct the desired DSR_m. (Aux_m does not matter here. For concreteness, we set it to be the constant-zero function.)

[52] For convenience, we will simply say that $A_m^{\mathsf{WH\text{-}TV}}$ computes F_k when it computes F_k on its prefix of length e_k.

Case II: $i_k < n_k$. We first note that by the definition of G_{k-1} and G_k in (10)

$$G_k(\vec{x}, \vec{y}) = G_{k-1}(\vec{x}, \vec{y}) + f_{n,i}(\vec{x}) \cdot y_i. \tag{15}$$

for every $\vec{x}, \vec{y} \in \mathbb{F}_n^n$.

We first show how to compute G_k with oracle access to G_{k-1}. From (15), it suffices to compute $f_{n,i}(\vec{x})$ with oracle access to G_{k-1}. Let $\vec{r}_{i+1}$ be the vector from $\mathbb{F}_n^n$ with all entries being 0 except for the $(i+1)$-th entry being 1, we have

$$f_{n,i+1}(\vec{x}) = G_{k-1}(\vec{x}, \vec{r}_{i+1}).$$

Hence, oracle access to $f_{n,i+1}(\vec{x})$ can be simulated by oracle access to G_{k+1} via the projection $\vec{x} \mapsto (\vec{x}, \vec{r}_{i+1})$. Now we can compute $f_{n,i}(\vec{x})$ by running the algorithm $\mathsf{Red}_{n,i}$ with oracle access to $f_{n,i+1}$ simulated by oracle access to G_{k-1}, and we define Aux_m to be a Boolean version[53] of $\mathsf{Term}_{n,i}$ and simulate oracle access to $\mathsf{Term}_{n,i}$ by oracle access to Aux_m. This gives us a non-adaptive $\mathsf{XOR} \circ \mathsf{AND}_2$ circuit computing $f_{n,i}(\vec{x})$ given oracle access to G_{k-1} and Aux_m.

Now, we note that $f_{n,i}(\vec{x}) \cdot y_i$ can now be computed by a non-adaptive $\mathsf{XOR} \circ \mathsf{AND}_3$ circuit given oracle access to G_k and Aux_m, and so does G_k (via (15)).[54] Finally, note that a single query to G_{k-1} can be simulated by $\log |\mathbb{F}_n|$ queries to F_{k-1} and recall the definition of F_k in (11), we obtain the desired non-adaptive $\mathsf{XOR} \circ \mathsf{AND}_3$ oracle computing F_k given oracle access to F_{k-1} and Aux_m, which completes the proof. ■

Finally, we show the PSPACE-completeness of $L^{\mathsf{WH\text{-}TV}}$.

Lemma 7.19. *$L^{\mathsf{WH\text{-}TV}}$ is PSPACE-complete.*

Proof. We first note that $L^{\mathsf{WH\text{-}TV}} \in \mathsf{PSPACE}$ since every downward self-reducible language is in PSPACE (see, *e.g.*, [AB09, Exercise 8.9]).

Let $L \in \mathsf{PSPACE}$, and let $(A_L^{\mathsf{len}}, A_L^{\mathsf{red}})$ be the pair of algorithms in Lemma 7.3. The following is a polynomial-time reduction R_L from L to $L^{\mathsf{WH\text{-}TV}}$:

1. Given an input $x \in \{0,1\}^n$ for $n \in \mathbb{N}$, let $m = A_L^{\mathsf{len}}(n)$.
2. Compute $\vec{z} = A_L^{\mathsf{red}}(x)$ and let $k \in \mathbb{N}$ be such that $g_k = f_{m,1}$.
3. Let $\vec{y}$ be the vector from $\mathbb{F}_m^m$ with all entries being 0 except for the first entry being 1, and $\vec{u} \in \{0,1\}^{\mathsf{sz}_n}$ be the vector that $u_1 = 1$ and $u_j = 0$ for $j > 1$.
4. Output $L_{e_k}^{\mathsf{WH\text{-}TV}}(\vec{z}, \vec{y}, \vec{u})$.

