NON-DETERMINISTIC QUASI-POLYNOMIAL TIME IS AVERAGE-CASE HARD FOR ACC CIRCUITS

LIJIE CHEN*

Abstract. Following the seminal work of [Williams, J. ACM 2014], in a recent breakthrough, [Murray and Williams, STOC 2018] proved that NQP (non-deterministic quasi-polynomial time) does not have polynomial-size 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).

We strengthen the above lower bound to an average case one, by proving that for all constants c, there is a language in NQP that cannot be $(1/2 + 1/\log^c n)$-approximated by polynomial-size ACC^0 circuits. Our work also improves the average-case lower bound for NEXP against polynomial-size ACC^0 circuits by [Chen, Oliveira, and Santhanam, LATIN 2018].

Our new lower bound builds on several interesting components, including:

1. Barrington's theorem and the existence of an NC^1-complete language that is random self-reducible.
2. The sub-exponential witness-size lower bound for NE against ACC^0 and the conditional non-deterministic PRG construction in [Williams, SICOMP 2016].
3. An "almost" almost-everywhere MA average-case lower bound (which strengthens the corresponding worst-case lower bound in [Murray and Williams, STOC 2018]).
4. A PSPACE-complete language that is downward self-reducible, same-length checkable, error-correctable, and paddable. Moreover, all its reducibility properties have corresponding low-depth non-adaptive oracle circuits. Our construction builds on [Trevisan and Vadhan, Computational Complexity 2007].

Like other lower bounds proved via the "algorithmic approach", the only property of ACC^0 exploited by us is the existence of a non-trivial SAT algorithm for ACC^0 [Williams, J. ACM 2014]. Therefore, for any typical circuit class $\mathscr{C}$, our results apply to $\mathscr{C}$ as well if a non-trivial SAT (in fact, $\mathsf{Gap}\text{-}\mathsf{UNSAT}$) algorithm for $\mathscr{C}$ is discovered.

Key words. Average-Case Complexity, Circuit Lower Bounds, ACC Circuits

AMS subject classifications. 68Q05, 68Q17

1. Introduction.

1.1. Background and Motivation.

Establishing *unconditional* circuit lower bounds for explicit functions (with the ultimate goal of proving $\mathsf{NP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$) is one of the holy grails of theoretical computer science. Back in the 1980s, there was a lot of significant progress in proving circuit lower bounds for AC^0 (constant depth circuits consisting of AND/OR gates of unbounded fan-in) [2, 24, 68, 33] and $\mathsf{AC}^0[p]$ [49, 55] (AC^0 circuits extended with MOD_p gates) for a prime p. But this quick development was then met with an obstacle—there was little progress in understanding the power of $\mathsf{AC}^0[m]$ for a composite m. In fact, it was a long-standing open question in computational complexity whether NEXP (non-deterministic exponential time) has polynomial-size ACC^0 circuits[1], until a seminal work by Williams [66] from a decade ago, which proved NEXP does not have polynomial-size ACC^0 circuits, via a new *algorithmic* approach to circuit lower bounds [64].

This circuit lower bound has been an exciting new development after a long gap, especially since is believed to bypass all previous known barriers for proving circuits lower bounds: relativization [11], algebrization [1], and natural proofs [50]. Moreover, the underlying approach, the algorithmic method [64], puts many important classical complexity results together, ranging from non-deterministic time hierarchy

*Massachusetts Institute of Technology `lijieche@mit.edu`.

[1] It had been stressed several times as one of the most *embarrassing* open questions in complexity theory, see [7].

theorem [52, 69], $\mathsf{IP} = \mathsf{PSPACE}$ [43, 54], hardness vs randomness [47], to the PCP Theorem [8, 9].

While the circuit lower bound by Williams is a significant breakthrough after a long gap, it still has some drawbacks when comparing to the previous lower bounds. First, it only holds for the gigantic class NEXP, while our ultimate goal is to prove lower bound for a much smaller class NP. Second, it only proves a *worst-case* lower bound, while previous lower bounds and their subsequent extensions often also worked in the average-case; and it seems hard to adapt the algorithmic approach to the average-case settings.

Motivated by the above limitations, subsequent works extend the worst-case $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ lower bound in several ways. In 2012, by refining the connection between circuit analysis algorithms and circuit lower bounds, Williams [67] proved that $(\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$ does not have polynomial-size ACC^0 circuits. Two years later, by designing a fast #SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, Williams [65] proved that NEXP does not have polynomial-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. Then in 2017, building on [67], Chen, Oliveira and Santhanam [20] proved that NEXP cannot be $1/2 + 1/\operatorname{polylog}(n)$-approximated by polynomial-size ACC^0 circuits. Recently, in an exciting new breakthrough, with a new easy-witness lemma for NQP, Murray and Williams [46] proved that NQP does not have polynomial-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.[2]

1.2. Our Results. In this work, we strengthen all the above results by proving an average-case lower bound for NQP against $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

THEOREM 1.1. *For all $a, c > 0$, there is an integer b, such that $\mathsf{NTIME}[2^{\log^b n}]$ cannot be $(1/2 + 1/\log^c n)$-approximated by $2^{\log^a n}$ size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. The same holds for $(\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$ in place of $\mathsf{NTIME}[2^{\log^b n}]$*[3].

In other words, there is a language L in $\mathsf{NTIME}[2^{\log^b n}]$ that cannot be $(1/2 + 1/\log^c n)$-approximated by $2^{\log^a n}$ size $\mathsf{AC}_d[m] \circ \mathsf{THR}$ circuits, for *all* constants d, m (*i.e.*, the language L is fixed and its hardness is against any choice of d and m). We also remark that our new circuit lower bound builds crucially on another classical complexity gem: Barrington's theorem [12] together with a random self-reducible NC^1-complete language [10, 39].

Either $\mathsf{NQP} \not\subset \mathsf{P}_{/\operatorname{poly}}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$. MCSP is the *Minimum Circuit Size Problem* such that, given a truth-table $T\colon \{0,1\}^{2^n}$ and an integer $0 \le s \le 2^n$, asks whether there is a circuit C of size at most s that computes the function described by the truth-table T (see [5] and the references therein for more information on this problem).

Applying Theorem 1.1, we also resolve an open question from [30]. [30] proved (among many other results) that $\mathsf{MAJ} \in (\mathsf{AC}^0)^{\mathsf{MCSP}}$, and used that together with $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ [66] to prove that either $\mathsf{NEXP} \not\subset \mathsf{P}_{/\operatorname{poly}}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$. It is asked whether one can further show either $\mathsf{NQP} \not\subset \mathsf{P}_{/\operatorname{poly}}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$. We answer that affirmatively by proving the following corollary of Theorem 1.1.

COROLLARY 1.2. *Either $\mathsf{NQP} \not\subset \mathsf{P}_{/\operatorname{poly}}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$.*

See Appendix D for a proof of the above corollary.

[2] We also remark that [21] improved the dependence on depth by showing NEXP does not have ACC^0 circuits of $o(\log n/\log\log n)$ depth.

[3] See Definition 3.11 for a formal definition of $(\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[T(n)]_{/1}$.

From Modest-Improvement on Gap-UNSAT Algorithms to Average-Case Lower Bounds. Like other lower bounds proved via the "algorithmic approach" [64], the only property of $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits exploited by us is the non-trivial satisfiability algorithm for them [65]. Hence, our results also apply to other natural circuit classes if corresponding algorithms are discovered.

We say a circuit class $\mathscr{C}$ is *typical*, if it is closed under both negation and projection (see Subsection 3.1.1 for a formal definition). Also, we say a circuit class $\mathscr{C}$ is *nice*, if it is typical and either $\mathscr{C} = \mathsf{Circuit}$ or $\mathscr{C}$ is weaker than formula.[4]

We first define the Gap-UNSAT problem: given a circuit C, the goal is to distinguish between the case that C is unsatisfiable and the case that C has at least $1/3 \cdot 2^n$ satisfying assignments.[5] Next, we define the non-trivial derandomization condition below.

Definition 1.3 (Non-trivial derandomization condition). *For a typical circuit class $\mathscr{C}$, we say that the non-trivial derandomization condition holds for $\mathscr{C}$, if there is $\varepsilon \in (0, 1)$ such that the Gap-UNSAT problem for 2^{n^ε}-size n-input $\mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ non-deterministic time.*

The following theorem generalizes Theorem 1.1 to any nice circuit class $\mathscr{C}$ such that $\mathsf{AC}_0 \circ \mathscr{C}$ admits a non-trivial Gap-UNSAT algorithm.

Theorem 1.4. *Let $\mathscr{C}$ be a nice circuit class. Suppose the non-trivial derandomization condition holds for $\mathsf{AC}_7 \circ \mathscr{C}$. Then for every $a, c \in \mathbb{N}$, there is $b \in \mathbb{N}$, and a language $L \in \mathsf{NTIME}[2^{\log^b n}]$ such that L cannot be $(1/2 + 1/\log^c n)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits. The same holds for $\mathsf{(N{\cap}coN)TIME}[2^{\log^b n}]_{/1}$ in place of $\mathsf{NTIME}[2^{\log^b n}]$.*

Moreover, our average-case lower bounds can be significantly strengthened if $\mathsf{AC}_0 \circ \mathsf{MAJ} \circ \mathscr{C}$ admits a non-trivial Gap-UNSAT algorithm.

Theorem 1.5. *Let $\mathscr{C}$ be a nice circuit class. Suppose the non-trivial derandomization condition holds for $\mathsf{AC}_5 \circ \mathsf{MAJ} \circ \mathscr{C}$. Then for every $a, c \in \mathbb{N}$, there is $b \in \mathbb{N}$, and a language $L \in \mathsf{NTIME}[2^{\log^b n}]$ such that L cannot be $(1/2 + 1/2^{\log^c n})$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits. The same holds for $\mathsf{(N{\cap}coN)TIME}[2^{\log^b n}]_{/1}$ in place of $\mathsf{NTIME}[2^{\log^b n}]$.*

In particular, for $\mathscr{C} \in \{\mathsf{TC}_0, \mathsf{Formula}, \mathsf{Circuit}\}$, a non-trivial Gap-UNSAT algorithm for $\mathscr{C}$ circuits implies that NQP is *strongly* average-case hard against $\mathscr{C}$. Hence, Theorem 1.4 and Theorem 1.5 essentially strengthen the similar algorithms-to-circuit-lower-bounds connections in [46] from worst-case lower bounds for NQP to average-case lower bounds for NQP.

We remark that our connection *does not* go through an easy-witness lemma, since it is not clear how can one get an average-case easy witness lemma (*i.e.*, a statement asserting that if NQP can be approximated by $\mathsf{P}_{/\mathrm{poly}}$, then all NQP verifiers have succinct witnesses). Rather, we use a different approach similar to [67] and prove the average case lower bound *directly*, without going through the easy-witness lemma.

[4] Recall that a circuit class $\mathscr{C}$ is weaker than Formula, if there is polynomial p such that every s-size $\mathscr{C}$ circuit has an equivalent $p(s)$-size formula. We note that most well-studied circuit classes $(\mathsf{AC}^0, \mathsf{ACC}^0, \mathsf{TC}^0, \mathsf{Formula}, \mathsf{Circuit})$ are nice.

[5] This problem is weaker than both the SAT problem and the CAPP problem. In the CAPP problem, one is given a circuit C and the goal is to approximate the acceptance probability of C over random assignments, within a constant additive error.

A Simpler Proof for the New Easy Witness Lemma for NP and NQP of [46]. As an interesting by-product of our new ideas, we give a simpler proof for new easy-witness lemma for NP and NQP of [46] (Lemma 1.6 and Lemma 1.7). The proof from [46] crucially depends on a certain "bootstrapping" argument ([46, Lemma 3.1]), while we provide a more direct and simpler proof without involving that bootstrapping. We believe that this new proof is an independent contribution of this work.

LEMMA 1.6 (Easy-witness lemma for NP, Lemma 1.2 of [46]). *For all $k \in \mathbb{N}$, there is $b \in \mathbb{N}$ such that if $\mathsf{NP} \subset \mathsf{SIZE}[n^k]$, then every $L \in \mathsf{NP}$ has witness circuits*[6] *of size at most n^b.*[7]

LEMMA 1.7 (Easy-witness lemma for NQP, Lemma 1.3 of [46]). *For all $k \in \mathbb{N}$, there is $b \in \mathbb{N}$ such that if $\mathsf{NQP} \subset \mathsf{SIZE}[2^{\log^k n}]$, then every $L \in \mathsf{NQP}$ has witness circuits of size at most $2^{\log^b n}$.*

Subsequent Work. In the conference version of this paper [17], two open questions was raised, and were (essentially) resolved by subsequently work. The first open question was whether the algorithmic approach can be used to construct rigid matrices (*i.e.*, proving average-case lower bounds against low-rank matrices), this was later answered in the affirmative by Alman and the author [6], whose results was then significantly simplified and strengthened by Bhangale, Harsha, Paradise, and Tal [14].

The second open question was whether we can strengthen Theorem 1.1 to that NQP cannot be $(1/2 + 1/n^{\omega(1)})$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$. Such a strengthening was later proved by the author and Ren [19]. In another follow-up work, the author, Lyu, and Williams [18] proved that there is a function $f \in \mathsf{E}^{\mathsf{NP}}$ that cannot be $(1/2+2^{-n^{o(1)}})$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits of $2^{n^{o(1)}}$-size, for all large enough input lengths n. The result of [18] is incomparable to [19], since it established a much harder function against $\mathsf{ACC}^0 \circ \mathsf{THR}$ but also in a much larger complexity class.

Moreover, one important technical ingredient of this paper, a new PSPACE-complete language with several useful properties (see Theorem 3.7), is also proven to be useful in the construction of better pseudodeterministic PRGs by Lu, Oliveira, and Santhanam [42].

2. Technique Overview.

In the following we discuss the intuition behind our new average-case lower bounds. For simplicity of argument, we will sketch a proof for NQP cannot be $(1-\delta)$-approximated by polynomial-size ACC^0 circuits, for a universal constant δ (δ can be thought of as $1/1000$).

2.1. Main Difficulty: The Absence of an Easy-Witness Lemma Under the Approximability Assumption.

First, it is instructive to see why it is hard to generalize the previous proofs for worst-case lower bound against ACC^0 [66, 46] to prove an average-case lower bound against ACC^0.

The first step of the $\mathsf{NQP} \not\subset \mathsf{ACC}^0$ lower bound by Murray and Williams [46], is applying the so called *easy witness lemma*. The easy witness lemma states: assuming $\mathsf{NQP} \subset \mathsf{ACC}^0$, for every language L in NQP with a verifier $V(x, y)$, whenever $V(x, \cdot)$ is satisfiable, it has a succinct witness y that is the truth-table of a small ACC^0 circuit. Then they apply a proof by contradiction[8]: assuming $\mathsf{NQP} \subseteq \mathsf{ACC}^0$, they use

[6] See Definition 3.13 for a formal definition.

[7] To simplify the presentation, we do not specify the relations between b and k here, but we nonetheless remark that one can take $b = \Theta(k^3)$, just as in [46].

[8] A similar argument is also used in [64, 66].

the existence of easy-witness circuits (implied by the easy-witness lemma) together with the non-trivial SAT algorithm for ACC^0 circuits in [66] to contradict the *non-deterministic* time hierarchy theorem [69].

Now for proving the average-case lower bound for NQP, we can only start with the assumption that NQP can be $(1-\delta)$-approximated by polynomial-size ACC^0 circuits (and hope to contradict the non-deterministic time hierarchy theorem). As already explained by [20], we cannot apply the easy witness lemma even if we start from the much stronger assumption that NEXP can be $(1-\delta)$-approximated by ACC^0: the proofs of both the original and the new easy-witness lemma [37, 46] completely break when we only have the approximability assumption.

2.1.1. Review of [20]'s Approach. To get around the difficulty above, [20] start from a worst-case lower bound against ACC^0 [67], and then apply a worst-case to average-case *hardness amplification*. Their approach works roughly as follows:

1. By [67], there is a language $L \in (\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$ that does not have $\mathrm{poly}(n)$-size ACC^0 circuits.
2. Using the locally list decodable codes of [29, 32], one can define a language $\widetilde{L} \in (\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$ that cannot be $(1/2+1/\log n)$-approximated by $\mathrm{poly}(n)$ size ACC^0 circuits. That is, we treat the truth-table of L_n as a message $z \in \{0,1\}^{2^n}$ of the locally-list-decodable code, and set $\widetilde{L}_m$ so that its truth-table is the codeword of z for an appropriate input length $m = m(n)$. (Note that here it is important to work with a language L in $(\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$, as otherwise we do not know how to compute the truth-table of L in NEXP.)
3. In particular, the above $\widetilde{L} \in \mathsf{NEXP}_{/1}$. They then get rid of the advice bit by an enumeration trick, and therefore prove the average case lower bound for NEXP.

Unfortunately, it seems very hard to generalize the approach above to prove an average-case lower bound for NQP: the second step of the approach above breaks, as we no longer can afford to compute an error correcting code on the entire truth-table of a particular input length, which takes (at least) exponential time.

Therefore, we have to take a different approach that proves the average-case lower bound *directly*, without going through the worst-case to average-case hardness amplification. In order to do that, it is helpful to review the proof of the new easy-witness lemma in [46].

2.2. Easy-Witness Lemma for NQP: "Almost" Almost-Everywhere (a.a.e.) MA Lower Bounds and i.o. Non-deterministic PRGs (NPRGs). (An instantiation of) the new easy-witness lemma of [46] states that if $\mathsf{NQP} \subset \mathsf{P}_{/\mathrm{poly}}$, then all verifiers for NQP languages have succinct (polynomial-size) witness (Lemma 1.7). For the sake of contradiction, we now suppose that $\mathsf{NQP} \subset \mathsf{P}_{/\mathrm{poly}}$ and some verifier for a language $L \in \mathsf{NQP}$ does not have $\mathrm{poly}(n)$-size witness circuits. That is, there is a polynomial-time verifier $V(x,y)$ with $|x| = n$ and $y = 2^{\log^b n}$ for some $b \in \mathbb{N}$, such that for infinitely many n's, there is an $x_n \in \{0,1\}^n$ such that $V(x_n,\cdot)$ is satisfiable, but for any y_n such that $V(x_n,y_n)=1$, we have $\mathsf{SIZE}(y_n) = n^{\omega(1)}$.

Now, y_n can be interpreted as a truth-table of a function on $\ell = \log^b n$ variables, and we have $\mathsf{SIZE}(y_n) \ge 2^{\omega(\ell^{1/b})}$. Therefore, given such a y_n, using the well-known hardness-to-pseudorandomness connection (see, *e.g.*, [47, 38, 57, 60]), one can construct a pseudorandom generator G_{y_n} with seed length $O(\ell)$, running time $2^{O(\ell)}$, and it fools all circuits of size $2^{a\cdot \ell^{1/b}}$, for all constants a.

Scaling everything properly by setting $S = 2^{a\cdot \ell^{1/b}}$, it follows that for an infinite number of S, if we are given the x_n (of length $|x_n| = S^{1/a}$) as advice, we can guess a y_n such that $V(x_n, y_n) = 1$, and compute the PRG G_{y_n}. Hence, for every $a \geq 1$, there is a non-deterministic PRG that has seed length $O(\log^b S)$, running time $2^{O(\log^b S)}$, and fools all S-size circuits, and takes $S^{1/a}$ bits as advice. (See Subsection 3.2 for a formal definition of NPRGs.)

The key ingredient of [46] is an "almost" almost-everywhere (a.a.e.) MA circuit lower bound, which builds on the MA circuit lower bound by Santhanam [51].[9] For the simplicity of arguments, we now pretend that we have an almost-everywhere MA circuit lower bound. Specifically, for each $c \in \mathbb{N}$, there is an integer $k = k(c)$ and a language $L^c \in \mathsf{MATIME}[n^k]$ such that $\mathsf{SIZE}(L^c_n) \geq n^c$ for all sufficiently large n.