By Lemma 7.3, we have $f_{m,1}(\vec{z}) = (L(x))_{\mathbb{F}_m}$. Since $L(x) \in \{0,1\}$ and we encode $\mathbb{F}_m$ as a Boolean string in $\{0,1\}^{\mathsf{sz}_m}$ via κ_m. One can see that

$$\left(\kappa_m^{-1}(f_{m,1}(\vec{z}))\right)_1 = L(x). \tag{16}$$

Now, by the definition of G_k in (10), we have that $G_k(\vec{z}, \vec{y}) = g_k(\vec{z}) = f_{m,1}(\vec{z})$. Then by the definition of F_k, Claim 7.14 and (16), we have

$$L_{e_k}^{\mathsf{WH\text{-}TV}}(\vec{z}, \vec{y}, \vec{u}) = F_k(\vec{z}, \vec{y}, \vec{u}) = \left(\kappa_m^{-1}(f_{m,1}(\vec{z}))\right)_1 = L(x).$$

Therefore, $L^{\mathsf{WH\text{-}TV}}$ is PSPACE-complete. ■

[53] For example, we can apply the Hadamard-Walsh encoding to turn $\mathsf{Term}_{n,i}$ into a Boolean function, similar to (11).

[54] Here we use the observation that degree-3 polynomials over $\mathbb{F}_n$ can be computed by $\mathsf{XOR} \circ \mathsf{AND}_3$ circuits.

7.4.3 The Final Language L^{PSPACE}

Finally, we modify $L^{\mathsf{WH\text{-}TV}}$ into a new language L^{PSPACE} that is also paddable via Algorithm 7.3.

Algorithm 7.3: Algorithm A^{PSPACE} for L^{PSPACE}

```
Given an input x ∈ {0,1}^m for some m ∈ ℕ;
Let i_cur = 1 and at = 1;
Let res = 0;
while at + i_cur − 1 ≤ m do
    res = res ⊕ L^{WH-TV}_{i_cur}(x_{[at,at+i_cur)}) ;      // ⊕ denotes XOR, and x_{[at,at+i_cur)} denotes
      (x_at, x_{at+1}, ..., x_{at+i_cur−1})
    at = at + i_cur;
    i_cur = i_cur + 1;
return res;
```

In other words, A^{PSPACE} partitions the input $x \in \{0,1\}^m$ into consecutive blocks $x^{[1]}, x^{[2]}, \ldots, x^{[t]}$ of length $1, 2, 3, \ldots$ until running out of the input bits, and output $\bigoplus_{i\in[t]} L_i^{\mathsf{WH\text{-}TV}}(x^{[i]})$.

Now we are ready to prove Theorem 3.15.

Proof of Theorem 3.15. The PSPACE-completeness of L^{PSPACE} follows from the PSPACE-completeness of $L^{\mathsf{WH\text{-}TV}}$. Given an input length $m \in \mathbb{N}_{\geq 1}$. We aim to establish the paddability from $L_{m-1}^{\mathsf{PSPACE}}$ to L_m^{PSPACE}, the downward self-reducibility from L_m^{PSPACE} to $L_{m-1}^{\mathsf{PSPACE}}$, the instance checkability of L_m^{PSPACE}, and the weak error correctability of L_m^{PSPACE}.

Let t be the largest integer such that $\binom{t+1}{2} \leq m$. We first note that if $\binom{t+1}{2} < m$, then indeed A_m^{PSPACE} and $A_{m-1}^{\mathsf{PSPACE}}$ compute the same function on their first $\binom{t+1}{2}$ input bits, and downward self-reducibility and paddability are trivial. So it suffices to consider $m = \binom{t+1}{2}$ for paddability and downward self-reducibility. We can also observe that only considering $m = \binom{t+1}{2}$ suffices for establishing instance checkability and weak error correctability as well. So from now on, we assume $m = \binom{t+1}{2}$ without loss of generality. For an input $x \in \{0,1\}^*$, we use $x^{[i]}$ to denote $x_{\left[\binom{i}{2}+1, \binom{i+1}{2}+1\right)}$.

From the definition of $\mathbb{F}_k$ in (11) and Algorithm 7.2, we can see that $L_\ell^{\mathsf{WH\text{-}TV}}(0^\ell) = 0$ for every $\ell \in \mathbb{N}$. Hence, we have

$$L_{m-1}^{\mathsf{PSPACE}}(x) = \bigoplus_{i=1}^{t-1} L_i^{\mathsf{WH\text{-}TV}}(x^{[i]}) = L_m^{\mathsf{PSPACE}}\left(x_{\leq \binom{t}{2}} \circ 0^t\right),$$

which establishes the paddability.