The crucial idea is that, using the above i.o. NPRG, one can non-deterministically derandomize L^c on an infinite number of input length n's (as the string y_n can be non-deterministically guessed-and-verified). To derandomize $\mathsf{MATIME}[n^k]$, it suffices to use the PRG that fools circuits of size $S = n^{2k}$. Therefore, by setting $a = 2k$, we have a language $L^\star \in \mathsf{NTIME}[2^{O(\log^{b+1} n)}]_{/n}$,[10] such that it agrees with L^c on infinitely many input lengths. Since c can be an arbitrary integer, we conclude that $\mathsf{NTIME}[2^{O(\log^{b+1} n)}]_{/n}$ is not in $\mathsf{P}_{/\,\mathrm{poly}}$. Thus, we obtain a contradiction to our assumption (the n bits of advice can be got rid of easily).[11]

Digest. To summarize, the proof of new easy witness lemma constructed i.o. NPRGs from the assumed non-existence of easy witness-circuits, and combined i.o. NPRG together with a.a.e. MA lower bounds to prove $\mathsf{NQP} \not\subseteq \mathsf{P}_{/\,\mathrm{poly}}$, a contradiction to the assumption that $\mathsf{NQP} \subseteq \mathsf{P}_{/\,\mathrm{poly}}$. Therefore, NQP must have easy witness-circuits assuming $\mathsf{NQP} \subseteq \mathsf{P}_{/\,\mathrm{poly}}$.

2.3. Our New Approach: "Almost" Almost-Everywhere Average-Case MA Lower Bound and i.o. NPRG. As mentioned before, we do not attempt to prove an average-case version of easy-witness lemma (and we do not know how to prove such an analogue). Instead, we will directly construct suitable i.o. NPRGs under the assumption that NQP is average-case easy for ACC^0, and combine that with an appropriate average-case hard language in MA. Derandomization of this average-case hard MA languages means that NQP is average-case hard for ACC^0, a contradiction to the assumption that NQP is average-case easy for ACC^0.

Nevertheless, the detailed implementation of the plan above is quite challenging, and we will give an outline below.

(i.o. NPRGs) Under the assumption that NQP can be approximated by ACC^0, we construct an i.o. NPRG fooling *low-depth* circuits.[12]

(New a.a.e. MA lower bounds) To complement the above new NPRG, we prove that there is a hard language $L \in \mathsf{MAQP} = \mathsf{MATIME}[2^{\mathrm{polylog}(n)}]$ such that:

1. L is average-case hard against *low-depth* circuits, and
2. L can be derandomized using an i.o. NPRG into NQP, while retaining its average-case hardness infinitely often.

[9] [46, 51]'s lower bounds are actually for MA with advice bits. We ignore the advice bits issue for the sake of simplicity in the intuition part. See the end of the this section for some discussions on how to deal with the advice bits.

[10] By choosing $a = 2k$, the seed length of the NPRG is bounded by $S^{1/2k} = n$, hence $L^\star$ only needs n bits of advice.

[11] Given $L \in \mathsf{NTIME}[2^{O(\log^{b+1} n)}]_{/n}$ that is not in $\mathsf{P}_{/\,\mathrm{poly}}$, one can define another language $L' \in \mathsf{NTIME}[2^{O(\log^{b+1} n)}]$ such that on inputs x of length $2n$, L' simulates L on $x_{\leq n}$ (the first half of x) with advice being set to $x_{>n}$ (the second half of x). It is easy to see that L' is not in $\mathsf{P}_{/\,\mathrm{poly}}$ as well.

[12] We informally use low-depth circuits to mean circuit with $\mathrm{polylog}(n)$ depth.

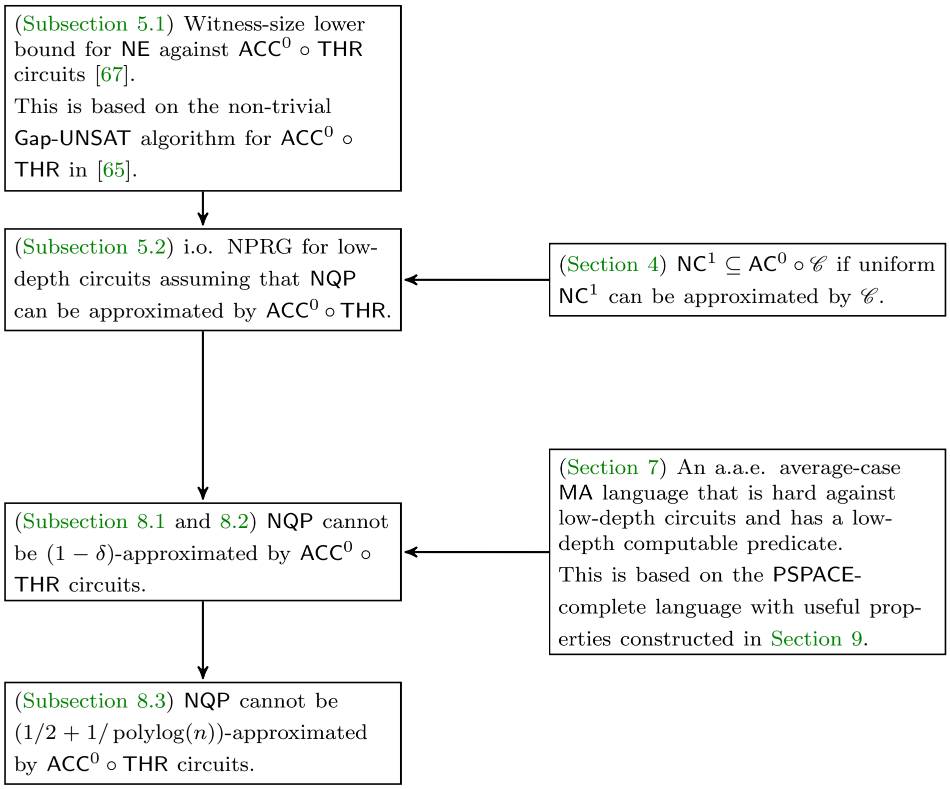

FIG. 1. *The structure of the whole argument.*

Crucially, the combination of the two components above is enough to conclude NQP cannot be $(1-\delta)$-approximated by ACC^0 circuits: assuming otherwise that NQP can be $(1-\delta)$-approximated by ACC^0, we can construct an i.o. NPRG G fooling low-depth circuits. Next, we use G to derandomize the MAQP average-case hard language L into NQP, this implies that NQP cannot be approximated by low-depth circuits, a contradiction to our assumption that NQP can be approximated by ACC^0. See Subsection 8.1 for details on how average-case lower bounds follow from i.o. NPRGs and new a.a.e. MA lower bounds.

Outline of Subsection 2.4 and Subsection 2.5. In Subsection 2.4, we will first explain why we can only get an i.o. NPRG fooling *low-depth* circuits (instead of general circuits as in [46]) from the assumption that NQP can be approximated by ACC^0. Then we will explain the technical challenges we have overcome in order to get such an i.o. NPRG. In Subsection 2.5, we explain how to prove the desired average-case MA lower bound. This part is more technical and contains several steps, see Subsection 2.5 for details.

See also Figure 1 for the structure of the whole argument.

2.4. i.o. Non-deterministic PRG. Williams [67] proved that assuming $\mathsf{P} \subseteq \mathsf{ACC}^0$, one can get an i.o. NPRG with $\operatorname{polylog}(n)$ seed length fooling poly-size *general circuits*. In this section, we will first explain why the construction of [67] does not directly work in our setting[13]. We then show that by lowering our goal to constructing an NPRG only for low-depth circuits, we can construct such NPRGs using the fact that NC^1 has a random self-reducible complete-problem.

2.4.1. The i.o. NPRG construction in [67]. The starting point in [67] is the (unconditional) witness-size lower bound for NE against ACC^0. That is, [67] proved that there is *unary* language in NE, whose verifier does not have 2^{n^ε}-size $\mathsf{AC}_{d_\star}[m_\star]$ witness ($\varepsilon = \varepsilon(d_\star, m_\star)$). Therefore, let the verifier be $V(x,y)$ with $|x| = n$ and $|y| = 2^n$; for infinitely many n, $V(1^n, \cdot)$ is satisfiable, yet for all y such that $V(1^n, y) = 1$, y is not the truth-table of a 2^{n^ε}-size $\mathsf{AC}_{d_\star}[m_\star]$ circuit.

Further assuming $\mathsf{P} \subset \mathsf{ACC}^0$, [67] showed that the above implies an i.o. NPRG for general circuits. Note that $\mathsf{P} \subset \mathsf{ACC}^0$ implies that the Circuit-Evaluation problem has an ACC^0 circuit, and consequently $\mathsf{P}_{/\,\mathrm{poly}}$ collapses to ACC^0. Therefore, for a y such that $V(1^n, y) = 1$, y cannot be computed by 2^{n^ε}-size general circuits as well, which means one can substitute y into the known hardness-to-pseudorandomness construction of [47, 60], and get a quasi-polynomial time i.o. NPRG.

However, starting with our assumption that NQP can be $(1-\delta)$-approximated by ACC^0, it is not clear how to show that $\mathsf{P}_{/\,\mathrm{poly}}$ collapses to ACC^0. So we have to take a more sophisticated approach. To make the situation worse, performing worst-case to average-case hardness amplification requires majority [53, 31][14]. Since it is not clear whether ACC^0 can compute majority, we do not even know how to get a PRG fooling ACC^0 circuits, from a truth-table y that is only worst-case hard against ACC^0.

2.4.2. i.o. Non-deterministic PRG for Low-Depth Circuits. So we wish to verify a truth-table y that is hard against a stronger circuit class, for which at least hardness amplification is possible, like NC^1. By an argument similar to that of [67], if NC^1 collapses to ACC^0, then the verifier V that verifies hard-truth tables for ACC^0 also verifies truth-tables that cannot be computed by low-depth circuits.

In more details, from $\mathsf{NC}^1 \subseteq \mathsf{ACC}^0$, there are $d_\star, m_\star \in \mathbb{N}$ such that any depth-d circuit has an equivalent $2^{O(d)}$-size $\mathsf{AC}_{d_\star}[m_\star]$ circuit. Now, get back to the verifier V. It follows that for an infinite number of n's, $V(1^n, \cdot)$ is satisfiable and for any y such that $V(1^n, y) = 1$, y is not the truth-table of an n^ε-depth circuit. This is enough to obtain a quasi-polynomial time i.o. NPRG that fools polylog(n)-depth circuits (see Theorem 3.3 for details).

So our goal now is to show that NC^1 collapses to ACC^0 under the assumption that NQP can be $(1-\delta)$-approximated by ACC^0. We call such a statement a *collapse theorem for* NC^1. Fortunately, we are able to prove such a collapse theorem using the existence of an NC^1-complete problem that admits a nice random self-reduction [12, 10, 39]. By our assumption, this problem can be $(1-\delta)$-approximated by ACC^0 circuits. Utilizing its random self-reduction and the fact that approximate-majority can be computed in AC^0 [2, 62], we can show that this NC^1-complete problem has polynomial-size ACC^0 circuits. This in particular means that NC^1 collapses to ACC^0.

The above construction of i.o. NPRG for low-depth circuits is detailed in Section 4 (where we prove the collapse theorem for NC^1) and Section 5 (where we construct the conditional i.o NPRGs).

[13] We can only assume that NQP is average-case easy for ACC^0, from which it is not clear how to derive $\mathsf{P} \subseteq \mathsf{ACC}^0$.

[14] That is, to get average-case lower bounds against $\mathscr{C}$ circuits using hardness amplification, one needs to start from worst-case lower bounds against $\mathsf{MAJ} \circ \mathscr{C}$ circuits.

2.5. An A.a.e. Average-Case MA Lower Bound. Next we explain how do we prove a suitable average-case MA lower bound that can be derandomized by our i.o. NPRGs fooling low-depth circuits.

2.5.1. Our MA Language Needs a Low-Depth Computable Predicate. We first note that in order to non-deterministically derandomize a general MA algorithm (*i.e.*, put it into NQP), a PRG for polylog(n)-depth circuits is not enough. Suppose the MA algorithm A takes an input x, guesses a witness string y, and flips some random coins r; in order to obtain a non-deterministic simulation, we would need to fool circuits $C_y(r) \coloneqq \mathcal{P}_A(x,y,r)$ for all possible y. Here, $\mathcal{P}_A(x,y,r)$ is called the predicate of the MA algorithm. Since there is no restriction on $\mathcal{P}_A$ other than a bound on its running time, the circuit C_y could well be a general circuit that does not necessarily have low depth.

The above difficulty brings us to our key component—an MA language L^{hard} that has a low-depth computable predicate, and is average-case hard against low-depth circuits. Now, since $\mathcal{P}_A(x,y,r)$ has a low-depth circuit, it follows that $C_y(r) \coloneqq \mathcal{P}_A(x,y,r)$ also has a *low-depth* circuit, and therefore our i.o. NPRG can be used to achieve an i.o. derandomization of L^{hard}, which results in a contradiction to our assumption.

2.5.2. A.a.e. Average-case MA Lower Bounds from a PSPACE-complete Language with Nice Properties. Roughly speaking, the MA circuit lower bounds in [51] and [46] make crucial use of a PSPACE-complete language by [59], which admits several nice properties, including being same-length checkable, downward self-reducible, and paddable (see Definition 3.4 for details). We modify the construction from [59] to obtain a PSPACE-complete language L^{PSPACE} that is also *error correctable*: that is, if it is hard in the worst-case, then it is also hard in the average-case. We think this new language L^{PSPACE} is of independent interest and may be useful for other problems.

The construction of such an average-case hard MA language is the technical centerpiece of this paper; the key observation is that all the nice properties of our PSPACE-complete problem L^{PSPACE} (*i.e.*, being same-length checkable, downward self-reducible, and paddable) have low-depth uniform oracle circuits. For instance, the instance checker in the same-length checkable property (see Definition 3.4), can actually be implemented by a uniform TC^0 *non-adaptive* oracle circuit. Using the existence of those oracle circuits, together with a careful case-analysis similar to previous work [51, 46], and some additional new ideas, we are able to construct the desired average-case hard MA language.

The PSPACE-complete language L^{PSPACE} is constructed in Section 9, and the a.a.e. average-case MA lower bounds are proved in Section 7.

2.5.3. A Technicality: Dealing with Advice Bits. In the above discussion, we (intentionally) omitted a technical detail—the a.a.e. MA lower bound proved in [46] is actually for $\mathsf{MA}_{/O(\log n)}$. Therefore our i.o. derandomization of the $\mathsf{MA}_{/O(\log n)}$ algorithm also needs $O(\log n)$ advice bits. But then, we only have that $\mathsf{NQP}_{/O(\log n)}$ is average-case hard for polynomial-size ACC^0 circuits. And the enumeration trick from [20] requires the advice to be $o(\log n)$.

Luckily, we further relax the definition of an "almost" almost-everywhere circuit lower bound in [46]. Our relaxation is weak enough for us to prove the required MA average-case lower bound with only *one* bit of advice, but also strong enough to allow us to prove the average-case circuit lower bound for $\mathsf{NQP}_{/O(1)}$. Then we can apply the enumeration trick from [20] to get the desired lower bound for NQP without advice.

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 U, we often use $x \in_{\mathsf{R}} U$ to denote that we pick an element x from U uniformly at random.

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 $\mathsf{tt}(f)$ to denote the truth-table of f (*i.e.*, $\mathsf{tt}(f)$ is a string of length 2^n such that $\mathsf{tt}(f)_i$ is the output of f on the i-th string from $\{0,1\}^n$ in the lexicographical order). For a string $Z\colon \{0,1\}^{2^n}$, 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 also use $f \circ g$ to denote the composition of two functions f and g. The meaning of the symbol $\circ$ (concatenation or composition) will always be clear from the context. 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.

3.1. Complexity Classes and Basic Definitions. We assume knowledge of basic complexity theory (see [7, 26] for excellent references on this subject).

3.1.1. Basic Circuit Families. Unless otherwise specified, the circuits appear in this paper are general circuits consisting of fan-in 2 AND/OR gates and fan-in 1 NOT gates.

A *circuit family* is a collection 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 *gates* in the circuit, and the size of a circuit family is a function of the input length that upper-bounds the size of circuits in the family. The *depth* of a circuit is the maximum number of wires on a path from an input gate to the output gate.

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

AC^0 is the class of circuit families of constant depth and polynomial size, with AND, OR and NOT gates, where AND and OR gates have unbounded fan-in. For an integer m, the function $\mathsf{MOD}_m\colon \{0,1\}^* \to \{0,1\}$ 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 circuit families consisting of polynomially-many unbounded fan-in AND, OR and MOD_m gates, along with unary NOT gates. We denote $\mathsf{ACC}^0 = \cup_{m\geq 2}\mathsf{AC}^0[m]$. We also use AC_d (resp. $\mathsf{AC}_d[m]$) 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 poly-size circuit families consisting of fan-in two AND and OR gates and unary NOT gates.

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 size of C_n, and outputs C_n.[15]

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

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.

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

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

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

For two circuit class $\mathscr{C}_1$ and $\mathscr{C}_2$, we say that $\mathscr{C}_1$ is weaker than $\mathscr{C}_2$, if there is polynomial p such that every s-size $\mathscr{C}$ circuit has an equivalent $p(s)$-size $\mathscr{C}$ circuits.

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

LEMMA 3.1 ([4, 3, 62]). *$\mathsf{Approx\text{-}MAJ}_n$ can be computed by uniform AC_3.*

3.1.2. Notation. For an approximation parameter $\gamma > 1/2$, 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 a γ fraction of inputs from $\{0,1\}^n$. For a function $f \colon \{0,1\}^n \to \{0,1\}$, we define $\mathsf{SIZE}(f)$ (resp. $\mathsf{DEPTH}(f)$) to be the minimum size (resp. depth) of a circuit computing f exactly. Similarly, for $\gamma > 1/2$, we define $\mathsf{Avg}_\gamma\text{-}\mathsf{SIZE}(f)$ (resp. $\mathsf{Avg}_\gamma\text{-}\mathsf{DEPTH}(f)$) to be the minimum size (resp. depth) of a circuit that γ-approximates f.

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

In other words, if a class of languages $\mathscr{L}$ cannot be $\gamma(n)$-approximated by $\mathscr{C}$, it means there exists a language $L \in \mathscr{L}$ such that, for every $\{C_n\}_{n\in\mathbb{N}} \in \mathscr{C}$, there are infinitely many n's such that C_n does not $\gamma(n)$-approximate L_n.

3.2. Pseudorandom Generators. We will deal with different types of pseudorandom generators (PRG) throughout the paper. In the following, we recall their definitions.

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 $D \in \mathcal{H}$

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

We also call r the *seed length* of G.

We also need the notion of *non-deterministic PRGs*, which is defined as 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 NPRG in general does not compute *the same PRG* for different witness u (*i.e.*, $G_{\mathsf{P}}(u_1, \cdot)$ and $G_{\mathsf{P}}(u_2, \cdot)$ can be two different PRG for $\mathcal{H}$), it is still useful for many tasks such as the derandomization of MA. The concept of NPRG is implicit in [37].

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 $r\colon \mathbb{N} \to \mathbb{N}$, 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). We also say G is an i.o. PRG (resp. i.o. NPRG) family for $\mathcal{H}$ if the above two conditions hold for infinitely many n instead of every n. When the meaning is clear, sometimes we just say G is a PRG (resp. NPRG) instead of a PRG (resp. NPRG) family.

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 $A_{\mathsf{W}}\colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}$ such that $(A_{\mathsf{P}})_n$ computes $(G_{\mathsf{P}})_n$ and $(A_{\mathsf{W}})_n$ computes $(G_{\mathsf{W}})_n$, 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 by default the running time T.

We will need the following PRG construction from [60].

THEOREM 3.2 ([60]). *There is a universal constant $c \in \mathbb{N}_{\ge 1}$ and an algorithm G such that:*

1. *G takes two integers ℓ and m such that $\ell \le m \le 2^{\ell/c}$, together with two strings $u \in \{0,1\}^{2^\ell}$ and $z \in \{0,1\}^{c\ell}$ as inputs, and outputs an m-bit string. G is also computable in $2^{O(\ell)}$ time.*
2. *If $f \in \mathcal{F}_{\ell,1}$ does not have S-size circuits for $S \ge m^c$, then $G_{\ell,m}(\mathsf{tt}(f), \cdot)$[16] is a PRG for $S^{1/c}$-size m-input circuits with error $1/m$ and seed length $c\ell$.*

PRGs for low-depth circuits. The following PRG construction follows directly from the local-list-decodable codes with low-depth decoder of [36, 29, 32], and the hardness-to-pseudorandomness transformation of [47].