To see the downward self-reducibility, we note that

$$L_m^{\mathsf{PSPACE}}(x) = \bigoplus_{i=1}^{t} L_i^{\mathsf{WH\text{-}TV}}(x^{[i]}) = L_{m-1}^{\mathsf{PSPACE}}(x_{\leq \binom{t}{2}} \circ 0^{t-1}) \oplus L_t^{\mathsf{WH\text{-}TV}}(x^{[t]}).$$

It is easy to see that oracle access to $L_{t-1}^{\mathsf{WH\text{-}TV}}$ can be simulated via oracle access to $L_{m-1}^{\mathsf{PSPACE}}$ by a projection, so the required algorithms DSR and Aux can be established using the corresponding algorithms from Lemma 7.18.

Now, to see the instance-checkability, we note that for every $i \in [t]$, $L_i^{\mathsf{WH\text{-}TV}}$ can be simulated via oracle access to L_m^{PSPACE} by a projection, hence we can first run the instance checker for $L^{\mathsf{WH\text{-}TV}}$

on each of $x^{[i]}$ with the simulated oracle, returns $\perp$ if any of them returns $\perp$, and output the XOR of all the outputs otherwise. And it is straightforward to see that the instance-checker above for L^{PSPACE} can also be implemented by uniform $\mathsf{AC}^0[2]$ circuits.

Finally, we will prove that the non-adaptive $\mathsf{AC}^0[2]$ weakly error correctability follows from that of $L^{\mathsf{WH\text{-}TV}}$. Let μ be the constant from the weakly error correctability of $L^{\mathsf{WH\text{-}TV}}$ (Lemma 7.17) and $\tilde{f}\colon \{0,1\}^m \to \{0,1\}$ be a function that $(1-m^{-\mu})$-approximates L_m^{PSPACE}. Now we fix an $i \in [t]$, by an averaging principle, there exists $\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}$ such that

$$\Pr_{x^{[i]} \in_{\mathsf{R}} \{0,1\}^i} \left[\tilde{f}(\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, x^{[i]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}) = L_m^{\mathsf{PSPACE}}(\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, x^{[i]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}) \right] \geq 1 - m^{-\mu}.$$

For notational convenience, we use $\tilde{x}^{[-i]}$ to denote $\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}$, and $\tilde{x}^{[-i]} \circ x^{[i]}$ to denote $\tilde{x}^{[1]}, \ldots, \tilde{x}^{[i-1]}, x^{[i]}, \tilde{x}^{[i+1]}, \ldots, \tilde{x}^{[t]}$. From the definition of L_m^{PSPACE}, the above simplifies to

$$\Pr_{x^{[i]} \in_{\mathsf{R}} \{0,1\}^i} \left[\tilde{f}(\tilde{x}^{[-i]} \circ x^{[i]}) = L_i^{\mathsf{WH\text{-}TV}}(x^{[i]}) \oplus \bigoplus_{\ell \in [t] \setminus \{i\}} L_\ell^{\mathsf{WH\text{-}TV}}(\tilde{x}^{[\ell]}) \right] \geq 1 - m^{-\mu} \geq 1 - i^{-\mu}.$$

Now we define $\tilde{g}(x^{[i]}) = \tilde{f}(\tilde{x}^{[-i]} \circ x^{[i]}) \oplus \bigoplus_{\ell \in [t] \setminus \{i\}} L_\ell^{\mathsf{WH\text{-}TV}}(\tilde{x}^{[\ell]})$. By Lemma 7.17, there is an i^μ-size non-adaptive $\mathsf{AC}^0[2]$ circuit $C_{[i]}$ such that $C_{[i]}^{\tilde{g}}$ computes $L_i^{\mathsf{WH\text{-}TV}}$. Since $\tilde{x}^{[-i]}$ is fixed, oracle access to $\tilde{g}$ can be simulated via oracle access to $\tilde{f}$ by a projection, and there is an $O(i^\mu \cdot m)$-size non-adaptive $\mathsf{AC}^0[2]$ circuit $D_{[i]}$ such that $D_{[i]}^{\tilde{f}}$ computes $L_i^{\mathsf{WH\text{-}TV}}$.

Finally, we define a non-adaptive $\mathsf{AC}^0[2]$ oracle circuit $E^{\tilde{f}}(x^{[1]}, \ldots, x^{[t]}) = \bigoplus_{i \in [t]} D_{[i]}^{\tilde{f}}(x^{[i]})$. From the discussions above we know that $E^{\tilde{f}}$ computes L_m^{PSPACE}, and it has size at most $m^{\mu+2}$. Setting $\tau_{\mathsf{wc2ac}} = \mu + 2$ completes the proof. ■

8 Sub-half-exponential Lower Bounds against ACC^0

Reminder of Theorem 1.5. *For every sub-half-exponential reasonable time-bound function $g(n)$, NE has no $g(n)$-size ACC^0 circuits.*

To prove Theorem 1.5, we will need the following lemma.

Lemma 8.1. *Let $g(n)$ be a reasonable time-bound function. Assuming that E has $g(n)$-size ACC^0 circuits, we have*

$$\mathsf{E} \subseteq \mathsf{MATIME}_{\mathsf{ACC}^0}[\mathrm{poly}(g(n))].$$

Proof. Fix $L \in \mathsf{E}$, we will show that $L \in \mathsf{MATIME}_{\mathsf{AC}^0_d[m]}[\mathrm{poly}(g(n))]$ for some constants $d, m \in \mathbb{N}_{\geq 1}$.

Let $T(n) = 2^{cn}$ for some constant $c \in \mathbb{N}_{\geq 1}$ be such that $L \in \mathsf{TIME}[T(n)]$. We define a machine M such that $M(x,y)$ always outputs $L(x)$ regardless of the second input y. Note that M runs in $T(n)$ time for $x \in \{0,1\}^n$. Let $r(n) \leq 2\log T(n)$ be the randomness parameter obtained by applying Lemma 3.11 to M. For simplicity, we can assume that $r(n) = 2\log T(n) = 2c \cdot n$ by ignoring the unused random bits.

Now we define a new language L' as follows: for every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, for every $w \in \{0,1\}^{r(n)}$, $L'(x,w) = 1$ if and only (1) $x \in L$ and (2) $E(x, 1^{T(n)})$ outputs a map $\pi\colon \{0,1\}^{r(n)}$ such that $\pi(w) = 1$.

Since E is computable in $\mathrm{poly}(T(n))$ time, $L' \in \mathsf{E}$ as well. From our assumption, we know that L' has $g(n)$-size $\mathsf{AC}^0_d[m]$ circuits for some $d, m \in \mathbb{N}$. In particular, it means that for every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, if $x \in L$, $E(x, 1^{T(n)})$ outputs a map $\pi\colon \{0,1\}^{r(n)} \to \{0,1\}$ that is computable by $g((2c+1)\cdot n)$-size $\mathsf{AC}^0_d[m]$ circuits.

Now, let $S(n) = g((2c+1)\cdot n) \leq \mathrm{poly}(g(n))$ (since g is a reasonable time-bound function). We define the following MA algorithm A_L for L:

1. Given an input $x \in \{0,1\}^n$ for some $n \in \mathbb{N}$.
2. A_L first applies Lemma 3.11 to compute an AC^0 oracle circuit V_x, such that:
   (a) If $x \in L$, then $E(x, 1^{T(n)})$ outputs a map $\pi\colon \{0,1\}^{r(n)} \to \{0,1\}$ such that
   $$\Pr_{w \in_R \{0,1\}^{r(n)}}[V^\pi_x(w) = 1] = 1.$$
   (b) If $x \notin L$, then for all oracles $\pi\colon \{0,1\}^{r(n)} \to \{0,1\}$, it holds that
   $$\Pr_{w \in_R \{0,1\}^{r(n)}}[V^\pi_x(w) = 1] \leq 1/3.$$
3. A_L guesses an $S(n)$-size $\mathsf{AC}^0_d[m]$ circuit $C\colon \{0,1\}^{2r(n)} \to \{0,1\}$, draws $w \in \{0,1\}^{r(n)}$, and outputs $V^C_x(w)$.