THEOREM 3.3. *Let $\delta \in (0,1)$ be a constant. There are universal constants $c \in (0,1)$ and $g > 1$, and an algorithm G such that:*

1. *G takes two integers ℓ and m such that $\ell \le m \le 2^{\ell^{c\delta}}$, together with two strings $u \in \{0,1\}^{2^\ell}$ and $z \in \{0,1\}^{\ell^g}$ as inputs, and outputs an m-bit string. G is also computable in $2^{O(\ell)}$ time.*
2. *For every large enough $\ell \in \mathbb{N}$, if $f \in \mathcal{F}_{\ell,1}$ does not have ℓ^δ-depth circuits, then $G_{\ell,m}(\mathsf{tt}(f), \cdot)$ is a PRG for $\ell^{c\delta}$-depth m-input circuits with error $1/m$ and seed length ℓ^g.*

[16] For notational convenience, we use $G_{\ell,m}$ to denote that the first two inputs of G are fixed to ℓ and m.

We provide a proof for the above theorem in Appendix C for completeness.

3.3. A PSPACE-complete Language with Low-complexity Reducibility Properties. A fundamental result often used in complexity theory is the existence of a PSPACE-complete language [59] that is based on the protocol underlying the IP = PSPACE proof of [43, 54], and satisfies strong reducibility properties. This PSPACE-complete language has found applications in the time-hierarchy theorem for BPP with one bit of advice [23], the fixed polynomial circuit lower bound $\mathsf{MA}_{/1} \subseteq \mathsf{SIZE}(n^k)$ for any k [51], and the recent easy witness lemmas for NQP and NP [46].

The key technical ingredient of our new average-case lower bounds is a modified construction of the PSPACE-complete language in [59]. Our new construction satisfies the additional property of being error correctable[17] (see Definition 3.4 for the precise definitions), which is useful for proving average-case lower bounds. Moreover, we prove that the reductions in these reducibility properties of our PSPACE-complete languages can be implemented by low-depth circuits classes. We believe this new construction would be of independent interest, and may be useful for resolving other open questions in complexity theory.

We first define these reducibility properties.

Definition 3.4. *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 $x \in \{0,1\}^n$, we have $x \in L$ if and only if $\mathsf{Pad}(x, 1^m) \in L$, where $\mathsf{Pad}(x, 1^m)$ always has length m.*
3. *L is $\mathscr{C}$* weakly error correctable*, if there is a constant c such that for all sufficiently large n, for every oracle $O\colon \{0,1\}^n \to \{0,1\}$ that 0.99-approximates L_n, there is an n^c-size $\mathscr{C}$ oracle circuit D, such that D^O computes L_n exactly.*
4. *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 $|x|$.*
   (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.*

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

Remark 3.5. *The paddable property implies that $\mathsf{SIZE}(L_n)$ and $\mathsf{DEPTH}(L_n)$ are non-decreasing.*

The following PSPACE-complete language is given by [51] (modifying a construction of Trevisan and Vadhan [59]).

[17] The error correctable property here is stronger than the piecewise random self-reducible property in [51].

THEOREM 3.6 ([59, 51]). *There is a PSPACE-complete language L^{TV} that is paddable, TC^0 downward self-reducible, and same-length checkable.*[18]

Based on the above language L^{TV}, we construct a modified PSPACE-complete language L^{PSPACE} that is also NC^3 weakly error correctable. Moreover, with a careful analysis, we prove that the instance checker for L^{PSPACE} can be implemented by uniform randomized non-adaptive TC^0 oracle circuits. That is, L^{PSPACE} is non-adaptive TC^0 same length checkable.

THEOREM 3.7. *There is a PSPACE-complete language L^{PSPACE} that is paddable, non-adaptive TC^0 downward self-reducible, non-adaptive TC^0 same-length checkable, and non-adaptive NC^3 weakly error correctable.*

See Section 9 for a proof of Theorem 3.7.

3.4. Average-Case Hard Languages with Low Space. We also need the following folklore result, which can be proved by a direct diagonalization.

THEOREM 3.8. *Let $s\colon \mathbb{N} \to \mathbb{N}$ be a space-constructible function such that $s(n) \le 2^{o(n)}$ and $s(n) \ge n$ for every n. There is a universal constant c and a language $L \in \mathsf{SPACE}[s(n)^c]$ that $\mathsf{Avg}_{0.99}\text{-}\mathsf{SIZE}(L_n) > s(n)$ for all sufficiently large n.*

Proof. In the following we always assume that n is large enough. Let $c_1 \ge 1$ be a large enough constant and let $\ell = c_1 \log s(n)$. There are $2^{2^\ell} = 2^{s(n)^{c_1}}$ many 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 $2^{-\Omega(2^\ell)} \le 2^{-\Omega(s(n)^{c_1})} \le 2^{-s(n)^3}$, 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 for 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 a sufficiently large n. Then our algorithm computes f_0 on the first ℓ bits of the input, and ignores the rest of the input. (Note that here we use the fact that $\ell \le O(\log s(n)) \le o(n)$.) 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. ∎

3.5. MA ∩ coMA and NP ∩ coNP Algorithms. We first 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 3.9. *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 constant $c \ge 1$ and a deterministic algorithm $A(x, y, z)$ (which is called the predicate) such that:*

- *A takes three strings x, y, z such that $|x| = n$, $|y| = |z| = c \cdot T(n)$ 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 $\{0, 1, \perp\}$.*

[18] [59] does not explicitly state the TC^0 downward self-reducible property, but it is evident from their proof.

- *(Completeness) For every* $x \in \{0,1\}^*$*, there exists a* y *such that*

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

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

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

Moreover, let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *be such that* $\mathcal{C}_n$ *is a set of* $c \cdot T(n)$*-input circuits. We say that the randomness part of the predicate* L *is computable by* $\mathcal{C}$*, if there is an algorithm* B *such that for every* $n \in \mathbb{N}$*, given* $x \in \{0,1\}^n$ *and* $y \in \{0,1\}^{c\cdot T(n)}$*,* $B(x,y)$ *outputs a circuit* $C \in \mathcal{C}_n$ *in* $O(T(n))$ *time such that* $A(x,y,z) = C(z)$ *for every* $z \in \{0,1\}^{c\cdot T(n)}$*.*

REMARK 3.10. ($\mathsf{MA} \cap \mathsf{coMA}$) *languages with advice are defined similarly, with* A *being an algorithm with the corresponding advice.*

DEFINITION 3.11. *Let* $T\colon \mathbb{N} \to \mathbb{N}$ *be a time-constructible function. A language* L *is in* $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$*, if there is an algorithm* $A(x,y)$ *(which is called the predicate) such that:*

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

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

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

$$A(x,y) \ne 1 - L(x).$$

REMARK 3.12. $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$ *languages with advice are defined similarly, with* A *being an algorithm with the corresponding advice.*

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

3.6. Witness Circuits. Here we provide formal definition regarding witness circuits. Our definition below is adapted from [46].

DEFINITION 3.13. *Let* $T\colon \mathbb{N} \to \mathbb{N}$ *be time-constructible, and let* $L \in \mathsf{NTIME}[T(n)]$*. We say an algorithm* V *is a verifier for* L*, if for some* $\ell\colon \mathbb{N} \to \mathbb{N}$ *such that* $\ell(n) \le \log T(n) + O(1)$*,* V *takes two inputs* $x \in \{0,1\}^n$ *and* $y \in \{0,1\}^{2^{\ell(n)}}$ *and satisfies the condition that* $x \in L$ *if and only if there is* $y \in \{0,1\}^{2^{\ell(|x|)}}$ *such that* $V(x,y) = 1$*.*[19]

We say that V *has witness circuits of size* $w(n)$*, if for every large enough* $n \in \mathbb{N}$ *and every* $x \in L_n$*, there is a* $w(n)$*-size* $\ell(n)$*-input circuit* C_x *such that* $V(x, \mathsf{tt}(C_x)) = 1$*. And we say that* L *has witness circuits of size* $w(n)$*, if every verifier* V *for* L *has witness circuits of size* $w(n)$*.*

[19]Note that here we assume the witness length of V to be a power of 2 for simplicity. This assumption is without loss of generality since a verifier can always ignore part of the witness.

3.7. Hardness Amplification. We will also need some results in hardness amplification.

For $n \in \mathbb{N}$, $f \in \mathcal{F}_{n,1}$, and $k \in \mathbb{N}$, we use $f^{\oplus k}$ to be denote the (kn)-input function $f^{\oplus k}(x_1, \dots, x_k) \coloneqq \oplus_{i \in [k]} f(x_i)$, where $x_i \in \{0,1\}^n$ for every $i \in [k]$.

The following Lemma follows from a careful analysis of Levin's proof of Yao's XOR Lemma [41, 28]. We provide a proof in Appendix B for completeness.

LEMMA 3.14. *Let $\mathscr{C}$ be a typical 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.*

4. Random self-reduction for NC^1. In this section, we prove that NC^1 collapses to $\mathsf{AC}^0 \circ \mathscr{C}$ if uniform-NC^1 can be approximated by $\mathscr{C}$ circuits (Theorem 4.3, we also call it a collapse theorem for NC^1). In Subsection 4.1 we introduce the NC^1-complete language by Barrington, together with its random self-reduction. Next, in Subsection 4.2 we define a special encoding of the inputs to that language. The purpose here is to make sure the random self-reduction can be implemented by a *projection*.[20] Finally, in Subsection 4.3, we prove Theorem 4.3.

4.1. A Random Self-reducible NC^1-Complete Problem. We first define the following problem, iterated group product over S_5 (the group of all permutations on [5], we use id to denote the identity permutation), denoted as W_{S_5}, as follows:

> **Iterated group product over S_5 (W_{S_5})**
>
> Given n permutations $m_1, m_2, \dots, m_n \in S_5$, compute $\prod_{i=1}^n m_i$.

From the classical theorem of Barrington [12], W_{S_5} is NC^1-complete under projections. Formally, we have:

LEMMA 4.1 ([12]). *For every depth-d n-input circuit C, there is a projection $P\colon \{0,1\}^n \to \{0,1\}^{2^{O(d)}}$ such that $C(x) = 1$ if and only if $\mathsf{W}_{S_5}(P(x)) = \mathsf{id}$, for all $x \in \{0,1\}^n$.*

The above problem is random self reducible [10, 39], which is crucial for the proof of our collapse theorem. Here we recall its random self-reduction:

> **The random self-reduction of W_{S_5}**
>
> Given an input $\vec{m} = (m_1, \dots, m_n) \in (S_5)^n$ to W_{S_5} and $\vec{u} = (u_1, \dots, u_{n+1}) \in S_5^{n+1}$, we define the following input to W_{S_5}:
>
> $$\mathsf{Rand}(\vec{m}, \vec{u}) \coloneqq (u_1 m_1 u_2^{-1}, u_2 m_2 u_3^{-1}, \dots, u_n m_n u_{n+1}^{-1}).$$
>
> For every $\vec{m} \in (S_5)^n$, if we draw $\vec{u} \in_{\mathsf{R}} S_5^{n+1}$, then $\mathsf{Rand}(\vec{m}, \vec{u})$ is distributed as a uniform random input to W_{S_5}. Moreover, for every $\vec{u} \in S_5^{n+1}$, we have
>
> $$\mathsf{W}_{S_5}(\vec{m}) = u_1^{-1} \cdot \mathsf{W}_{S_5}(\mathsf{Rand}(\vec{m}, \vec{u})) \cdot u_{n+1}.$$

[20] We remark that projections are required only for proving average-case lower bounds against $\mathsf{ACC}^0 \circ \mathsf{THR}$. See Subsection 4.2 for more details.

4.2. A Special Encoding. It may seem that Lemma 4.1 and the random self-reduction of W_{S_5} are already sufficient for proving our collapse theorem for NC^1. But there are still some remaining technical problems.[21]

1. First, we have to encode W_{S_5} as a *Boolean function*. A naive way would be to construct a bijection between [120] and S_5, and then divide the input into blocks of 7 bits, each representing one element in S_5. The problem is that most of the Boolean inputs would be invalid in this encoding; therefore, this would make it a *promise problem* only defined on a negligible fraction of the inputs, which is not suited for our purpose.
2. Second, a straightforward implementation of the random self-reduction requires NC^0 circuits, as one needs to implement multiplication of two elements in S_5. This would collapse NC^1 to $\mathsf{ACC}^0 \circ \mathsf{THR} \circ \mathsf{NC}^0$ rather than $\mathsf{ACC}^0 \circ \mathsf{THR}$, and we currently do not know any non-trivial circuit-analysis algorithms for $\mathsf{ACC}^0 \circ \mathsf{THR} \circ \mathsf{NC}^0$.[22]

A special encoding for the second issue. We first deal with the second issue via a special encoding of the group elements. Note that $|S_5| = 120$. For each $i \in [120]$, let $e_i \in \{0,1\}^{120}$ be the vector with i-th bit being 1 while others are all 0. We identify S_5 with [120] (*i.e.*, we fix a bijection between S_5 and [120]), and use e_a to represent the element $a \in S_5$. Now the problem is formally defined as follows:

Iterated group product over S_5 with Boolean inputs (BW_{S_5})
Given n vectors $e_{a_1}, \dots, e_{a_n} \in \{0,1\}^{120}$, compute $a = \prod_{i=1}^{n} a_i$ and output e_a.

The advantage of this special encoding is that for all $p, q \in S_5$, there is a projection $P_{p,q} \colon \{0,1\}^{120} \to \{0,1\}^{120}$ (in fact, a permutation), such that for all $a \in S_5$, $P_{p,q}(e_a) = e_{p \cdot a \cdot q}$. This is crucial to make sure the random self-reduction can be implemented by a *projection* (so we can collapse NC^1 to $\mathsf{ACC}^0 \circ \mathsf{THR}$ instead of $\mathsf{ACC}^0 \circ \mathsf{THR} \circ \mathsf{NC}^0$).

Note that for $a \in S_5$, $(e_a)_{\mathsf{id}} = 1$ if and only if $a = \mathsf{id}$. We also have the following simple corollary of Lemma 4.1.

COROLLARY 4.2. *For every depth-d n-input circuit C, there is a projection $P \colon \{0,1\}^n \to \{0,1\}^{2^{O(d)}}$ such that $C(x) = \mathsf{BW}_{S_5}(P(x))_{\mathsf{id}}$ for every $x \in \{0,1\}^n$.*

Slightly abusing notation, we sometimes use $p \cdot m \cdot q$ to denote $P_{p,q}(m)$ for $p, q \in S_5$ and $m \in \{0,1\}^{120}$.

A redundant encoding for the first issue. The first issue still remains: BW_{S_5} is a promise problem as well, since we require all vectors to be one of the e_a's. We will use a redundant encoding to make this problem defined on all possible inputs.

Let $\mathcal{S}_{\mathsf{good}}$ be the set of all the e_a's for $a \in S_5$ (*i.e.*, all vectors in $\{0,1\}^{120}$ with hamming weight 1), and $\mathcal{S}_{\mathsf{bad}}$ be all other vectors in $\{0,1\}^{120}$.

We define the following problem $\mathsf{Redundant}\text{-}\mathsf{W}_{S_5}$:

Iterated group product over S_5 with a redundant encoding

[21] We remark that similar issues arise in [29] as well.

[22] This is not an issue if we only wish to prove average-case lower bounds against ACC^0, since $\mathsf{ACC}^0 \circ \mathsf{NC}^0$ is contained in ACC^0.

(Redundant-W_{S_5})

We are given n^2 vectors $\{m_{i,j}\}_{(i,j)\in[n]\times[n]}$ from $\{0,1\}^{120}$.
For each $i \in [n]$, let j_i be the first integer such that $m_{i,j_i} \in \mathcal{S}_{\mathsf{good}}$.

- We call the input a bad input, if there is no such j_i for some i, and we just output the all-zero vector of length 120 in this case.
- Otherwise, we call the input a good input. For every $i \in [n]$, let $a_i \in S_5$ be such that $m_{i,j_i} = e_{a_i}$. Our goal is to compute $a = \prod_{i=1}^{n} a_i$ and output e_a.

The definition of Redundant-W_{S_5} above ensures that only a negligible fraction of the inputs are bad, and resolves our first issue.

We note that Redundant-W_{S_5} is in uniform NC.[23] For each $i \in [120]$, we use Redundant-$\mathsf{W}_{S_5}^{(i)}$ to denote the Boolean language corresponding to the i-th output bit of Redundant-W_{S_5}. Formally, given an input $z \in \{0,1\}^*$, Redundant-$\mathsf{W}_{S_5}^{(i)}(z)$ outputs the i-th bit of Redundant-$\mathsf{W}_{S_5}(z)$ if $|z| = 120n^2$ for some $n \in \mathbb{N}$, and outputs 0 otherwise. Clearly, for every $i \in [120]$, Redundant-$\mathsf{W}_{S_5}^{(i)}$ is in also uniform NC.

4.3. NC^1 Collapses to $\mathsf{AC}^0 \circ \mathscr{C}$ if Uniform NC^1 can be Approximated by $\mathscr{C}$. Now we are ready to show that for a general circuit class $\mathscr{C}$, NC^1 collapses to $\mathsf{AC}^0 \circ \mathscr{C}$, if uniform NC^1 can be approximated by $\mathscr{C}$.

THEOREM 4.3. *Let $\mathscr{C}$ be a typical circuit class, and let $S\colon \mathbb{N} \to \mathbb{N}$ be a size parameter. There is a universal constant $\delta \in (0,1)$ such that, if for every $i \in [120]$ Redundant-$\mathsf{W}_{S_5}^{(i)}$ can be $(1-\delta)$-approximated by S-size $\mathscr{C}$ circuit families, then every depth-d n-input circuit D inputs has an equivalent* $\mathrm{poly}(S(2^{O(d)}), n)$*-size $\mathsf{AC}_3 \circ \mathscr{C}$ circuit.*

Proof. Let $\delta = 1/480$, and D be a depth-d circuit on n input. By Corollary 4.2, there is a projection $P\colon \{0,1\}^n \to \{0,1\}^\ell$ where $\ell \le 2^{O(d)}$, such that $D(x) = \mathsf{BW}_{S_5}(P(x))_{\mathsf{id}}$ for every $x \in \{0,1\}^n$. Without loss of generality, we can assume that n is sufficiently large and $d \ge \log n$.

Construction of the circuit C approximating Redundant-W_{S_5}. Now, let $t = \ell/120$ (*i.e.*, BW_{S_5} on ℓ bits computes the iterated group product of t permutations from S_5). Now we consider the Redundant-W_{S_5} problem on t^2 vectors.

From the assumption, there are 120 $\mathscr{C}$ circuits $\{C_i\}_{i\in[120]}$ such that C_i $(1-\delta)$-approximates the i-th output bit of Redundant-W_{S_5}. We also use $C(x) \in \{0,1\}^{120}$ to denote the vector $(C_1(x), C_2(x), \ldots, C_{120}(x))$.

By a simple union bound, we have

$$\Pr_{z \in_{\mathsf{R}} \{0,1\}^{120t^2}} [\text{Redundant-}\mathsf{W}_{S_5}(z) = C(z)] \ge 1 - \delta \cdot 120 \ge 0.75. \tag{4.1}$$

On the other hand, note that a random input to Redundant-W_{S_5} is a good input with probability at least

$$1 - t \cdot \left(\frac{|\mathcal{S}_{\mathsf{bad}}|}{2^{120}}\right)^t \ge 0.99, \tag{4.2}$$

[23] We can first compute all the j_i for $i \in [n]$ in uniform NC^1. If any of the j_i does not exist, we output the all-zero vector with length 120. Otherwise, we compute BW_{S_5} with inputs being all the m_{i,j_i} for $i \in [n]$, which can be done in uniform NC^1 as well.

when n (and therefore t) is sufficiently large.

Let $\mathcal{RW}_{\mathsf{good}}$ be the set of all good inputs to $\mathsf{Redundant\text{-}W}_{S_5}$. Combining (4.1) and (4.2) and applying another union bound, it follows that

$$\Pr_{z \in_{\mathsf{R}} \mathcal{RW}_{\mathsf{good}}} [\mathsf{Redundant\text{-}W}_{S_5}(z) = C(z)] \geq 0.7. \tag{4.3}$$

Implementation of the random self-reduction. Now we define the function $\mathsf{First}\colon \{0,1\}^{120t} \to \mathcal{S}_{\mathsf{good}} \cup \{\bot\}$. Given an input $\vec{m} = (m_1, m_2, \ldots, m_t) \in (\{0,1\}^{120})^t$, letting j be the first integer that $m_j \in \mathcal{S}_{\mathsf{good}}$, we define $\mathsf{First}(\vec{m}) = m_j$. If there is no such j, we define $\mathsf{First}(\vec{m}) = \bot$.

For each $m \in \mathcal{S}_{\mathsf{good}}$, we define $\mathcal{M}_m$ be the uniform distribution over the set $\{z \in \{0,1\}^{120t} : \mathsf{First}(z) = m\}$. Note that a sample from $\mathcal{M}_m$ can be generated as follows:

- For $j \in [t]$, let p_j be the probability that a random sample $\vec{w} = (w_1, \ldots, w_t)$ from $\mathcal{M}_m$ satisfies that j is the first integer that $w_j \in \mathcal{S}_{\mathsf{good}}$ (note that we must have $w_j = m$).
- We first draw $j \in [t]$ according to the probabilities p_j's. Then a sample $\vec{w} = (w_1, w_2, \ldots, w_t)$ from $\mathcal{M}_m$ can be generated as follows: for $k \in [j-1]$, we set w_k to be a uniform sample from $\mathcal{S}_{\mathsf{bad}}$; we set $w_j = m$; for $k \in \{j+1, j+2, \ldots, t\}$, we set w_k to be a uniform sample from $\{0,1\}^{120}$.

Note that when the randomness in the above process is fixed (*i.e.*, j is fixed, together with w_k for $k \in [t] \setminus j$), then a sample generated as above is a projection of m. (Indeed, only the j-th part of the sample is now set to m, and other parts are completely fixed by the randomness.)

Next, given a valid input $\vec{m} = (m_1, m_2, \ldots, m_t)$ to BW_{S_5} (*i.e.*, $\vec{m} \in \mathcal{S}_{\mathsf{good}}^t$), we define an input distribution to $\mathsf{Redundant\text{-}W}_{S_5}$, denoted by $\mathcal{N}_{\vec{m}}$, generated as follows:

1. We draw $\vec{u} = (u_1, u_2, \ldots, u_t, u_{t+1}) \in_{\mathsf{R}} S_5^{t+1}$, and set
$$\vec{v} = \mathsf{Rand}(\vec{m}, \vec{u}) = (u_1 m_1 u_2^{-1}, u_2 m_2 u_3^{-1}, \ldots, u_t m_t u_{t+1}^{-1}).$$
2. For each $i \in [t]$, we draw w_i from $\mathcal{M}_{v_i}$ independently. Then we output $w_1, w_2, \ldots, w_t$.

We claim that for every $\vec{m} \in \mathcal{S}_{\mathsf{good}}^t$, $\mathcal{N}_{\vec{m}}$ is distributed identically to a random good input to $\mathsf{Redundant\text{-}W}_{S_5}$.

To see this, note that for every $\vec{m} \in \mathcal{S}_{\mathsf{good}}^t$, $\vec{v} = \mathsf{Rand}(\vec{m}, \vec{u})$ is distributed uniformly random on the set $\mathcal{S}_{\mathsf{good}}^t$. Therefore, the distribution of $\mathcal{N}_{\vec{m}}$ is identical to the following distribution: one first draws $\vec{v} \in_{\mathsf{R}} \mathcal{S}_{\mathsf{good}}^t$, and then draws w_i from $\mathcal{M}_{v_i}$ independently for every $i \in [t]$. By the definition of good inputs to $\mathsf{Redundant\text{-}W}_{S_5}$, the later distribution is identical to the uniform distribution over good inputs to $\mathsf{Redundant\text{-}W}_{S_5}$.

Moreover, for every $\vec{u} \in S_5^{t+1}$, it holds that

$$\mathsf{BW}_{S_5}(\vec{m}) = u_1^{-1} \cdot \mathsf{BW}_{S_5}(\mathsf{Rand}(\vec{m}, \vec{u})) \cdot u_{t+1}. \tag{4.4}$$

Note that a sample of $\mathcal{N}_{\vec{m}}$ is generated from both the randomness over $\vec{u} \in S_5^{t+1}$, and the randomness used in generating all the w_i from $\mathcal{M}_{v_i}$. Formally, there is a set $\mathcal{R}$ and a function $\mathsf{Gen}(\vec{m}, \vec{u}, r)$ (here we use r to denote the randomness used to generate all the w_i), such that $\mathsf{Gen}(\vec{m}, \vec{u}, r)$ is distributed identically to $\mathcal{N}_{\vec{m}}$ when r is drawn from $\mathcal{R}$ and $\vec{u}$ is drawn from S_5^{t+1}.

Finally, applying (4.3) and (4.4), for any $\vec{m} \in \mathcal{S}_{\mathsf{good}}^t$, we have

$$\Pr_{\vec{u} \in_{\mathsf{R}} \mathcal{S}_{\mathsf{good}}^{t+1}} \Pr_{r \in_{\mathsf{R}} \mathcal{R}} \left[\mathsf{W}_{S_5}(\vec{m}) = u_1^{-1} \cdot C(\mathsf{Gen}(\vec{m}, \vec{u}, r)) \cdot u_{t+1}\right] \geq 0.7.$$

Construction of the final circuit E. Now, one can see that when $\vec{u}$ is *fixed*, $\mathsf{Rand}(\vec{m},\vec{u})$ is a projection of $\vec{m}$ (since $u_i m_i u_{i+1}^{-1} = P_{u_i,u_{i+1}^{-1}}(m_i)$ is a projection of m_i). And when r is fixed, $\mathsf{Gen}(\vec{m},\vec{u},r)$ is also a projection of $\mathsf{Rand}(\vec{m},\vec{u})$. Therefore, when both $\vec{u}$ and r are fixed, $\mathsf{Gen}(\vec{m},\vec{u},r)$ is a projection of $\vec{m}$.

Next, we pick $T = 100n$ i.i.d. samples $\vec{u}^1, \vec{u}^2, \dots, \vec{u}^T$ from $\mathcal{S}_{\mathsf{good}}^{t+1}$, and $r^1, r^2, \dots, r^T$ from $\mathcal{R}$. For each $j \in [T]$, we define the circuit

$$E_j(x) \coloneqq \left((u_1^j)^{-1} \cdot C(\mathsf{Gen}(P(x), \vec{u}^j, r^j)) \cdot u_{t+1}^j\right)_{\mathsf{id}}.$$

Note that E_j can be computed by a $\mathscr{C}$ circuit of size $S_1 = \mathrm{poly}(S(2^{O(d)}), n)$. Moreover, for each $x \in \{0,1\}^n$, over the randomness of $\vec{u}^j$ and r^j, we have

$$\Pr[E_j(x) = D(x)] \geq 0.7.$$

Therefore, we set our final circuit to be an approximate-majority of these T circuits $E_1, E_2, \dots, E_T$. By a simple Chernoff bound, there is a fixed choice of all the $\vec{u}^j$'s and r^j's, such that the resulting circuit E computes D exactly. By Lemma 3.1, E is an $\mathsf{AC}_3 \circ \mathscr{C}$ circuit of size $T \cdot S_1 + \mathrm{poly}(T) \leq \mathrm{poly}(S(2^{O(d)}), n)$, which completes the proof. ∎

The following corollary follows immediately from Theorem 4.3.

Corollary 4.4. *Let $\mathscr{C}$ be a typical circuit class, and let $S\colon \mathbb{N} \to \mathbb{N}$ be a size parameter. There is a universal constant $\delta \in (0,1)$ such that, if all languages in uniform NC^1 can be $(1-\delta)$-approximated by S-size $\mathscr{C}$ circuit families, then any depth-d n-input circuit D has an equivalent $\mathrm{poly}(S(2^{O(d)}), n)$-size $\mathsf{AC}_3 \circ \mathscr{C}$ circuit.*

5. Construction of i.o. NPRG for Low-Depth Circuits.

In this section we construct the required i.o. NPRG for low-depth circuits, under the assumption that for some typical circuit class $\mathscr{C}$, (1) uniform NC^1 can be approximated by $\mathscr{C}$ circuits and (2) $\mathsf{Gap\text{-}UNSAT}$ for $\mathsf{AC}_0 \circ \mathscr{C}$ has a non-trivial algorithm. (See Theorem 5.3 for details.) We also a non-trivial algorithm for $\mathsf{Gap\text{-}UNSAT}$ for $\mathsf{Circuit}$ implies an i.o. NPRG for general circuits.

In Subsection 5.1 we show that for every typical circuit class $\mathscr{C}$, witness lower bounds against $\mathscr{C}$ circuits follows from a non-trivial $\mathsf{Gap\text{-}UNSAT}$ algorithm for $\mathsf{AC}_2 \circ \mathscr{C}$ circuits. Then in Subsection 5.2, we construct our conditional i.o. NPRGs.

5.1. Witness-Size Lower Bound for NE.

The following lemma is proved by combining ideas from [67] with the new PCP construction of [13].

Lemma 5.1. *Let $\mathscr{C}$ be a typical circuit class. Suppose there is an $\varepsilon \in (0,1)$ such that the $\mathsf{Gap\text{-}UNSAT}$ problem for $2^{n^{\varepsilon}}$-size n-input $\mathsf{AC}_2 \circ \mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ non-deterministic time. Then there is a polynomial-time verifier $V(x,y)$ with $|x| = n$ and $|y| = 2^n$, such that for infinitely many n, $V(1^n, \cdot)$ is satisfiable, and $V(1^n, y) = 1$ implies that $\mathsf{func}(y)$ cannot be computed by $2^{n^{\varepsilon/2}}$-size $\mathscr{C}$ circuits.*

To prove Lemma 5.1, we need the following PCP construction from [13].

Lemma 5.2 ([13]). *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. *If there is a* $y \in \{0,1\}^{T(n)}$ *such that* $M(x,y)$ *accepts then there is a map* $\pi\colon \{0,1\}^r \to \{0,1\}$ *such that for all* $z \in \{0,1\}^r$, $R(\pi(q_1),\dots,\pi(q_t)) = 1$ *where* $(q_1,\dots,q_t) = Q(z)$.

Soundness. *If no* $y \in \{0,1\}^{T(n)}$ *causes* $M(x,y)$ *to accept, then for every map* $\pi\colon \{0,1\}^r \to \{0,1\}$, *at most* $\frac{2^r}{n^{10}}$ *many* $z \in \{0,1\}^r$ *have* $R(\pi(q_1),\dots,\pi(q_t)) = 1$ *where* $(q_1,\dots,q_t) = Q(z)$.

Complexity. *Q is a projection, i.e., each output bit of Q is a bit of input, the negation of a bit, or a constant. R is a 3-CNF.*

Now we are ready to prove Lemma 5.1.

Proof of Lemma 5.1. Let L be a unary language such that $L \in \mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n]$, whose existence is guaranteed by the non-deterministic time hierarchy theorem [69].

Given an input 1^n to L, we apply Lemma 5.2 to L to obtain a $\mathrm{poly}(n)$-size AC_2 oracle circuit VPCP_n that takes $\ell(n) = n + O(\log n)$ random bits as input, and queries an oracle $\mathcal{O}\colon \{0,1\}^\ell \to \{0,1\}$. From the complexity part of Lemma 5.2, for a $\mathscr{C}$ circuit C of size S, VPCP_n^C is an $\mathsf{AC}_2 \circ \mathscr{C}$ circuit with size at most $S \cdot \mathrm{poly}(n)$. Moreover, from the completeness and soundness part of Lemma 5.2, we have:

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

$$\Pr_{r \in_{\mathsf{R}} \{0,1\}^\ell}[\mathsf{VPCP}_n^C(r) = 1] = 1.$$

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

$$\Pr_{r \in_{\mathsf{R}} \{0,1\}^\ell}[\mathsf{VPCP}_n^C(r) = 1] \le 1/n^{10}.$$

Now we consider the following non-deterministic algorithm A_{PCP} attempting to solve L: Given an input 1^n, A_{PCP} guesses a $2^{\ell^{\varepsilon/2}}$-size ℓ-input $\mathscr{C}$ circuit C, and runs the assumed non-deterministic algorithm for $\mathsf{Gap\text{-}UNSAT}$ on $\neg\mathsf{VPCP}_n^C$. It accepts if $\neg\mathsf{VPCP}_n$ is a yes instance of $\mathsf{Gap\text{-}UNSAT}$, and rejects if it is a no instance.[24]

By previous discussions, VPCP_n^C is an $\mathsf{AC}_2 \circ \mathscr{C}$ circuit of size at most 2^{ℓ^ε}, and therefore A_{PCP} runs in at most $2^\ell/\ell^{\omega(1)} \le 2^n/n$ non-deterministic time.

Since $L \notin \mathsf{NTIME}[2^n/n]$, it follows that A_{PCP} does not compute L. From the soundness property of VPCP, $1^n \notin L$ implies that $\neg\mathsf{VPCP}_n^C$ is a no instance of $\mathsf{Gap\text{-}UNSAT}$ for every C, and A_{PCP} rejects 1^n. Hence, for infinitely many n, we have $1^n \in L$ and yet A_{PCP} rejects 1^n. We call these n good.

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

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

Finally we argue that for every good n, $V(1^{\ell(n)}, \cdot)$ satisfies our requirements. First, since $1^n \in L$, from the completeness of VPCP, it follows that there is $y \in \{0,1\}^{2^\ell}$ such that $V(1^\ell, y)$ accepts. Second, since A_{PCP} rejects 1^n, it means for every $2^{\ell^{\varepsilon/2}}$-size $\mathscr{C}$ circuit $C\colon \{0,1\}^\ell \to \{0,1\}$, we must have $V(1^\ell, \mathsf{tt}(C)) = 0$. Meaning that for every y such that $V(1^\ell, y)$ accepts, $\mathsf{func}(y)$ cannot be computed by $2^{\ell^{\varepsilon/2}}$-size $\mathscr{C}$ circuits. ∎

[24] A_{PCP} may either accept or reject when $\neg\mathsf{VPCP}_n$ is neither a yes instance nor a no instance. We will see this does not affect our proof.

5.2. The NPRG Construction. Now we are ready to give the construction of our conditional NPRGs.

THEOREM 5.3. *(Conditional i.o. NPRG for low-depth circuits) Let $\mathscr{C}$ be a typical circuit class. There is a universal constant $\delta \in (0,1)$ such that, suppose the following hold*

1. *there is an $\varepsilon \in (0,1)$ such that the $\mathsf{Gap\text{-}UNSAT}$ problem for 2^{n^ε}-size n-input $\mathsf{AC}_5 \circ \mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ non-deterministic time, and*
2. *uniform NC^1 can be $(1-\delta)$-approximated by $2^{\log^c n}$-size $\mathscr{C}$ circuit families for some $c \in \mathbb{N}$.*

Then for every $a \in \mathbb{N}$, there is $b \in \mathbb{N}$ and an NPRG family $G = \{G_n\}$ such that

1. *For infinitely many n, for $S = 2^{\log^a n}$, G_n is an NPRG for S-size $\log S$-depth circuits with S-bit inputs, with error $1/S$.*
2. *G is computable in $2^{\log^b n}$ time and has seed length $\log^b n$.*

In other words, let $\mathcal{H} = \{\mathcal{H}_n\}$ be such that $\mathcal{H}_n$ is the set of S-size $\log S$-depth circuits with S-bit inputs. G is an i.o. NPRG for $\mathcal{H}$ with error $1/S$.

Proof. Let δ be the universal constant in Corollary 4.4. Without of loss generality, we assume that n is a sufficiently large integer. Recall that an NPRG G_n is a pair of functions $(G_{\mathsf{P}})_n$ and $(G_{\mathsf{W}})_n$. We will write the pair as $G_{\mathsf{P}}^{(n)}$ and $G_{\mathsf{W}}^{(n)}$ for notational convenience.

Construction of the "hardness certifier" V' for low-depth circuits. We first combine Corollary 4.4 with the witness-size lower bound from Lemma 5.1 to construct a hardness certifier V'.

Let $d = \log^k n$ for a constant k to be specified later. By Corollary 4.4 and our second assumption, we know that a depth-d n-input circuit has an equivalent $2^{c_e \cdot d^c}$-size $\mathsf{AC}_3 \circ \mathscr{C}$ circuit for a universal constant c_e.

Let $a_1 \in \mathbb{N}$ to be specified later. Applying Lemma 5.1 for the circuit class $\mathsf{AC}_3 \circ \mathscr{C}$, there is a large enough constant $b_1 = b_1(a_1)$ and a polynomial-time algorithm $V'(x,y)$ with $|x| = \log^{b_1} n$, $|y| = 2^{\log^{b_1} n}$, such that for infinitely many n's, we have that $V'(1^{\log^{b_1} n}, \cdot)$ is satisfiable, and $V'(1^{\log^{b_1} n}, y) = 1$ implies that $\mathsf{func}(y)$ cannot be computed by a $2^{\log^{a_1} n}$-size $\mathsf{AC}_3 \circ \mathscr{C}$ circuit. We will call these n's good.

Now, we set $a_1 = ck + 1$ (hence $\log^{a_1} n > c_e \log^{ck} n = c_e d^c$) so that for a good n and a string y of length $2^{\log^{b_1} n}$ such that $V'(1^{\log^{b_1} n}, y) = 1$, we know that $\mathsf{func}(y)$ cannot be computed by depth-d circuits.

Construction of the NPRG. Now we can plug this y into a standard construction of a PRG. Let c_2, g and G be the constants and the algorithm in Theorem 3.3. We also set $\ell = \log^{b_1} n$, $w = 2^\ell$, and $m = 2^{\log^a n}$. Now we are ready to define $G_{\mathsf{P}}^{(n)}$ and $G_{\mathsf{W}}^{(n)}$ as follows:

- $G_{\mathsf{W}}^{(n)}$ takes a w-bit string y as input, and outputs $V'(1^\ell, y)$.
- $G_{\mathsf{P}}^{(n)}$ takes a w-bit string y and an ℓ^g-bit string z as input, and outputs $G_{\ell,m}(y,z)$.

Now we set $k = a/c_2$ and verify that $G_n = (G_{\mathsf{P}}^{(n)}, G_{\mathsf{W}}^{(n)})$ is an NPRG for $\mathcal{H}_n$ with error $1/m$ when n is good.

Since n is good, we know that there exists $y \in \{0,1\}^w$ such that $G_{\mathsf{W}}^{(n)}(y) = 1$, and for such y, by previous discussions, $\mathsf{func}(y)$ cannot be computed by $\log^k n$-depth circuits. By Theorem 3.3 and the fact that $\log^{c_2 k} n \ge \log^a n$, $G_{\ell,m}(y,\cdot)\colon \{0,1\}^{\ell^g} \to \{0,1\}^m$ is a PRG for $\mathcal{H}_n$ with error $1/m$, and is computable in $\mathrm{poly}(|y|) \le 2^{O(\ell)}$ time. Finally we set $b = b_1 \cdot g$, and this completes the proof. ∎

The following theorem is a direct corollary of Theorem 3.2 and Lemma 5.1.

THEOREM 5.4. *Suppose there is an* $\varepsilon \in (0,1)$ *such that the* Gap-UNSAT *problem for* 2^{n^ε}*-size* n*-input circuits can be solved in* $2^n/n^{\omega(1)}$ *non-deterministic time. Then there is an NPRG family* $G = \{G_n\}$ *such that*

1. *For infinitely many* n*, for* $S = 2^{\log^a n}$*,* G_n *is an NPRG for* S*-size* S*-input circuits with error* $1/S$*.*
2. G *is computable in* $2^{\log^b n}$ *time and has seed length* $\log^b n$*.*

6. A Simpler Proof for the New Easy Witness Lemma for NP and NQP of [46]. In this section, we present our simpler proof of the easy-witness lemma for NP from [46] (it is straightforward to adapt that for NQP). This also serves as a warm-up for our a.a.e. average-case MA lower bound in **??**, which is the technical centerpiece of this paper.

As already discussed in Section 2, the technical centerpiece of the new easy witness lemma of [46] is an a.a.e. MA circuit lower bound. In Subsection 6.1, we first give a simpler proof of that MA lower bound. Then in Subsection 6.2, we sketch how to prove the easy-witness lemma for NP based on that (this is basically an adaption of the proof of [46, Lemma 4.1]).

We also remark that our proof in fact follows a case-analysis that is similar to the fixed polynomial-size circuit lower bounds for $\mathsf{MA}_{/1}$ in [51], while relying on additional nice properties (paddability and downwards self-reducibility) of the PSPACE-complete language L^{PSPACE} from Theorem 3.7.

6.1. A.a.e. Fixed-polynomial Lower Bounds for $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$**.** Now we are ready to prove the a.a.e. fixed-polynomials lower bounds for $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$.