Now we verify that the algorithm above is an $\mathsf{MATIME}_{\mathsf{AC}^0_{d+O(1)}[m]}[\mathrm{poly}(g(n))]$ algorithm for L. First, for each guess C, we can see that the verifier's restriction on random inputs, $V^C_x(w)$, is an $\mathsf{AC}^0_{d+O(1)}[m]$ circuits of size at most $S(n) + \mathrm{poly}(n) \leq \mathrm{poly}(g(n))$, since V_x is an AC^0 oracle circuit and C is an $\mathsf{AC}^0_d[m]$ circuit. Next, for $x \in L$, by Condition (2.a) above and previous discussions, there exists an $S(n)$ size $\mathsf{AC}^0_d[m]$ circuit C such that $V^C_x(w) = 1$ for all w. When $x \notin L$, by Condition (2.b), $\Pr_w[V^C_x(w) = 1] \leq 1/3$. Therefore, A_L is an $\mathsf{MATIME}_{\mathsf{AC}^0_{d+O(1)}[m]}[\mathrm{poly}(g(n))]$ algorithm computing L, which completes the proof. ∎

Now we are ready to prove Theorem 1.5.

Proof of Theorem 1.5. Let $g(n)$ be a sub-half-exponential reasonable time-bound function. If E has no $g(n)$-size ACC^0 circuits, then clearly NE also has no $g(n)$-size ACC^0 circuits. Hence, we can assume that E has $g(n)$-size ACC^0 circuits from now on.

By Lemma 8.1, we know have

$$\mathsf{E} \subseteq \mathsf{MATIME}_{\mathsf{ACC}^0}[\mathrm{poly}(g(n))]. \tag{17}$$

By a standard diagonalization, there is language $L \in \mathsf{TIME}\left[2^{g(n)^2}\right]$ such that

$$L \notin \text{i.o.-}\mathsf{SIZE}[g(n)].$$

Now we use a padding argument to show $L \in \mathsf{MATIME}_{\mathsf{ACC}^0}[2^{n^{o(1)}}]$. Formally, we define another language L' so that $x \in L$ if and only

$$x \circ 0^{g(|x|)^2 - |x|} \in L'.$$

We can see that $L' \in \mathsf{E}$, and therefore $L' \in \mathsf{MATIME}_{\mathsf{ACC}^0}[\mathrm{poly}(g(n))]$ by (17). This further implies that

$$L \in \mathsf{MATIME}_{\mathsf{ACC}^0}[\mathrm{poly}(g(g(n)^2))].$$

From the assumption that $g(n)$ is sub-half exponential, it follows that $L \in \mathsf{MATIME}_{\mathsf{ACC}^0}[2^{n^{o(1)}}]$. Applying Corollary 4.8 completes the proof. ∎

Acknowledgments

The author would like to thank Hanlin Ren, Roei Tell, Nikhil Vyas, and Ryan Williams for helpful discussions.

References

[Aar16] Scott Aaronson. P =? NP. In John Forbes Nash Jr. and Michael Th. Rassias, editors, *Open Problems in Mathematics*, pages 1–122. Springer, 2016.

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

[ABO84] Miklos Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 471–474, 1984.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[Ajt83] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

[Ajt90] Miklós Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop, New Jersey, USA, December 3-7, 1990*, pages 1–20, 1990.

[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.

[BV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 163–173, 2014.

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

[CLW20] Lijie Chen, Xin Lyu, and Richard Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–12, 2020.

[Coh13] Gil Cohen. A taste of circuit complexity pivoted at NEXP $\not\subset$ ACC^0 (and more). In *Lecture Notes, Electronic Colloquium on Computational Complexity (ECCC)*. Citeseer, 2013.

[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, Proceedings*, pages 317–330, 2018.

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

[CR21] Lijie Chen and Hanlin Ren. Strong average-case circuit lower bounds from nontrivial derandomization. *SIAM Journal of Computing*, 51(3):STOC20–115, 2021.

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

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 273–301. Springer, 2011.

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

[GP18] Joshua A. Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system. *J. ACM*, 65(6):37:1–37:59, 2018.

[GSV18] Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 956–966, 2018.

[HAB02] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.

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

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Proc. 23rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 672–683, 2006.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 244–256, 2002.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[JMV15] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Proc. 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 749–760, 2015.

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

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4):859–868, 1992.

[MW18] Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: An easy witness lemma for NP and NQP. In *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 890–901, 2018.

[MW20] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM Journal of Computing*, 49(5), 2020.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[Raz87] Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematical Notes of the Academy of Science of the USSR*, 41(4):333–338, 1987.

[San09] Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. *SIAM Journal of Computing*, 39(3):1038–1061, 2009.

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

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[SM73] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time. In *Proc. 5th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–9, 1973.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 77–82, 1987.