LEMMA 6.1. *For all constants* k*, there is* $c \in \mathbb{N}$ *and a language* $L \in (\mathsf{MA} \cap \mathsf{coMA})_{/1}$*, such that for all sufficiently large* $\tau \in \mathbb{N}$ *and* $n = 2^\tau$*, either*

- $\mathsf{SIZE}(L_n) > n^k$ *or*
- $\mathsf{SIZE}(L_m) > m^k$ *for some* $m \in (n^c, 2\cdot n^c) \cap \mathbb{N}$*.*

Our relaxation of the a.a.e. condition. The statement of Lemma 6.1 also illustrates our relaxation of the a.a.e. condition that is crucial in the average-case setting. In [46], the lower bound shows that for almost all n's and $m = n^c$, either $\mathsf{SIZE}(L_n) > n^k$ or $\mathsf{SIZE}(L_m) > m^k$. This lower bound of [46] only holds for an $\mathsf{MA}_{/O(\log n)}$ language. Here we relax the a.a.e. condition by only requiring the lower bound to hold for almost all n that is a power of 2 and some $m \in (n^c, 2\cdot n^c)$. This relaxation enables us to prove a lower bound for an $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$ language. In Subsection 6.2, we show how the simplification above still suffices for the proof of the easy witness lemma for NP.

Proof of Lemma 6.1. Let L^{PSPACE} be the language specified by Theorem 3.7. By Theorem 3.8, there is $c_1 \in \mathbb{N}$ and a language $L^{\mathsf{diag}} \in \mathsf{SPACE}(n^{c_1})$ such that $\mathsf{SIZE}(L^{\mathsf{diag}}_n) \geq n^k$ for all sufficiently large n. Since L^{PSPACE} is PSPACE-complete and paddable, there is $c_2 \in \mathbb{N}$ such that L^{diag}_n can be reduced to L^{PSPACE} on input length n^{c_2} in $O(n^{c_2})$ time. We set $c = c_2$.

The algorithm. Let $\tau \in \mathbb{N}$ be sufficiently large. We also let b be a large enough constant to be specified later (we will make sure $b \gg k$). Given an input x of length $n = 2^\tau$ and for $m = n^c$, we first provide an informal description of the $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$ algorithm A_L that computes the language L. There are two cases:

1. When $\mathsf{SIZE}(L^{\mathsf{PSPACE}}_m) \leq n^b$. That is, when L^{PSPACE}_m is *easy*. In this case, on inputs of length n, we guess-and-verify a circuit for L^{PSPACE}_m of size 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 $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \le n^b$.[25] On inputs of length $m_1 = m + \ell$, we guess-and-verify a circuit for L_ℓ^{PSPACE}, and compute it (*i.e.*, compute L_ℓ^{PSPACE} on the first ℓ input bits and ignore the rest).[26]

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_1. A formal description of A_L is given in Algorithm 6.1, and an algorithm A_{adv} for setting the advice sequence of A_L is given in Algorithm 6.2.

To complete the description of our $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ algorithm, we claim that an α_n can only be set once in Algorithm 6.2. To see this, we first note that Line 5 only sets α_n such that n is a power of 2. And also, whenever one enters Line 8, we have that (1) $m = n^c$ is a power of 2 and (2) $1 \le \ell < m$ since $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) > n^b$ and $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}})$ is nondecreasing. Hence, at Line 8, $m + \ell$ is never a power of 2. The above discussions means that an α_n cannot be set by both Line 5 and Line 8. Further observing that an α_n cannot be set twice by Line 5 or Line 8 finishes the proof of our claim.

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

Given an input x with input length $n = |x|$;
Given an advice bit $\alpha = \alpha_n \in \{0,1\}$;
Let $m = n^c$;
Let $n_0 = n_0(n)$ be the largest integer such that $n_0^c \le n$;
Let $m_0 = n_0^c$;
Let $\ell = n - m_0$;
if $\alpha = 0$ **then**
    Output 0 and terminate
if n *is a power of* 2 **then**
    // We are in the case that $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \le n^b$.
    Compute $z \in \{0,1\}^m$ in $O(n^c)$ time such that $L_n^{\mathsf{diag}}(x) = L_m^{\mathsf{PSPACE}}(z)$;
    Guess an m-input circuit C of size at most n^b;
    Let M be the instance checker for L_m^{PSPACE};
    Flip an appropriate number of random coins, let them be r;
    Accept if $M^C(z, r) = 1$;
else
    // We are in the case that $\mathsf{SIZE}(L_{m_0}^{\mathsf{PSPACE}}) > n_0^b$ and ℓ is the largest integer such that $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \le n_0^b$.
    Let $z \in \{0,1\}^\ell$ be the first ℓ bits of x;
    Guess an ℓ-input circuit C of size at most n_0^b;
    Let M be the instance checker for L_ℓ^{PSPACE};
    Flip an appropriate number of random coins, let them be r;
    Accept if $M^C(z, r) = 1$;

[25] Here we have $\mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}}) > n^b$ by the choice of ℓ. Since L^{PSPACE} is downward self-reducible and b is a large enough constant, we have $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \ge n^{b/2}$. Therefore, L_ℓ^{PSPACE} is hard as well.

[26] We choose input length $m_1 = m + \ell$ instead of ℓ because we wish to show L is hard on an input length in $(n^c, 2 \cdot n^c) \cap \mathbb{N}$ and ℓ can be smaller than n^c.

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

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

Now it remains to show that (1) A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise (see Definition 3.9) and (2) A_L computes a hard language.

A_L **satisfies the $\mathsf{MA}\cap\mathsf{coMA}$ promise**. We first show A_L satisfies the $\mathsf{MA}\cap\mathsf{coMA}$ promise. The intuition is that A_L only tries to guess-and-verify a circuit for L^{PSPACE} when it exists, and the properties of the instance checker (see Definition 3.4) ensure that in this case A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise . There are three cases:

1. $\alpha_n = 0$. In this case, A_L computes the all zero function, and clearly satisfies the promise.
2. $\alpha_n = 1$ and n is a power of 2. In this case, from Algorithm 6.2, we know that $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$ 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 3.4). 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 know that $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n_0^b$. 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 $1/3$.

To summarize, we have the following claim.

CLAIM 1. *The algorithm 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 $n \in \mathbb{N}$, 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.*

A_L **computes a hard language**. Next we show that the algorithm 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. $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \leq n^b$. In this case, we have $\alpha_n = 1$ by Algorithm 6.2. By Item (2) of Claim 1, we have that L_n is the same function as L_n^{diag}, and therefore $\mathsf{SIZE}(L_n) > n^k$.
2. $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) > n^b$. Let ℓ be the largest integer such that $\mathsf{SIZE}(L_\ell^{\mathsf{PSPACE}}) \leq n^b$. By Remark 3.5, we have $0 < \ell < m$.

Note that $\mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}}) \leq (\ell+1)^d \cdot \mathsf{SIZE}(L_{\ell}^{\mathsf{PSPACE}})$ for a universal constant d, because L^{PSPACE} is downward self-reducible. Therefore,

$$\mathsf{SIZE}(L_{\ell}^{\mathsf{PSPACE}}) \geq \mathsf{SIZE}(L_{\ell+1}^{\mathsf{PSPACE}})/(\ell+1)^d \geq n^b/m^d \geq n^{b-c\cdot d}.$$

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 $\ell \in (0, m)$). Then by Item (3) of Claim 1, we have that L_{m_1} is the m_1-input function that computes $L_{\ell}^{\mathsf{PSPACE}}$ on the first ℓ bits and ignores the last m input bits. Hence, we have

$$\mathsf{SIZE}(L_{m_1}) = \mathsf{SIZE}(L_{\ell}^{\mathsf{PSPACE}}) \geq n^{b-c\cdot d}.$$

We set b such that $n^{b-c\dot{d}} \geq (2m)^k \geq m_1^k$ (we can set $b = cd + 3 \cdot ck$), which completes the proof. ∎

6.2. An Easy-Witness Lemma for NP. Now we sketch the proof for the easy-witness lemma for NP, which also illustrates why our relaxation of a.a.e. condition is still enough for the purpose of proving lower bounds.

First we need the following simple lemma.

LEMMA 6.2. *For a constant k, if $\mathsf{NP}_{/O(n)}$ is not in $\mathsf{SIZE}[O(n^k)]$, then NP is not in $\mathsf{SIZE}[n^k]$.*

Proof. We prove the contrapositive. Suppose NP is in $\mathsf{SIZE}(n^k)$ for an integer k. Let $L \in \mathsf{NP}_{/cn}$ for a constant c, and M and $\{\alpha_n\}_{n\in\mathbb{N}}$ be its corresponding non-deterministic Turing machine and advice sequence. Let $p(n)$ be a polynomial running time upper bound of M on inputs of length n.

Now we define a language L' such that a pair $(x, \alpha) \in L'$ if and only if $c|x| = |\alpha|$ and M accepts x with advice bits set to α in $p(|x|)$ steps. Clearly, $L' \in \mathsf{NP}$ from the definition, so it has an n^k-size circuit family. Fixing the advice bits to the actual α_n's in the circuit family, we have an $O(n^k)$-size circuit family for L as well. This completes the proof. ∎

Reminder of Lemma 1.6. *For all $k \geq 1$, there is a constant b such that if $\mathsf{NP} \subset \mathsf{SIZE}[n^k]$, then every $L \in \mathsf{NP}$ has witness circuits of size at most n^b.*

Proof Sketch. Fix $k \geq 1$, let $b = b(k)$ be a constant to be specified later. We prove the contrapositive of the lemma: if some $L \in \mathsf{NP}$ does not have witness circuits of size at most n^b, then $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$.

Now we assume that there is a language $L \in \mathsf{NP}$ that does not have n^b-size witness circuits. From definition 3.13, there is a constant $a \in \mathbb{N}$, $\ell(n) = \lceil a \log n \rceil$, and a polynomial-time verifier $V(x, y)$ for L ($x \in L \Leftrightarrow \exists y\ V(x, y) = 1$) with $|x| = n$ and $|y| = 2^{\ell(n)}$ such that for infinite many $n \in \mathbb{N}$, there is $x_n \in L_n$ satisfying that (1) $V(x_n, \cdot)$ is satisfiable and (2) $V(x_n, y) = 1$ implies that $\mathsf{func}(y)$ does not have n^b size circuits. We call these n good.

Let c_1 and G^{Umans} be the constant and the algorithm from Theorem 3.2. We construct an NPRG $G_n = (G_{\mathsf{P}}^{(n)}, G_{\mathsf{W}}^{(n)})$ as follows:

- Both $G_{\mathsf{P}}^{(n)}$ and $G_{\mathsf{W}}^{(n)}$ takes an input $x_n \in \{0,1\}^n$ as the advice.
- $G_{\mathsf{W}}^{(n)}$ takes a string $y \in \{0,1\}^{2^{\ell(n)}}$ as input, and outputs $V(x_n, y)$.
- $G_{\mathsf{P}}^{(n)}$ takes a string $y \in \{0,1\}^{2^{\ell(n)}}$ and a string $z \in \{0,1\}^{c_1\ell(n)}$ as input, and outputs $G_{\ell(n),n^{b/c_1}}^{\mathsf{Umans}}(y, z)$.

From Theorem 3.2, for every good n, there is an advice $x_n \in \{0,1\}^n$ such that G_n is an NPRG fooling n^{b/c_1}-size n^{b/c_1}-input circuits with error $1/10$.

Applying Lemma 6.1 with parameter $2k$. There are constants $t, c \in \mathbb{N}$ and a language $L^{\mathsf{hard}} \in \mathsf{MATIME}[n^t]_{/1}$ such that the following holds: For every sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either $\mathsf{SIZE}(L^{\mathsf{hard}}_n) > n^{2k}$ or $\mathsf{SIZE}(L^{\mathsf{hard}}_m) > m^{2k}$ for some $m \in (n^c, 2n^c) \cap \mathbb{N}$.

Hence, for a sufficiently large good n, let $n_1 = n_1(n)$ be the smallest power of 2 that is at most n. We have either

(1) $\mathsf{SIZE}(L^{\mathsf{hard}}_{n_1}) \geq n_1^{2k}$ or

(2) $\mathsf{SIZE}(L^{\mathsf{hard}}_m) \geq m^{2k}$ for some $m \in (n_1^c, 2 \cdot n_1^c)$.

We now set $b \gg t \cdot c \cdot c_1$, and consider the following two cases.

(1) holds for infinitely many good n's. In this case, we define an $\mathsf{NP}_{/O(n)}$ language given by the following algorithms:

1. On an input $z \in \{0,1\}^n$. We are given two advice bit α_n, β_n and an advice input $x_n \in \{0,1\}^n$. α_n is 1 if n is good and (1) holds for n_1, and is 0 otherwise. When $\alpha_n = 1$, β_n is supposed to be the advice of L^{hard} on n_1-bit inputs, and x_n is supposed to be the advice input such that G_n is an NPRG fooling n^{b/c_1}-size n^{b/c_1}-input circuits with error $1/10$.
2. If $\alpha_n = 0$ we simply output 0. Otherwise, we use G_n with advice x_n, together with the advice β_n for $L^{\mathsf{hard}}_{n_1}$ to compute $L^{\mathsf{hard}}(z_{\leq n_1})$ in non-deterministic $\mathrm{poly}(n)$ time. (We can use G_n to derandomize the n_1^t-time $\mathsf{MA}_{/1}$ algorithm for $L^{\mathsf{hard}}_{n_1}$ because we have set $b \gg t \cdot c \cdot c_1$ and hence $n^{b/c_1} \gg (n_1)^t$.)

(2) holds for infinitely many good n's. In this case, we define an $\mathsf{NP}_{/O(n)}$ language given by the following algorithms:

1. On an input $z \in \{0,1\}^m$. We are given two advice bit α_m, β_m and an advice integer $n \leq m$ and an advice string $x_n \in \{0,1\}^n$. α_m is 1 there is a good $n \in \mathbb{N}$ such that $m \in (n_1^c, 2 \cdot n_1^c)$ and $\mathsf{SIZE}(L^{\mathsf{hard}}_m) > m^k$. When $\alpha_m = 1$, β_m is supposed to be the advice of L^{hard} on m-bit inputs, and x_n is supposed to be the advice input that G_n is an NPRG fooling $n^{b/c}$-size $n^{b/c}$-input circuits with error $1/10$.
2. If $\alpha_m = 0$ we simply output 0. Otherwise, we use G_n with advice x_n, together with the advice β_m for L^{hard}_m to compute $L^{\mathsf{hard}}(z)$ in non-deterministic $\mathrm{poly}(n)$ time. (We can use G_n to derandomize the m^t-time $\mathsf{MA}_{/1}$ algorithm for L^{hard}_m because we have set $b \gg t \cdot c \cdot c_1$ and hence $n^{b/c_1} \gg (2n_1)^{t \cdot c} \geq m^t$.

We can see that in both cases above, there is an $\mathsf{NP}_{/O(n)}$ language that cannot be computed by $\Omega(n^{2k})$-size circuits. By Lemma 6.2, we have $\mathsf{NP} \not\subset \mathsf{SIZE}[n^k]$ and this completes the proof. ∎

7. Average-Case "Almost" Almost Everywhere Lower Bounds for MA∩coMA.

In this section, we prove the average-case circuit lower bounds for $\mathsf{MA} \cap \mathsf{coMA}$, which is the most important technical component of the paper.

We will need the following lemma, which is a direct corollary of Theorem 3.8.

LEMMA 7.1. *For all $a \in \mathbb{N}$, there is $h \in \mathbb{N}$ and a language $L^{\mathsf{diag}} \in \mathit{SPACE}(2^{\log^h n})$ such that for all sufficiently large n,*

$$\mathit{Avg}_{0.99}\text{-}\mathit{SIZE}(L^{\mathsf{diag}}_n) > 2^{\log^a n} \quad \text{and} \quad \mathit{Avg}_{0.99}\text{-}\mathit{DEPTH}(L^{\mathsf{diag}}_n) > \log^a n.$$

Now we are ready to prove the technical centerpiece of the paper, an $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$ language that has a low-depth computable predicate and is average-case hard for low-depth circuits.

THEOREM 7.2. *For all $a \in \mathbb{N}$, there are $b, c \in \mathbb{N}$ and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}(2^{O(\log^b n)})_{/1}$ such that the following hold:*

1. *For all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*
   - $\mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_n) > \log^a n$, *or*
   - $\mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_m) > \log^a m$, *for an $m \in (2^{\log^c n}, 2^{\log^c n+1}) \cap \mathbb{N}$.*
2. *The randomness part of the predicate of L is computable by $O(\log^b n)$-depth circuits.*
3. *For every $n \in \mathbb{N}$, if the advice for L on n-bit inputs is 0, then L_n is the all-zero function.*

Algorithm 7.1: The $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}(2^{O(\log^b n)})_{/1}$ algorithm A_L

Given an input x with length $n = |x|$;
Given an advice integer $\alpha = \alpha_n \in \{0, 1\}$;
Let $m = \lceil 2^{\log^c n} \rceil$;
Let $n_0 = n_0(n)$ be the largest integer such that $2^{\log^c n_0} \le n$;
Let $m_0 = 2^{\log^c n_0}$;
Let $\ell = n - m_0$;
if $\alpha = 0$ **then**
    Output 0 and terminate
if *n is a power of 2* **then**
    `// We are in the case that` $\mathsf{DEPTH}(L_m^{\mathsf{PSPACE}}) \le \log^b n$.
    Compute a z in $2^{O(\log^c n)}$ time such that $L_n^{\mathsf{diag}}(x) = L_m^{\mathsf{PSPACE}}(z)$;
    Guess a circuit C of $\log^b n$ depth;
    Compute in $\mathrm{poly}(m)$ time a TC^0 oracle circuit D_{checker} that implements the instance checker for L_m^{PSPACE};
    Flip an appropriate number of random coins, let them be r;
    Output $D_{\mathsf{checker}}^C(z, r)$;
else
    `// We are in the case that` $\mathsf{DEPTH}(L_{m_0}^{\mathsf{PSPACE}}) > \log^b n_0$ `and` ℓ `is the largest integer such that` $\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \le \log^b n_0$.
    Let z be the first ℓ bits of x;
    Guess a circuit C of $\log^b n_0$ depth;
    Compute in $\mathrm{poly}(\ell)$ time a TC^0 oracle circuit D_{checker} that implements the instance checker for L_ℓ^{PSPACE};
    Flip an appropriate number of random coins, let them be r;
    Output $D_{\mathsf{checker}}^C(z, r)$;

Proof of Theorem 7.2. Let L^{PSPACE} be the language from Theorem 3.7. Applying Lemma 7.1 with parameter a, there is $h \in \mathbb{N}$ and a language $L^{\mathsf{diag}} \in \mathsf{SPACE}(2^{\log^h n})$ such that $\mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_n^{\mathsf{diag}}) > \log^a n$ for all sufficiently large 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 $2^{\log^{c_1} n}$ in $2^{O(\log^{c_1} n)}$ time. We set $c = c_1$ and $b = 3ac$ so that $\log^b n \ge \log^{2a}(2m)$.

The algorithm. Let $\tau \in \mathbb{N}$ be sufficiently large, $n = 2^\tau$, and $m = 2^{\log^c n}$. We first provide an informal description of the $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}(2^{O(\log^b n)})_{/1}$ algorithm

Algorithm 7.2: The algorithm A_{adv} for setting advice bits in Algorithm 7.1

```
1 All the α_n are set to 0 by default;
2 for τ = 1 → ∞ do
3     Let n = 2^τ;
4     Let m = 2^{log^c n};
5     if DEPTH(L_m^PSPACE) ≤ log^b n then
6         Set α_n = 1;
7     else
8         Let ℓ = max{ℓ : DEPTH(L_ℓ^PSPACE) ≤ log^b n};
9         Set α_{m+ℓ} = 1;
```

A_L that computes the language L. There are two cases:

1. When $\mathsf{DEPTH}(L_m^{\mathsf{PSPACE}}) \leq \log^b n$. That is, when L_m^{PSPACE} is *easy*. In this case, on inputs of length n, we guess-and-verify a circuit for L_m^{PSPACE} of depth $\log^b n$, 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 $\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \leq \log^b n$. On input of length $m_1 = m + \ell$, we guess-and-verify a circuit for L_ℓ^{PSPACE}, and compute L_ℓ^{PSPACE} on the first ℓ input bits. Note that by Remark 3.5, we have $0 < \ell < m$ and therefore $m + \ell$ is not a power of 2.

Intuitively, the A_L computes an average-case hard function because either it computes the average-case hard language L_n^{diag} on inputs of length n, or it computes the average-case hard language L_ℓ^{PSPACE} on inputs of length m (L^{PSPACE} is NC^3 weakly error correctable). A formal description of A_L is given in Algorithm 7.1, and the algorithm A_{adv} for setting the advice bits of A_L is given in Algorithm 7.2. Since $m+\ell$ at Line 9 is never a power of 2, α_n can only be set once in Algorithm 7.2.

Now we verify that the algorithm above computes a language satisfying our requirements.

The algorithm satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise. We first show that A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise (Definition 3.9). 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 (Definition 3.4) ensure that in this case A_L satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise. We state the following claim that summarizes the properties of A_L and L that are needed by us.

CLAIM 2. *A_L with advice set by A_{adv} is an $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}(2^{O(\log^b n)})_{/1}$ algorithm for a language L such that, for every $n \in \mathbb{N}$, 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.*

We omit the proof of Claim 2, since it is identical to the proof of Claim 1 in the proof of Lemma 6.1. Also, note that Item (3) of the theorem follows directly from Item (1) of Claim 2.

A_L computes an "almost" almost everywhere average-case hard language for low depth circuits. Next, we show that A_L indeed computes an average-case hard language. Let τ be a sufficiently large integer, $n = 2^\tau$, and $m = 2^{\log^c n}$.

According to Algorithm 7.2, there are two cases:

1. $\mathsf{DEPTH}(L_m^{\mathsf{PSPACE}}) \le \log^b n$. In this case, Algorithm 7.2 sets $\alpha_n = 1$. By Item(2) of Claim 2 and Lemma 7.1, we have $\mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_n) > \log^a n$ as n is sufficiently large.
2. $\mathsf{DEPTH}(L_m^{\mathsf{PSPACE}}) > \log^b n$. Let ℓ be the largest integer such that

$$\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \le \log^b n.$$

By Remark 3.5, we have $\ell < m$. Note that $\mathsf{DEPTH}(L_{\ell+1}^{\mathsf{PSPACE}}) \le d\log(\ell+1) + \mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}})$ for a universal constant d.[27] Therefore,

$$\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \ge \mathsf{DEPTH}(L_{\ell+1}^{\mathsf{PSPACE}}) - d\log(\ell+1) \ge \Omega(\log^b n),$$

the last inequality holds since $\mathsf{DEPTH}(L_{\ell+1}^{\mathsf{PSPACE}}) > \log^b n$, $d\log(\ell+1) \le O(\log \ell) \le O(\log m) \le O(\log^c n)$, and $b = 3ac$.

Now, on inputs of length $m_1 = m + \ell$, we have $\alpha_{m_1} = 1$ by Algorithm 7.2. By Item (3) of Claim 2, it follows that

$$\mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_{m_1}) = \mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}).$$

Since L^{PSPACE} is NC^3 weakly error correctable and the corresponding NC^3 oracle circuit is *non-adaptive*, there is a universal constant d such that

$$\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \le d\log^3 \ell + \mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}).$$

Therefore, recall that $b = 3ac$, it follows

$$\begin{aligned}\mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) &\ge \mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) - d\log^3 \ell \\ &\ge \Omega(\log^b n) - O(\log^{3c} n) \ge \Omega(\log^b n).\end{aligned}$$

Finally, note that $\Omega(\log^b n) \ge \Omega(\log^{2a}(2m)) \ge \log^a(m_1)$. We have

$$\mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_{m_1}) = \mathsf{Avg}_{0.99}\text{-}\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \ge \log^a(m_1).$$

This completes the proof of the Item (1) of the theorem.

The randomness part of the predicate of L. Finally, we have to show that the randomness part of the predicate of L is computable by $O(\log^b n)$-depth circuits (*i.e.*, Item (2) of the theorem). Note that at the end of Algorithm 7.1, given the guessed circuit C and the input x, A_L always first computes a circuit $D_{\mathsf{checker}}^C(z,\cdot)$ in $2^{O(\log^b n)}$ time, and then output $D_{\mathsf{checker}}^C(z,r)$.[28] From Definition 3.9, it suffices for us to argue that $D_{\mathsf{checker}}^C(z,\cdot)$ is an $O(\log^n b)$-depth circuit, which holds since C is of depth at most $\log^b n$ and D_{checker} is in TC^0 (and therefore has an $O(\log n)$-depth circuit). ∎

Finally, we remark that from a proof that is identical to the proof of Theorem 7.2 but working with the complexity measure SIZE instead of the complexity measure DEPTH, the following holds.

[27] Note that L^{PSPACE} is TC^0 downward self-reducible, and the corresponding TC^0 oracle circuit is *non-adaptive*. Also, a TC^0 circuit admits an $O(\log n)$-depth circuit since we can replace each majority gate by an $O(\log n)$-depth circuit.

[28] Unless $\alpha_n = 0$ and A_L simply outputs 0. In this case our claim holds trivially.

THEOREM 7.3. *For all $a \in \mathbb{N}$, there are $b, c \in \mathbb{N}$, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}(2^{O(\log^b n)})_{/1}$, such that the following hold:*
1. *For all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*
   - $\mathsf{Avg}_{0.99}\text{-}\mathsf{SIZE}(L_n) > 2^{\log^a n}$, *or*
   - $\mathsf{Avg}_{0.99}\text{-}\mathsf{SIZE}(L_m) > 2^{\log^a m}$, *for an $m \in (2^{\log^c n}, 2^{\log^c n+1}) \cap \mathbb{N}$.*
2. *The randomness part of the predicate of L is computable by $2^{O(\log^b n)}$-size circuits.*
3. *For every $n \in \mathbb{N}$, if the advice for L on n-bit inputs is 0, then L_n is the all-zero function.*

8. Average-case circuit lower bounds for NQP.

In this section, we first prove that NQP cannot be $(1/2 + 1/\operatorname{polylog}(n))$-approximated by $2^{\operatorname{polylog}(n)}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits (Theorem 1.1) by combining the i.o. NPRG construction from Section 5 with the a.a.e. MA lower bounds from Section 7. We then generalize the average-case lower bounds to all typical circuit classes that admit non-trivial Gap-UNSAT algorithms, and prove Theorem 1.4 and Theorem 1.5.

Notation. We first introduce some notation. For an integer $a \in \mathbb{N}$, we use $\mathsf{bin}(a)$ to denote the Boolean string representing a in binary (from the most significant bit to the least significant bit).

Given two integers $m, n \in \mathbb{N}$, we construct an integer $\mathsf{pair}(m, n)$ as follows. First letting $\ell = |\mathsf{bin}(n)|$, we duplicate each bit in $\mathsf{bin}(\ell)$ and to get a string z_{len} of length $2 \cdot |\mathsf{bin}(\ell)|$ (for example, if $\mathsf{bin}(\ell) = 101$, we get 110011). Then we let $z = \mathsf{bin}(m) \circ \mathsf{bin}(n) \circ 01 \circ z_{\mathsf{len}}$, where $\circ$ means concatenation, and define $\mathsf{pair}(m, n)$ as the integer with binary representation z.

It is easy to see that $\mathsf{pair}(m, n) \le O(mn^2)$. Also, given the integer $a = \mathsf{pair}(m, n)$, one can decode the pair of numbers m and n in $\operatorname{poly}(|\mathsf{bin}(a)|)$ time.

8.1. $(1-\delta)$ Average-Case Lower Bounds for NQP from NPRGs and MA Lower Bounds.

For a typical circuit class $\mathscr{C}$, we first define the following two conditions.

DEFINITION 8.1 (i.o. NPRG condition). *For a typical circuit class $\mathscr{C}$, we say that the i.o. NPRG condition holds for $\mathscr{C}$, if for every $a \in \mathbb{N}_{\ge 1}$, there is $b \in \mathbb{N}$ and an NPRG family $G = \{G_n\}$ such that*
1. *For infinitely many n and $S = 2^{\log^a n}$, G_n is an NPRG for S-size S-input $\mathscr{C}$ circuits with error $1/S$.*
2. *G is computable in $2^{\log^b n}$ time and has seed length $\log^b n$.*

DEFINITION 8.2 (a.a.e. average-case hardness condition). *For a typical circuit class $\mathscr{C}$, we say that the a.a.e. average-case hardness condition holds for $\mathscr{C}$, if for every $a \in \mathbb{N}_{\ge 1}$, there are $b, c \in \mathbb{N}$ and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}(2^{O(\log^b n)})_{/1}$ such that the following hold:*
1. *For all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*
   - $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) > 2^{\log^a n}$, *or*
   - $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_m) > 2^{\log^a m}$ *for some $m \in (2^{\log^c n}, 2^{\log^c n+1}) \cap \mathbb{N}$.*
2. *The randomness part of the predicate of L is computable by $2^{O(\log^b n)}$-size $\mathscr{C}$ circuits.*
3. *For every $n \in \mathbb{N}$, if the advice for L on n-bit inputs is 0, then L_n is the all-zero function.*

The following is an immediate corollary of Theorem 7.2 and Theorem 7.3.

COROLLARY 8.3. *For $\mathscr{C} \in \{\textsf{Formula}, \textsf{Circuit}\}$, the a.a.e. average-case hardness condition holds for $\mathscr{C}$.*

Now we show that the i.o. NPRG condition and a.a.e. average-case hardness condition for a typical circuit class $\mathscr{C}$ imply average-case lower bounds against $\mathscr{C}$.

THEOREM 8.4. *For a typical circuit class $\mathscr{C}$, if both the i.o. NPRG condition and the a.a.e. average-case hardness condition hold for $\mathscr{C}$, then for every $a \in \mathbb{N}$, there is $b \in \mathbb{N}$, a universal constant $\delta \in (0, 1/2)$, and a language $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ such that L cannot be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits.*

Proof. Let b be an integer to be specified later and $\delta = 0.01$. Without loss of generality, we can assume that a is large enough.

Since the a.a.e average-case hardness condition holds for $\mathscr{C}$. There are $b_1, c \in \mathbb{N}$ and a language $L^{\mathsf{hard}} \in (\mathsf{MA}\cap\mathsf{coMA})\mathsf{TIME}(2^{\log^{b_1} n})_{/1}$ such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either

- $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L^{\mathsf{hard}}_n) > 2^{\log^{2a} n}$, or
- $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L^{\mathsf{hard}}_m) > 2^{\log^{2a} m}$ for some $m \in (2^{\log^c n}, 2^{\log^c n+1}) \cap \mathbb{N}$.

Let $T_1(n) = 2^{\log^{b_1} n}$, $A^{\mathsf{hard}}(x, y, z)$ be the predicate of L^{hard}, and $\{\alpha^{\mathsf{hard}}_n\}$ be the advice sequence of L^{hard}. Let c^{hard} be the constant so that $c^{\mathsf{hard}} \cdot T_1(n)$ is the length of y and z in A^{hard}. Let $n_{\mathsf{w}} = n_{\mathsf{w}}(n) = c^{\mathsf{hard}} \cdot T_1(n)$ for convenience.

Moreover, since the randomness part of the predicate of L^{hard} is computable by $2^{O(\log^b n)}$-size $\mathscr{C}$ circuits (see Definition 3.9), there is an $O(2^{\log^{b_1} n})$-time algorithm B^{hard} such that:

- Given an input $x \in \{0,1\}^n$, a witness $y \in \{0,1\}^{n_{\mathsf{w}}(n)}$, and the correct advice $\alpha = \alpha^{\mathsf{hard}}_n \in \{0,1\}$, $B^{\mathsf{hard}}_{/\alpha}(x, y)$ outputs an $n_{\mathsf{w}}(n)$-input $2^{O(\log^b n)}$-size $\mathscr{C}$ circuit D, such that $A^{\mathsf{hard}}_{/\alpha}(x, y, z) = D(z)$ for all $z \in \{0,1\}^{n_{\mathsf{w}}(n)}$.[29]

Now we try to derandomize L^{hard} non-deterministically and get a hard language in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$. In the following we always assume that n is sufficiently large.

Let $a_1 = \max(2b_1c^2, a)$ and $S = S(n) = 2^{\log^{a_1} n}$. Since the i.o. NPRG condition holds for $\mathscr{C}$, there is $b_2 \in \mathbb{N}$ and an NPRG family $G = \{G_n\}$ such that:

1. For infinitely many n, G_n is an NPRG for S-size S-input $\mathscr{C}$ circuits with error $1/S$.
2. G is computable in $2^{\log^{b_2} n}$ time and has seed length $\log^{b_2} n$.

We call an integer n good if the Item (1) above holds for n.

Now, fix a good n. Let n_1 be the largest power of 2 that is at most n. We first provide an informal description of our $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ algorithm for our hard language L. There are two cases according to Theorem 7.2.

- Case I: $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L^{\mathsf{hard}}_{n_1}) > 2^{\log^{2a} n_1}$. In this case, on inputs of length n, we apply the NPRG G_n to compute $L^{\mathsf{hard}}_{n_1}$ on the first n_1 bits in $2^{O(\log^{b_2} n)}$ non-deterministic time.
- Case II: $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L^{\mathsf{hard}}_m) > 2^{\log^{2a} m}$ for some $m \in (2^{\log^c n_1}, 2^{\log^c n_1+1}) \cap \mathbb{N}$. In this case, on inputs of length $n_2 = \mathsf{pair}(m, n) \le O(mn^2)$, we apply the NPRG G_n to compute L^{hard}_m on the first m bits in $2^{O(\log^{b_2} n)} \le 2^{O(\log^{b_2} n_2)}$ non-deterministic time.

Formally, the algorithm is specified in Algorithm 8.1, with a key subroutine Derand

[29] We use $A^{\mathsf{hard}}_{/\alpha}$ and $B^{\mathsf{hard}}_{/\alpha}$ to denote that the advice of these two algorithms are set to α.

given in Algorithm 8.2. The advice bits α_n and β_n are set by Algorithm 8.3.[30]

Algorithm 8.1: The $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ Algorithm A_L

```
Given an input x with length n = |x|;
Given advice bits α = α_n ∈ {0,1} and β = β_n ∈ {0,1};
if β = 0 then
    return 0 ;
if α = 0 then
    Let n_1 be the largest power of 2 that is at most n;
    // α = 0 indicates we are in the case that
        Avg_{0.99}-𝒞-SIZE(L^hard_{n_1}) > 2^{log^{2a} n_1} and n is good.
    Let w be the first n_1 bits of x;
    return Derand(w, n);
else
    Parse n as two integers (m_0, n_0) (that is, n = pair(m_0, n_0));
    // α = 1 indicates we are in the case that
        Avg_{0.99}-𝒞-SIZE(L^hard_{m_0}) > 2^{log^{2a} m_0} and n_0 is good.
    Let w be the first m_0 bits of x;
    return Derand(w, n_0);
```

Algorithm 8.2: $\mathsf{Derand}(x, n_0)$

```
Given an input x with length n = |x|, n_0;
// n_0 is suppose to be good.
Guess an n_w(n)-bit witness y and run B^hard_{/1}(x, y) to obtain an n_w(n)-input
    2^{O(log^{b_1} n)}-size 𝒞 circuit D_{x,y};
Let (G_P, G_W) be the pair of algorithm in the NPRG G_{n_0};
Guess a string y_hard ∈ {0,1}^{2^{log^{b_2} n_0}};
if G_W(y_hard) = 0 then
    return ⊥;
for w ∈ {0, 1, ⊥} do
    p_w = Pr_{r ∈_R {0,1}^{log^{b_2} n_0}} [D_{x,y}(G_P(y_hard, r)) = w];
if p_1 > 0.6 then
    return 1;
if p_0 > 0.6 then
    return 0;
return ⊥;
```

Analysis of Derand(x, z, n_0). We first prove the following claim regarding the algorithm $\mathsf{Derand}(x, n_0)$.

[30]Here it is possible that an α_n or β_n is set twice by Algorithm 8.3, but this does not affect our analysis.

Algorithm 8.3: The algorithm A_{adv} for setting advice bits of Algorithm 8.1

All α_n's and β_n's are set to 0 by default;
for $n = 1 \to \infty$ **do**
  if *n is good* **then**
    Let n_1 be the largest power of 2 that is at most n;
    if $\mathit{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathit{SIZE}(L^{\mathsf{hard}}_{n_1}) > 2^{\log^{2a} n_1}$ **then**
      $\alpha_n = 0$;
      $\beta_n = \alpha^{\mathsf{hard}}_{n_1}$;
    else
      Let m be the smallest integer from $(2^{\log^c n_1}, 2^{\log^c n_1 + 1}) \cap \mathbb{N}$ such that $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L^{\mathsf{hard}}_m) > 2^{\log^{2a} m}$;
      $n_2 = \mathsf{pair}(m, n)$;
      $\alpha_{n_2} = 1$;
      $\beta_{n_2} = \alpha^{\mathsf{hard}}_m$;

Claim 3. *For $n, n_0 \in \mathbb{N}$, if n_0 is good, $\alpha^{\mathsf{hard}}_n = 1$ and $\log^{a_1} n_0 \ge \log^{b_1+1} n$, then for every $x \in \{0,1\}^n$, $\mathsf{Derand}(x, n_0)$ computes $L^{\mathsf{hard}}(x)$ with respect to Definition 3.11.*

Proof. Fix an $x \in \{0,1\}^n$. Since $\alpha^{\mathsf{hard}}_n = 1$, we have that $B^{\mathsf{hard}}_{/1}(x, y)$ outputs an $n_{\mathsf{w}}(n)$-input $2^{O(\log^{b_1} n)}$-size circuit $D_{x,y}$ such that $A^{\mathsf{hard}}_{/1}(x, y, z) = D_{x,y}(z)$ for all $z \in \{0,1\}^{n_{\mathsf{w}}(n)}$.

Since $A^{\mathsf{hard}}(x, y, z)$ is the predicate of an $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T_1(n)]_{/1}$ time algorithm for L^{hard}, we have (1) there exists $y \in \{0,1\}^{n_{\mathsf{w}}(n)}$ such that $D_{x,y}(z) = L^{\mathsf{hard}}(z)$ for all $z \in \{0,1\}^{n_{\mathsf{w}}(n)}$ and (2) for all $y \in \{0,1\}^{n_{\mathsf{w}}(n)}$, $D_{x,y}(z) = 1 - L^{\mathsf{hard}}(z)$ happens with probability at most $1/3$ over z.

Now, since n_0 is good, we further know that (1) there is $y_{\mathsf{hard}} \in \{0,1\}^{2^{\log^{b_2} n_0}}$ such that $G_{\mathsf{W}}(y_{\mathsf{hard}}) = 1$ and (2) for all y_{hard} such that $G_{\mathsf{W}}(y_{\mathsf{hard}}) = 1$, $G_{\mathsf{P}}(y_{\mathsf{hard}}, \cdot)$ is a PRG fooling $\mathscr{C}$ circuits of $2^{\log^{a_1} n_0} \ge 2^{\log^{b_1+1} n}$ size with error at most $1/100$.

Finally, we show Derand computes $L^{\mathsf{hard}}(x)$ with respect to Definition 3.11. First, there exists $y \in \{0,1\}^{n_{\mathsf{w}}(n)}$ and $y_{\mathsf{hard}} \in \{0,1\}^{2^{\log^{b_2} n_0}}$ such that $D_{x,y}(z) = L^{\mathsf{hard}}(z)$ for all $z \in \{0,1\}^{n_{\mathsf{w}}(n)}$ and $G_{\mathsf{W}}(y_{\mathsf{hard}}) = 1$. Since $D_{x,y}$ is of size $2^{O(\log^{b_1} n)} \le 2^{\log^{b_1+1} n}$, we know that $G_{\mathsf{P}}(y_{\mathsf{hard}}, \cdot)$ fools $D_{x,y}$ with error at most $1/100$.[31] This in particular means that $p_{L^{\mathsf{hard}}(x)} \ge 0.99$, and $\mathsf{Derand}(x, n_0)$ outputs $L^{\mathsf{hard}}(x)$ on the guess y and y_{hard}.

Second, for all $y \in \{0,1\}^{n_{\mathsf{w}}(n)}$ and $y_{\mathsf{hard}} \in \{0,1\}^{2^{\log^{b_2} n_0}}$. We know that $D_{x,y}(z) = 1 - L^{\mathsf{hard}}(z)$ happens with probability at most $1/3$ over z. Now, if $G_{\mathsf{W}}(y_{\mathsf{hard}}) = 0$, $\mathsf{Derand}(x, n_0)$ outputs $\bot$ immediately. Otherwise, $G_{\mathsf{W}}(y_{\mathsf{hard}}) = 1$, and $G_{\mathsf{P}}(y_{\mathsf{hard}}, \cdot)$ fools $D_{x,y}$ with error at most $1/100$. This in particular means that $p_{1 - L^{\mathsf{hard}}(x)} \le 0.35$, and hence $\mathsf{Derand}(x, n_0)$ does not output $1 - L^{\mathsf{hard}}(x)$ on the guess y and y_{hard}.