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

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[Vio09] Emanuele Viola. On approximate majority and probabilistic time. *Computational Complexity*, 18(3):337–375, 2009.

[VL99] Jacobus Hendricus Van Lint. *Introduction to coding theory*, volume 86. Springer-Verlag Berlin Heidelberg, 1999.

[Vya19] Nikhil Vyas. Unpublished Manuscript, 2019.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 231–240, 2010.

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

[Wil13a] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal of Computing*, 42(3):1218–1244, 2013.

[Wil13b] Ryan Williams. Natural proofs versus derandomization. In *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2013.

[Wil14] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM*, 61(1):2:1–2:32, 2014.

[Wil16] R. Ryan Williams. Natural proofs versus derandomization. *SIAM Journal of Computing*, 45(2):497–529, 2016.

[Wil18a] Richard Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In *Proc. 33rd Annual IEEE Conference on Computational Complexity (CCC)*, pages 6:1–6:24, 2018.

[Wil18b] Richard Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory of Computing*, 14:Paper No. 17, 25, 2018.

[Yao85] Andrew C-C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1985.

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

A Witness Lower Bounds from Non-trivial Derandomization

Now we are ready to prove Lemma 2.6. The proof idea will be to use the assumed non-trivial CAPP algorithm to contradict a certain NTIME hierarchy theorem. For this purpose, we will need to construct a slightly faster nondeterministic algorithm for a language $L \in \mathsf{NTIME}[T]$, which is described in Algorithm A.1.

We summarize the important properties of Algorithm A.1 below.

Lemma A.1. *Let $T, L, \ell, K, S, \mathsf{VPCP}_x$ be stated as in Algorithm A.1. Under the assumption from Algorithm A.1, the following holds:*

1. APCP *runs in* $\mathrm{poly}(n, \log T) + T/(\log T)^{\omega(1)}$ *time and guesses* $\mathrm{poly}(S(\ell))$ *bits.*
2. *For every $x \in \{0,1\}^*$ such that $L(x) = 0$, it holds that $\mathsf{APCP}(x) = 0$.*[55]
3. *For every $x \in \{0,1\}^*$ such that $L(x) = 1$ and $\mathsf{APCP}(x) = 0$, it holds that*

[55] Since APCP is a nondeterministic algorithm, we use $\mathsf{APCP}(x) = 0$ to denote that APCP reject *all* guesses on the input x, and $\mathsf{APCP}(x) = 1$ to denote that APCP accepts *some* guesses on the input x.

Algorithm A.1: The algorithm APCP attempting to speed up L

Setting: Let $T(n)$ be a time-constructive function, and $L \in \mathsf{NTIME}[T(n)]$. Let $\ell(n) = \log T + O(\log \log T)$ be the number of random bits from Lemma 3.11 when the running time is set to $T(n)$. Let $K \in \mathbb{N}_{\geq 1}$ be a sufficiently large universal constant.

Parameters: Let $S(n)$ be a size parameter.

Assumption: CAPP for $n^K \cdot S(n)$-size $\mathsf{AC}^0_2 \circ \mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ time.

Input: $x \in \{0,1\}^n$

1 Apply Lemma 3.11 to L to obtain a $\mathrm{poly}(\ell)$-size AC^0_2 oracle circuit $\mathsf{VPCP}_x \colon \{0,1\}^\ell \to \{0,1\}$ that queries an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$;

2 **Guess** an $S(\ell)$-size ℓ-input $\mathscr{C}$ circuit C, and run the assumed algorithm for CAPP on VPCP^C_x to obtain an estimate $p \in [0,1]$;

`// Note that` VPCP^C_x `is an` $\ell^K \cdot S(\ell)$`-size` $\mathsf{AC}^0_2 \circ \mathscr{C}$ `from the complexity part of Lemma 3.11`

3 **if** $p > 1/2$ **then accept**;

4 **else reject**;

(a) *There exists* $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ *such that*

$$\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}^{\mathcal{O}}_x(r) = 1] = 1.$$

(b) *For every* $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ *satisfying the above,* $\mathcal{O}$ *does not have* $S(\ell)$*-size* $\mathscr{C}$ *circuits.*

Before proving Lemma A.1, we show it immediately imply Lemma 2.6. In fact, we will prove a more general version of it.