To summarize, for every $x \in \{0,1\}^n$, (1) there are $y \in \{0,1\}^{n_{\mathsf{w}}(n)}$ and $y_{\mathsf{hard}} \in \{0,1\}^{2^{\log^{b_2} n_0}}$ such that $\mathsf{Derand}(x, n_0)$ outputs $L^{\mathsf{hard}}(x)$ and (2) for every y and y_{hard}, $\mathsf{Derand}(x, n_0) \in \{L^{\mathsf{hard}}(x), \bot\}$. This completes the proof. ∎

Analysis of A_L. Next we prove the following claim regarding A_L.

[31] More formally, for every $w \in \{0, 1, \bot\}$, it fools the circuit deciding whether $D_{x,y}(z) = w$.

CLAIM 4. *A_L with advice set by A_{adv} is an $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ algorithm for a language L defined as follows:*

1. *If $\beta_n = 0$, then L_n is the all-zero function.*
2. *If $\beta_n = 1$ and $\alpha_n = 0$, then L_n is the n-input function that computes $L^{\mathsf{hard}}_{n_1}$ on the first n_1 bits and ignores the rest of the input.*
3. *If $\beta_n = 1$ and $\alpha_n = 1$, then L_n is the n-input function that computes $L^{\mathsf{hard}}_{m_0}$ on the first m_0 bits and ignores the rest of the input.*

Proof. Item (1) of the theorem follows immediately from Line 4 of Algorithm 8.1. In the following we show Item (2) and Item (3) hold separately.

We first consider Item (2). In this case, we have that $\beta_n = 1$ and $\alpha_n = 0$. By Algorithm 8.3, it follows that n is good and $\alpha^{\mathsf{hard}}_{n_1} = 1$. Since n_1 is the largest power of 2 that is at most n, we have $n_1 \le n$. Recall that $a_1 = \max(2b_1c^2, a)$, we have $\log^{a_1} n \ge \log^{b_1+1} n \ge \log^{b_1+1} n_1$. Hence, by Claim 3, it follows that $\mathsf{Derand}(w, n)$ computes $L^{\mathsf{hard}}(w)$ with respect to Definition 3.11 at Line 8. This proves Item (2).

We next consider Item (3). Now we have $\beta_n = 1$ and $\alpha_n = 1$. By Algorithm 8.3, it follows that (1) n_0 is good and $\alpha^{\mathsf{hard}}_{m_0} = 1$ and (2) $m_0 \le 2^{\log^c n_0 + 1}$. By our choice of a_1, we have that $\log^{a_1} n_0 \ge \log^{b_1+1} m_0$. Again by Claim 3, it follows that $\mathsf{Derand}(w, n_0)$ computes $L^{\mathsf{hard}}(w)$ with respect to Definition 3.11 at Line 12. This proves Item (3). ∎

Average-case lower bound. Finally, we are ready to prove that L is average-case hard, which completes the proof.

CLAIM 5. *L cannot be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits.*

Proof. Since there are infinitely many good n, either Line 7 or Line 12 of Algorithm 8.3 is executed for an infinite number of times. Moreover, from Item (3) of Definition 8.2, it follows that at Line 7 (resp. Line 12), $\alpha^{\mathsf{hard}}_{n_1}$ (resp. $\alpha^{\mathsf{hard}}_{m}$) must be 1.[32] Hence, we know that there are infinitely many n such that $\beta_n = 1$. We now consider the following two cases.

Case I. There are infinitely many n such that $\beta_n = 1$ and $\alpha_n = 0$. By Algorithm 8.3 and Claim 4, we know that

$$\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) \ge \mathsf{Avg}_{0.99}\text{-}\mathsf{SIZE}\mathscr{C}(L^{\mathsf{hard}}_{n_1}) \ge 2^{\log^{2a} n_1} \ge 2^{\log^a n}.$$

The last inequality above follows from the fact that $n_1 \ge n/2$.

Case II. There are infinitely many n such that $\beta_n = 1$ and $\alpha_n = 1$. By Algorithm 8.3 and Claim 4, we know that

$$\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) \ge \mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L^{\mathsf{hard}}_{m_0}) \ge 2^{\log^{2a} m_0}$$

. Moreover, from Algorithm 8.3 we also have $m_0 \le n \le O(m_0 n_0^2)$ and $m_0 \ge 2^{\log^c (n_0/2)}$. Hence, we have $m_0 \ge n^{0.99}$ and $2^{\log^{2a} m_0} \ge 2^{\log^a n}$ since a is large enough.

Hence, in both cases, we have that $\mathsf{Avg}_{0.99}\text{-}\mathscr{C}\text{-}\mathsf{SIZE}(L_n) \ge 2^{\log^a n}$ for infinitely many n. ∎

8.2. $(1-\delta)$ **Average-Case Lower Bounds for NQP from Non-trivial Derandomization.** Recall that for a typical circuit class $\mathscr{C}$, we say the non-trivial derandomization condition holds for $\mathscr{C}$, if there is $\varepsilon \in (0,1)$ such that the $\mathsf{Gap}\text{-}\mathsf{UNSAT}$

[32] Since if $\alpha^{\mathsf{hard}}_{n_1} = 0$, then $L^{\mathsf{hard}}_{n_1}$ is the trivial all-zero function. The same holds for $\alpha^{\mathsf{hard}}_{m}$ and L^{hard}_{m} as well.

problem for 2^{n^ε}-size n-input $\mathscr{C}$ circuits can be solved in $2^n/n^{\omega(1)}$ non-deterministic time.

Recall that a circuit class $\mathscr{C}$ is nice, if it is typical and either $\mathscr{C} = \mathsf{Circuit}$ or $\mathscr{C}$ is weaker than $\mathsf{Formula}$.

From Theorem 5.4 and Theorem 5.3, we have the following corollary.

Corollary 8.5. *Let $\mathscr{C}$ be a typical circuit class such that the non-trivial derandomization condition holds for $\mathsf{AC}_5 \circ \mathscr{C}$. There is a universal constant $\delta \in (0, 1/2)$ such that the following hold:*

1. *If uniform NC^1 can be $(1-\delta)$-approximated by $2^{\log^c n}$-size $\mathscr{C}$ circuit families for some $c \in \mathbb{N}$, then the i.o. NPRG condition holds for $\mathsf{Formula}$.*
2. *If $\mathscr{C} = \mathsf{Circuit}$, then the i.o. NPRG condition holds for $\mathsf{Circuit}$.*

Next we show that, for a nice circuit class $\mathscr{C}$, the non-trivial derandomization condition for $\mathscr{C}$ implies average-case lower bounds against $\mathscr{C}$.

Theorem 8.6. *Let $\mathscr{C}$ be a nice circuit class. Suppose the non-trivial derandomization condition holds for $\mathsf{AC}_5 \circ \mathscr{C}$. Then for every $a \in \mathbb{N}$, there is $b \in \mathbb{N}$, a universal constant $\delta \in (0, 1/2)$, and a language $L \in \mathsf{(N{\cap}coN)TIME}[2^{\log^b n}]_{/2}$ such that L cannot be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits.*

Proof. The case for $\mathscr{C} = \mathsf{Circuit}$ follows directly from Corollary 8.5, Corollary 8.3 and Theorem 8.4. So we will focus on the case that $\mathscr{C}$ is weaker than $\mathsf{Formula}$. We will consider the following two cases.

Case I. If uniform NC cannot be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits. In this case, since uniform NC is contained in $\mathsf{(N{\cap}coN)TIME}[2^{\log^2 n}]_{/2}$, we can simply set $b = 2$.

Case II. If uniform NC can be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits. In this case, by Item (1) of Corollary 8.5, the i.o. NPRG condition holds for $\mathsf{Formula}$.

Now, by Corollary 8.3 and Theorem 8.4, there is $b \in \mathbb{N}$ and a language $L \in \mathsf{(N{\cap}coN)TIME}[2^{\log^b n}]_{/2}$ that cannot be $(1-\delta)$-approximated by $2^{\log^{a+1} n}$-size formulas. Since $\mathscr{C}$ is weaker than $\mathsf{Formula}$, it follows that L also cannot be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits. ∎

Recall the the following SAT algorithm for $\mathsf{AC}_d[m] \circ \mathsf{THR}$ by [65].

Theorem 8.7 ([65]). *For every $d, m \in \mathbb{N}$, there is an $\varepsilon = \varepsilon(d, m) > 0$ such that the satisfiability of a 2^{n^ε}-size n-input $\mathsf{AC}_d[m] \circ \mathsf{THR}$ circuit can be determined deterministically in 2^{n-n^ε} time.*

In other words, the non-trivial derandomization condition holds for $\mathsf{AC}_d[m] \circ \mathsf{THR}$, for every $d, m \in \mathbb{N}$.

Combining Theorem 8.7 with Theorem 8.6, we immediately have the following average-case lower bounds against $\mathsf{AC}_d[m] \circ \mathsf{THR}$.

Corollary 8.8. *For every $a, d_\star, m_\star \in \mathbb{N}$, there is $b \in \mathbb{N}$, a universal constant $\delta > 0$, and a language $L \in \mathsf{(N{\cap}coN)TIME}[2^{\log^b n}]_{/2}$ such that L cannot be $(1-\delta)$-approximated by $2^{\log^a n}$ size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits.*

Next we show Corollary 8.8 indeed imply the following stronger lower bounds. We remark that we cannot directly apply Theorem 8.6 to $\mathsf{ACC}^0 \circ \mathsf{THR}$, since the non-trivial derandomization condition does not necessarily hold for $\mathsf{ACC}^0 \circ \mathsf{THR}$. Our proof below uses a case-analysis to resolve this issue.

COROLLARY 8.9. *For every $a \in \mathbb{N}$, there is $b \in \mathbb{N}$, a universal constant $\delta > 0$, and a language $L \in (\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ such that L cannot be $(1-\delta)$-approximated by $2^{\log^a n}$ size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.*[33]

Proof. Let $b \geq 1$ be an integer to be specified later, and δ be the minimum of the universal constants in Corollary 8.8 and Theorem 4.3.

For the sake of contradiction, suppose every language in $(\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ can be $(1-\delta)$-approximated by a $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuit family.

In particular, there are $d_\circ, m_\circ \in \mathbb{N}$ such that for every $i \in [120]$ $\mathsf{Redundant\text{-}W}_{S_5}^{(i)}$ can be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathsf{AC}_{d_\circ}[m_\circ] \circ \mathsf{THR}$ circuits. Therefore, by Theorem 4.3, there is a constant $c_e \geq 1$ such that any depth-d circuit has an equivalent $2^{c_e \cdot d^a}$-size $\mathsf{AC}_{d_\circ+3}[m_\circ] \circ \mathsf{THR}$ circuit. Hence, any depth-$\log^{2a} n$ circuit has an equivalent $2^{c_e \cdot \log^{2a^2} n}$-size $\mathsf{AC}_{d_\circ+3}[m_\circ] \circ \mathsf{THR}$ circuit.

Finally, by Corollary 8.8, there is a language $L \in (\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ (now we set b) such that L cannot be $(1-\delta)$-approximated by $2^{\log^{2a^2+1} n}$-size $\mathsf{AC}_{d_\circ+3}[m_\circ] \circ \mathsf{THR}$ circuits. By the previous discussion, it follows that L cannot be $(1-\delta)$-approximated by $\log^{2a} n$-depth circuits. Consequently, L cannot be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, a contradiction. ∎

8.3. $1/2 + 1/\operatorname{polylog}(n)$ **Average-Case Lower Bounds against $\mathsf{ACC}^0 \circ \mathsf{THR}$.** Now we are ready to prove our main theorem Theorem 1.1 from Corollary 8.9 and Lemma 3.14.

We first prove the following lemma, which gives us a convenient way to apply hardness amplification to languages in $(\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$.

LEMMA 8.10. *For every $b \geq 2$ and every language $L \in (\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$, there is a language $L' \in (\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ such that, for every typical circuit class $\mathscr{C}$ and two nondecreasing unbounded functions $S, \ell \colon \mathbb{N} \to \mathbb{N}$ such that $\ell(n) \leq 2^{o(n)}$, and for every constant $\delta_0 \in (0, 1/2)$, the following holds:*

- *If L cannot be $(1-\delta_0)$-approximated by $O(\ell(n)S(n))$-size $\mathsf{MAJ}_{\ell(n)} \circ \mathscr{C}$ circuits, then L' cannot be $(1/2 + \ell(n^{1/3})^{-1/3})$-approximated by $S(n^{1/3})$-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 $m^2 \leq n$, and $k = \min(n - m^2, m)$, we define $L'(x) = L^{\oplus k}(x_{\leq km})$, where $x_{\leq km}$ denotes the first km bits of x. (Since $k \leq m$, we have $km \leq m^2 \leq n$.) Using the straightforward algorithm for computing L', it follows that $L' \in (\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$.[34]

Now, suppose for a constant $\delta_0 \in (0, 1/2)$, there are infinitely many n such that L_n cannot be $(1-\delta_0)$-approximated by $\ell(n)S(n)$-size $\mathsf{MAJ}_{\ell(n)} \circ \mathscr{C}$ circuits. We call these n good. Without loss of generality we can assume $\delta_0 \in (0, 0.01)$. We also set $\delta = \delta_0/5$.

For every sufficiently large good n, we set $k = k(n)$ to be the first k so that $\varepsilon_k^{-1} \geq \ell(n)^{1/3}$, where $\varepsilon_k = (1-\delta)^{k-1}(1/2 - \delta)$. Let c_1 be the universal constant in Lemma 3.14. Since n is sufficiently large and ℓ is unbounded and nondecreasing, $\ell_0 = c_1 \frac{\log \delta^{-1}}{\varepsilon_k^2} < \ell(n)$. Now, by Lemma 3.14 and the fact that L_n cannot be $(1 - 5\delta)$-

[33] In other words, L cannot be $(1-\delta)$-approximated by $2^{\log^a n}$ size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits, for every $d_\star, m_\star \in \mathbb{N}$.

[34] We remark that this step crucially uses the fact that L is in $(\mathsf{N}{\cap}\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ instead of $\mathsf{NTIME}[2^{\log^b n}]_{/2}$.

approximated by $\ell_0 \cdot S(n) + 1 \le \ell(n) \cdot S(n)$-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 $\varepsilon_k \le \ell(n)^{-1/3}$ from our choice) by $S(n)$-size $\mathscr{C}$ circuits.

From our definition of L', it follows that for infinitely many n, $L'_{n^2+k(n)}$ (from our choice of k and the assumption that $\ell(n) = 2^{o(n)}$, we have that $k \le n$) 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. ∎

Then we apply Lemma 8.10 to amplify the $(1 - \delta)$-average-case lower bound from Corollary 8.9 to a $(1/2 + 1/\operatorname{polylog}(n))$-average-case lower bound.

Lemma 8.11. *For every $a, c \in \mathbb{N}$, there is $b \in \mathbb{N}$ and $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ such that L cannot be $(1/2 + 1/\log^c n)$-approximated by $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.*

Proof. Let $a_1 = \max(5c, a + 1)$. By Corollary 8.9, there is $b_1 \in \mathbb{N}$ and a language $L_1 \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^{b_1} n}]_{/2}$ such that L_1 cannot be $(1-\delta)$-approximated by $2^{\log^{a_1} n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, for a universal constant $\delta \in (0, 1/2)$. Without loss of generality, we can assume that $b_1, a, c \ge 2$.

We apply Lemma 8.10 to L_1 to get our language $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^{b_1} n}]_{/2}$. We also let $\ell(n) = \log^{4c} n$ and $S(n) = 2^{\log^{a_1} n - 1}$. Now, for every $d_\star, m_\star \in \mathbb{N}$, we note that an $\mathsf{MAJ}_{\ell(n)} \circ \mathsf{AC}_{d_\star}[m_\star]$ circuit of size $S(n)$ has an equivalent $\mathsf{AC}_{d_\star+2}[m_\star]$ circuit of size $S(n) + 2^{\ell(n)} < 2^{\log^{a_1} n}$, by replacing the top $\mathsf{MAJ}_{\ell(n)}$ gate by an AC_2 circuit of size at most $2^{\ell(n)}$. Hence, since L_1 cannot be $(1 - \delta)$-approximated by $2^{\log^{a_1} n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, it also follows that L_1 cannot be $(1 - \delta)$-approximated by $S(n)\ell(n)$-size $\mathsf{MAJ}_{\ell(n)} \circ \mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits. By Lemma 8.10, it follows that L cannot be $(1/2 + \ell(n^{1/3})^{-1/3})$-approximated by $S(n^{1/3})$-size $\mathsf{AC}_{d_\star}[m_\star]$ circuits.

Finally, note that for a sufficiently large n, we have $\ell(n^{1/3})^{1/3} \ge \Omega(\log^{4c/3} n) \ge \log^c n$ and $S(n^{1/3}) \ge 2^{\Omega(\log^{a_1} n)} \ge 2^{\Omega(\log^{a+1} n)} \ge 2^{\log^a n}$. It then follows that L cannot be $(1/2 + 1/\log^c n)$-approximated by $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star]$ circuits, for every $d_\star, m_\star \in \mathbb{N}$. This completes the proof. ∎

Next we need the following lemma to get rid or reduce the advice in Lemma 8.11. The same trick was used in [20] as well.

Lemma 8.12. *For every $b \ge 2$ and every language $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$, there are languages $L_1 \in \mathsf{NTIME}[2^{\log^b n}]$ and $L_2 \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$ such that the following holds:*

- *For every typical circuit class $\mathscr{C}$, $S \colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon \colon \mathbb{N} \to (0, 1/2)$, 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 $1/2 + \varepsilon(\lfloor n/4 \rfloor)$-approximated by $S(\lfloor n/4 \rfloor)$-size $\mathscr{C}$ circuits.*

Proof. Let $w_0, w_1, w_2, w_3 \in \{0,1\}^2$ be an enumeration of the set $\{0,1\}^2$. We will prove the lemma for L_1 and L_2 separately.

NQP lower bounds. We first prove the case for $L_1 \in \mathsf{NTIME}[2^{\log^b n}]$. We define $L_1 \in \mathsf{NTIME}[2^{\log^b n}]$ by the following algorithm A_1: on an input of length n, let $n' = \lfloor n/4 \rfloor$ and $k = n - 4 \cdot n'$; A_1 simulates the non-deterministic algorithm for $L'_{n'}$ with the advice w_k on the first n' bits of the input.

Since L cannot be $1/2 + \varepsilon(n)$-approximated by $S(n)$-size $\mathscr{C}$ circuits, there are infinitely many pairs $(n_i, a_i) \in \mathbb{N} \times \{0, 1, 2, 3\}$ such that the non-deterministic algorithm for L_{n_i} with advice w_{a_i} computes a function that cannot be $(1/2 + \varepsilon(n_i))$-

approximated by $S(n_i)$-size $\mathscr{C}$ circuits. By the construction of L_1, $(L_1)_{4n_i+a_i}$ cannot be $(1/2+\varepsilon(n_i))$-approximated by $S(n_i)$-size $\mathscr{C}$ circuits. Therefore, L_1 cannot be $1/2+\varepsilon(\lfloor n/4 \rfloor)$-approximated by $S(\lfloor n/4 \rfloor)$-size $\mathscr{C}$ circuits.

$(\mathsf{NQP} \cap \mathsf{coNQP})_{/1}$ **lower bounds**. Now we define $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$ by the following algorithm A_2: on an input of length n, let $n' = \lfloor n/4 \rfloor$ and $k = n - 4 \cdot n'$; we set the advice bit $\alpha_n = 1$ if and only if w_k is the correct advice for input length n' of language L; when $\alpha_n = 1$, A_2 simulates $L_{n'}$ with the advice w_k on the first n' bits of the input; otherwise, A_2 simply outputs 0. A similar argument as that of the previous case completes the proof. ◻

Now, Theorem 1.1 follows as a direct corollary of Lemma 8.12 and Lemma 8.11.

8.4. Generalization to Other Natural Circuit Classes. Finally, we generalize our average-case lower bounds to other natural circuits $\mathscr{C}$ if the non-trivial derandomization condition holds for them.