Theorem A.2 (General version of Lemma 2.4 and Lemma 2.6)**.** *Let* $\mathscr{C}$ *be a typical concrete circuit class,* $K \in \mathbb{N}_{\geq 1}$ *be a sufficiently large constant, and* $S(n)$ *be a size parameter. If* CAPP *for* $n^K \cdot S(n)$*-size* $\mathsf{AC}^0_2 \circ \mathscr{C}$ *circuits can be solved in* $2^n/n^{\omega(1)}$ *time, then unary* NE *does not admit* $S(n)$*-size* $\mathscr{C}$ *witnesses.*

Proof. Let $T(n) = 2^n$ and L be a unary language such that $L \in \mathsf{NTIME}[T(n)] \setminus \mathsf{NTIME}[T(n)/n]$, whose existence is guaranteed by the nondeterministic time hierarchy theorem [Žák83]. Now we consider APCP with T, L, and size parameter S, and set K to be the constant K in Algorithm A.1. Note that the assumption of Algorithm A.1 is satisfied with our choice of $S(n)$ and K.

By Item (1) of Lemma A.1, APCP runs in $\mathrm{poly}(n) + 2^n/n^{\omega(1)} < T(n)/n$ time, meaning that APCP cannot solve L. Hence, from the fact that L is a unary language and Item (2) of Lemma A.1, it follows that for infinitely many $n \in \mathbb{N}_{\geq 1}$, we have $L(1^n) = 1$ and yet $\mathsf{APCP}(1^n) = 0$. We say these n are good.

Now we are ready to define our verifier $V(x, y)$. Without loss of generality, we can assume that $\ell(n) = n + O(\log n)$ is an increasing function. For every $\alpha \in \mathbb{N}_{\geq 1}$, $V(1^\alpha, y)$ rejects immediately if there is no $n \in \mathbb{N}_{\geq 1}$ such that $\ell(n) = \alpha$. Otherwise, there is a unique $n \in \mathbb{N}_{\geq 1}$ such that $\ell(n) = \alpha$,

and $V(1^\alpha, y)$ accepts if and only if

$$\Pr_{r \in_R \{0,1\}^{\ell(n)}}[\mathsf{VPCP}_{1^n}^{\mathsf{func}(y)}(r) = 1] = 1.$$

Note that $V(x, y)$ runs in $2^{O(|x|)}$ time. Also, by Item (3) of Lemma A.1, for every good n, $V(1^{\ell(n)}, y)$ accepts some y, and all the accepted y have no $S(\ell(n))$-size $\mathscr{C}$ circuits. This implies that unary NE does not admit $S(n)$-size witnesses. ■

Finally, we are ready to prove Lemma A.1.

Proof of Lemma A.1. Item (1) follows from the fact that the running time is dominated by the complexity of applying Lemma 3.11 (which is $\mathrm{poly}(n, \log T)$) and running the assumed CAPP algorithm (which is $2^\ell/\ell^{\omega(1)} = T/(\log T)^{\omega(1)}$).

Let $x \in \{0,1\}^n$. To prove Item (2) and Item (3), note that from the completeness and soundness part of Lemma 3.11, we have:

(Completeness) If $L(x) = 1$, then there is an oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$ such that

$$\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^{\mathcal{O}}(r) = 1] = 1.$$

(Soundness) If $L(x) = 0$, then for all oracle $\mathcal{O} \colon \{0,1\}^\ell \to \{0,1\}$, it holds that

$$\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_n^{\mathcal{O}}(r) = 1] \leq 1/n^{10}.$$

To see Item (2), from the soundness condition above, we know that when $L(x) = 0$, for all guessed circuits C, the estimate of $\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^C(r) = 1]$ is at most $1/n^{10} + 1/3 < 1/2$ (recall that the error parameter of CAPP is set to $1/3$ by default). Hence $\mathsf{APCP}(x) = 0$ as well.

Now we turn to establish Item (3). Item (3.a) follows immediately from the completeness condition above. To see Item (3.b), suppose for the sake of contradiction that there exists an $S(\ell)$-size $\mathscr{C}$ circuit C such that $\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^C(r) = 1] = 1$. Then such C would be guessed by APCP on the input x and its estimate of $\Pr_{r \in_R \{0,1\}^\ell}[\mathsf{VPCP}_x^C(r) = 1]$ is at least $1 - 1/3 > 1/2$, meaning that $\mathsf{APCP}(x) = 1$. This contradicts the assumption of Item (3). ■

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