Reminder of Theorem 1.4. *Let $\mathscr{C}$ be a nice circuit class. Suppose the non-trivial derandomization condition holds for $\mathsf{AC}_7 \circ \mathscr{C}$. Then for every $a, c \in \mathbb{N}$, there is $b \in \mathbb{N}$, and a language $L \in \mathsf{NTIME}[2^{\log^b n}]$ such that L cannot be $(1/2 + 1/\log^c n)$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits. The same holds for $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$ in place of $\mathsf{NTIME}[2^{\log^b n}]$.*

Proof. We set $a_1 = 3(a+1)(c+1)$. Let $\mathscr{C}_1 = \mathsf{AC}_2 \circ \mathscr{C}$. Note that $\mathscr{C}_1$ is also nice since $\mathscr{C}$ is nice. From our assumption, it follows that the non-trivial derandomization condition holds for $\mathsf{AC}_5 \circ \mathscr{C}_1$. By Theorem 8.6, there is a universal constant $\delta \in (0, 1/2)$, $b_1 \in \mathbb{N}$, and a language $L_1 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{b_1} n}]_{/2}$ such that L_1 cannot be $(1-\delta)$-approximated by $2^{\log^{a_1} n}$-size $\mathsf{AC}_2 \circ \mathscr{C}$ circuits.

Again, we apply Lemma 8.10 to L_1 to get $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{b_1} n}]_{/2}$. We also let $\ell(n) = \log^{4(c+1)} n$ and $S(n) = 2^{\log^{a_1} n - 1}$. Now applying an identical argument as in the proof of Lemma 8.11, it follows that L cannot be $(1/2 + 1/\log^{c+1} n)$-approximated by $2^{\log^{a+1} n}$-size $\mathscr{C}$ circuits. Applying Lemma 8.12 completes the proof. ◻

Reminder of Theorem 1.5. *Let $\mathscr{C}$ be a nice circuit class. Suppose the non-trivial derandomization condition holds for $\mathsf{AC}_5 \circ \mathsf{MAJ} \circ \mathscr{C}$. Then for every $a, c \in \mathbb{N}$, there is $b \in \mathbb{N}$, and a language $L \in \mathsf{NTIME}[2^{\log^b n}]$ such that L cannot be $(1/2 + 1/2^{\log^c n})$-approximated by $2^{\log^a n}$-size $\mathscr{C}$ circuits. The same holds for $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$ in place of $\mathsf{NTIME}[2^{\log^b n}]$.*

Proof. Let $\mathscr{C}_1 = \mathsf{MAJ} \circ \mathscr{C}$. We set $a_1 = 3(a+1)(c+1)$. Note that $\mathscr{C}_1$ is also nice since $\mathscr{C}$ is nice and the non-trivial derandomization condition holds for $\mathsf{AC}_5 \circ \mathscr{C}_1$. By Theorem 8.6, there is a universal constant $\delta \in (0, 1/2)$, $b_1 \in \mathbb{N}$, and a language $L_1 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{b_1} n}]_{/2}$ such that L_1 cannot be $(1-\delta)$-approximated by $2^{\log^{a_1} n}$-size $\mathsf{MAJ} \circ \mathscr{C}$ circuits.

Again, we apply Lemma 8.10 to L_1 to get $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^{b_1} n}]_{/2}$. We also let $\ell(n) = 2^{\log^{4(c+1)} n}$ and $S(n) = 2^{\log^{a_1} n - 1}$. Now applying an identical argument as in the proof of Lemma 8.11, it follows that L cannot be $(1/2 + 1/2^{\log^{c+1} n})$-approximated by $2^{\log^{a+1} n}$-size $\mathscr{C}$ circuits. Applying Lemma 8.12 completes the proof. ◻

9. A PSPACE-complete Language with Nice Reducibility Properties.

In this section, we construct a PSPACE-complete language with the needed nice reducibility properties, and prove Theorem 3.7.

In Subsection 9.1, we introduce the necessary technical preliminaries. In Subsection 9.2, we review the original construction in [59]. In Subsection 9.2.1, we briefly discuss what adaptions are required to make it suitable for our purpose, and we prove some additional properties of the construction of [59] in Subsection 9.2.2. In Subsection 9.3, we construct the needed PSPACE-complete language.

9.1. Preliminaries.

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

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 a simple representation 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 $n = \mathsf{pw}_\ell = 2 \cdot 3^\ell$ for some $\ell \in \mathbb{N}$. We will always represent $\mathbb{F}^{(\ell)} = \mathsf{GF}(2^n)$ as $\mathbb{F}_2[\mathbf{x}]/(\mathbf{x}^n + \mathbf{x}^{n/2} + 1)$.[35] That is, we identify an element of $\mathsf{GF}(2^n)$ with an $\mathbb{F}_2[\mathbf{x}]$ polynomial with degree less than n. To avoid confusion, given a polynomial $P(\mathbf{x}) \in \mathbb{F}_2[\mathbf{x}]$ with degree less than n, we will use $(P(\mathbf{x}))_{\mathbb{F}^{(\ell)}}$ to denote the unique element in $\mathbb{F}^{(\ell)}$ identified with $P(\mathbf{x})$.

The most important property of the fields $\{\mathbb{F}^{(\ell)}\}_{\ell\in\mathbb{N}}$ is that, there is a very simple embedding τ_ℓ of $\mathbb{F}^{(\ell)}$ into $\mathbb{F}^{(\ell+1)}$: τ_ℓ maps $(\mathbf{x})_{\mathbb{F}^{(\ell)}}$ to $(\mathbf{x}^3)_{\mathbb{F}^{(\ell+1)}}$ (this induces a mapping from $\mathbb{F}^{(\ell)}$ to $\mathbb{F}^{(\ell+1)}$).[36] We sometimes abuse notation and identify $\mathbb{F}^{(\ell)}$ as a subset of $\mathbb{F}^{(\ell+1)}$ via the embedding τ_ℓ, and omit the subscript of $(\mathbf{x})_{\mathbb{F}^{(\ell+1)}}$ when the underlying field is clear from the context.

Let $\ell_1, \ell_2 \in \mathbb{N}$ be such that $\ell_1 < \ell_2$, we use $\tau_{\ell_1\to\ell_2}$ to denote the composed mapping $\tau_{\ell_2-1} \circ \cdots \circ \tau_{\ell_1+1} \circ \tau_{\ell_1}$. That is, $\tau_{\ell_1\to\ell_2}$ is an embedding of $\mathbb{F}^{(\ell_1)}$ into $\mathbb{F}^{(\ell_2)}$.

Let $n \in \mathbb{N}$ and $p\colon (\mathbb{F}^{(\ell_1)})^n \to \mathbb{F}^{(\ell_1)}$ be a polynomial with degree less than $|\mathbb{F}^{(\ell_1)}|$. For every $i \in \{0, 1, \ldots, n\}$, there is a unique polynomial $p'\colon (\mathbb{F}^{(\ell_2)})^i \times (\mathbb{F}^{(\ell_1)})^{n-i} \to \mathbb{F}^{(\ell_2)}$ that agrees with p on all points in $(\mathbb{F}^{(\ell_1)})^n$ (here we identify $(\mathbb{F}^{(\ell_1)})^n$ as a subset of $(\mathbb{F}^{(\ell_2)})^n$ via the embedding $\tau_{\ell_1\to\ell_2}$) and has the same degree of p. We call p' the unique extension of p to the domain $(\mathbb{F}^{(\ell_2)})^i \times (\mathbb{F}^{(\ell_1)})^{n-i}$.[37]

Let $\kappa^{(\ell)}$ be the natural bijection between $\{0,1\}^n$ and $\mathbb{F}^{(\ell)} = \mathsf{GF}(2^n)$: for every $a \in \{0,1\}^n$, $\kappa^{(\ell)}(a) = \left(\sum_{i\in[n]} a_i \cdot \mathbf{x}^{i-1}\right)_{\mathbb{F}^{(\ell)}}$. 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, when we say the i-th element in $\mathbb{F}^{(\ell)}$, we mean the element in $\mathbb{F}^{(\ell)}$ encoded by the i-th lexicographically smallest Boolean string in $\{0,1\}^{\mathsf{pw}_\ell}$.

The following lemma will be very useful for us.

LEMMA 9.1. *Let $\ell \in \mathbb{N}$ and $n = \mathsf{pw}_\ell$. There are* $\mathrm{poly}(n)$*-time computable projections $\mathsf{Emd}_\ell\colon \{0,1\}^n \to \{0,1\}^{3n}$ and $\mathsf{Emd}_\ell^{-1}\colon \{0,1\}^{3n} \to \{0,1\}^n$ such that:*

[35] $\mathbf{x}^n + \mathbf{x}^{n/2} + 1 \in \mathbb{F}_2[x]$ is irreducible, see [61, Theorem 1.1.28].

[36] To see this, note that the mapping $\mathbf{x} \mapsto \mathbf{x}^3$ maps $\mathbf{x}^n + \mathbf{x}^{n/2} + 1$ to $\mathbf{x}^{3n} + \mathbf{x}^{3n/2} + 1$.

[37] In more details, p' is obtained by evaluating the polynomial p on the domain $(\mathbb{F}^{(\ell_2)})^i \times (\mathbb{F}^{(\ell_1)})^{n-i}$. Although p has coefficients in $\mathbb{F}^{(\ell_1)}$, we can interpret its coefficients as elements in $\mathbb{F}^{(\ell_2)}$ via the mapping $\tau_{\ell_1\to\ell_2}$ for evaluating p'.

1. $\tau_\ell((b)_{\mathbb{F}^{(\ell)}}) = (\mathit{Emd}_\ell(b))_{\mathbb{F}^{(\ell+1)}}$ *for every* $b \in \{0,1\}^n$.
2. $\mathit{Emd}_\ell^{-1} \circ \mathit{Emd}_\ell$ *is the identity function on* $\{0,1\}^n$.

For $\ell_1, \ell_2 \in \mathbb{N}$ such that $\ell_1 < \ell_2$, we also use $\mathsf{Emd}_{\ell_1 \to \ell_2}$ to denote the composition $\mathsf{Emd}_{\ell_2 - 1} \circ \mathsf{Emd}_{\ell_2 - 2} \circ \cdots \circ \mathsf{Emd}_{\ell_1}$, and $\mathsf{Emd}_{\ell_2 \to \ell_1}$ to denote the composition $\mathsf{Emd}_{\ell_1}^{-1} \circ \cdots \mathsf{Emd}_{\ell_2-2}^{-1} \circ \mathsf{Emd}_{\ell_2-1}^{-1}$. From Lemma 9.1, both $\mathsf{Emd}_{\ell_1 \to \ell_2}$ and $\mathsf{Emd}_{\ell_2 \to \ell_1}$ are poly(pw_{ℓ_2})-time computable projections.

In other words, $\mathsf{Emd}_{\ell_1 \to \ell_2}$ transforms the Boolean encoding of $\beta \in \mathbb{F}^{(\ell_1)}$ into the Boolean encoding of $\tau_{\ell_1 \to \ell_2}(\beta) \in \mathbb{F}^{(\ell_2)}$; $\mathsf{Emd}_{\ell_2 \to \ell_1}$ takes the Boolean encoding of an element $\tau_{\ell_1 \to \ell_2}(\beta) \in \mathbb{F}^{(\ell_2)}$ for $\beta \in \mathbb{F}^{(\ell_1)}$, and outputs the Boolean encoding of β. Also note that $\mathsf{Emd}_\ell = \mathsf{Emd}_{\ell \to \ell+1}$, $\mathsf{Emd}_\ell^{-1} = \mathsf{Emd}_{\ell+1 \to \ell}$, and $\mathsf{Emd}_{\ell_2 \to \ell_1} \circ \mathsf{Emd}_{\ell_1 \to \ell_2}$ is the identity function on $\{0,1\}^{\mathsf{pw}_{\ell_1}}$.

Proof of Lemma 9.1. Given $b \in \{0,1\}^n$, from the definition of τ_ℓ, we have

$$\tau_\ell((b)_{\mathbb{F}^{(\ell)}}) = \sum_{i=1}^{n} b_i \cdot \mathbf{x}^{3(i-1)}.$$

From the above equation, we can simply define $\mathsf{Emd}_\ell(b) \in \{0,1\}^{3n}$ such that for each $j \in [3n]$,

$$(\mathsf{Emd}_\ell(b))_j = \begin{cases} b_{j/3} & 3 \text{ divides } j, \\ 0 & \text{otherwise.} \end{cases}$$

Item (1) of the lemma then follows immeidately. We also define $\mathsf{Emd}_\ell^{-1} \colon \{0,1\}^{3n} \to \{0,1\}^n$ as follows: for every $a \in \{0,1\}^{3n}$ and every $j \in [n]$, $(\mathsf{Emd}_\ell^{-1}(a))_j = a_{3j}$. It is straightforward to verify that $\mathsf{Emd}_\ell^{-1} \circ \mathsf{Emd}_\ell$ is the identity function on $\{0,1\}^n$. This proves Item (2) of the lemma. ∎

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 $2^n \leq |\mathbb{F}_n| \leq 2^{3n}$.

9.1.2. Uniform TC^0 Circuits for Arithmetic Operations over $\mathbb{F}_n$. We will need the uniform TC^0 circuits for arithmetic operations over $\mathbb{F}_n$ in [35, 34].

LEMMA 9.2 ([35, 34]). *Let $n \in \mathbb{N}$. There are uniform TC^0 circuits for the following three 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 $a_1, \ldots, a_t \in \mathbb{F}_n$, compute $\prod_{i \in [t]} a_i$.*
3. Division: *Given $a, b \in \mathbb{F}_n$ such that $b \neq 0$, compute a/b.*[38]

COROLLARY 9.3. *There is an algorithm D^{intp} satisfying the following:*

1. *D^{intp} takes $n \in \mathbb{N}$, $t \in [|\mathbb{F}_n|]$, a list $(a_1, b_1), \ldots, (a_t, b_t) \in \mathbb{F}_n \times \mathbb{F}_n$ with distinct a_i's, and an element $x \in \mathbb{F}_n$ as input, and outputs an element from $\mathbb{F}_n$.*
2. *Let $p(\mathbf{x}) \colon \mathbb{F}_n \to \mathbb{F}_n$ is the unique polynomial with degree at most $t-1$ such that $p(a_i) = b_i$ for every $i \in [t]$. D^{intp} outputs $p(x)$.*
3. *D^{intp} can be implemented by a uniform TC^0 circuit family.*

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

$$e_i(\mathbf{x}) = \prod_{j \in [n] \setminus \{i\}} \frac{\mathbf{x} - a_j}{a_i - a_j}.$$

[38] [34] gave a uniform TC^0 circuit family computing x^t given $x \in \mathbb{F}_n$ and an integer t encoded in binary. This allows us to compute the inverse $x^{-1} = x^{|\mathbb{F}_n|-2}$ given $x \in \mathbb{F}_n$ by a uniform TC^0 circuit family.

We have that $p(\mathbf{x}) = \sum_{i\in[n]} e_i(\mathbf{x})\cdot b_i$. Using Item (2) of Lemma 9.2, $e_i(x)$ can be computed by uniform TC^0 circuits given input $x \in \mathbb{F}_n$ and the list $\{(a_i, b_i)\}_{i\in[t]}$. Then using Item (1) of Lemma 9.2, $p(x)$ can be computed in uniform TC^0 given $x \in \mathbb{F}_n$ and the list $\{(a_i, b_i)\}_{i\in[t]}$. This completes the proof. ∎

9.2. Review of the Construction in [59]. We need the following lemma from [59], which builds on the proof of $\mathsf{IP} = \mathsf{PSPACE}$ theorem [43, 54].

LEMMA 9.4 (Adapted from [59, Lemma 4.1]). *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. *(Self-reducibility) There is an algorithm* Red *satisfying the following:*
   (a) Red *takes $n, i \in \mathbb{N}_{\geq 1}$ such that $i < n$, and $\vec{x} \in \mathbb{F}_n^n$ as input, and a function $h \colon \mathbb{F}_n^n \to \mathbb{F}_n$ as oracle.*
   (b) $\mathsf{Red}_{n,i}^{f_{n,i+1}}$ *computes $f_{n,i}$.*
   (c) Red *can be implemented by a uniform non-adaptive TC^0 oracle circuit family.*
2. *(Base-case) There is an algorithm* Base *satisfying the following:*
   (a) Base *takes $n \in \mathbb{N}_{\geq 1}$ and $\vec{x} \in \mathbb{F}_n^n$ as input, and outputs $f_{n,n}(\vec{x})$.*
   (b) Base *can be implemented by a uniform TC^0 circuit family.*
3. *(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 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 $\vec{z} = A_L^{\mathsf{red}}(x)$.*[39]
4. *(Low degree) For every $n \in \mathbb{N}_{\geq 1}$ and $i \in [n]$, $f_{n,i}$ has degree at most n.*
5. *(Instance checker) There is a randomized algorithm* IC *such that,* IC *takes $n, i \in \mathbb{N}_{\geq 1}$ such that $i \leq n$, 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 $2/3$, over the internal randomness of* IC.

For completeness, we will prove Lemma 9.4 together with some additional properties of $\mathscr{F}^{\mathsf{TV}}$ in Section 9.2.2.

9.2.1. Technical Challenges in Adapting [59] for Our Purpose. The original language in [59] just computes $f_{n,i}$ in the order of first increasing in n and then decreasing in i. By Lemma 9.4, this direct construction gives a PSPACE-complete language that is both downward self-reducible and error correctable (as polynomials are error correctable). To make it further paddable, [23, 51] used a padding construction such that on inputs of an appropriate length $e_{n,i}$, the new language L encodes $f_{n,i}$ and all polynomials that come before it as subfunctions. However, as we will see, such a direct construction does not have error correctability.

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

Error correctability and paddability. We first describe the technical challenges we need to overcome for constructing a PSPACE-complete language that is both error correctable and paddable.

The construction of [23, 51] gives us a PSPACE-complete language that encodes not a single polynomial, but many different polynomials on a single input length. This ruins the error correctability so we need to do some interpolation combine these many polynomials into a single polynomial again. One obvious problem is that these polynomials are over different fields and may have different numbers of variables, we resolve that by a careful choice of the fields (for all $n < m$, $\mathbb{F}_n$ is a *subfield* of $\mathbb{F}_m$) and adding dummy variables.

An immediate idea is to use the following direct interpolation: for some field $\mathbb{F}$, suppose we have k polynomials $g_1, g_2, \ldots, g_k \colon \mathbb{F}^n \to \mathbb{F}$ of degree n; we then construct a single polynomial $G_k \colon \mathbb{F}^{n+1} \to \mathbb{F}$ with degree $n+k$, such that $G_k(w_i, x) = g_i(x)$ via a simple interpolation, where w_i is the i-th element in $\mathbb{F}$. The issue here is that then G_{k-1} cannot be reduced to G_k easily (so it is not paddable). Therefore, we make a different choice of interpolation that allows us to preserve the paddability. Specifically, we define $G_k : \mathbb{F}^n \times \mathbb{F}^k \to \mathbb{F}$ as

$$G_k(x, y_1, y_2, \ldots, y_k) \coloneqq \sum_{i=1}^{k} g_i(x) \cdot y_i. \tag{9.1}$$

In more details, we will set $g_1, \ldots, g_k$ be (padded and extended versions of) the first k polynomials in the sequence

$$f_{1,1}, f_{2,2}, \ldots, f_{2,1}, f_{3,3}, \ldots, f_{3,1}, \ldots, f_{n,n}, \ldots, f_{n,1}, \ldots,$$

and define G_k as in (9.1). See (9.9) for a formal definition.

Finally, the polynomials are over a large alphabet $\mathbb{F}_n$, and we have to convert them into Boolean functions. This step is a standard application of Walsh-Hadamard codes.

The next step is to verify all the reducibility properties from Theorem 3.7 holds for our new PSPACE-complete language, and the corresponding reductions have implementations by low-depth oracle circuits.

For the paddability it is straightforward from our definition. For the weakly error correctability, it is still relatively straightforward from the local decoders of Reed-Muller codes and Walsh-Hadamard codes. The main difficulty here is to verify same-length checkability and downward self-reducibility, and construct low-depth reductions for them.

Same-length checkability. Here we need to argue the instance-checker in [59, 23, 51] can be implemented in TC^0. This looks counter-intuitive at first—the instance checker in [59, 23, 51] simulates the interactive proof protocol for PSPACE [43, 54]. Since it is an interactive proof protocol, it appears that this instance-checker should proceed one step after another step (*i.e.*, it is highly sequential), and it should not have a low-depth implementation such as a non-adaptive TC^0 oracle circuit family.

Recall the reason why the $\mathsf{IP} = \mathsf{PSPACE}$ protocol has to be adaptive: the verifier does not want the prover's answer to her current question depends on her future questions.[40] The crucial observation here is that: in the instance-checker setting, the

[40] If the prover in a standard sum-check protocol can know *in advance* all verifier's questions, then it can easily cheat.

prover's strategy *is already committed to the given oracle*, so the issue above would not arise. This enables us to check different stages of the interactive proof protocol in the same time, and from which we can construct a TC^0 oracle circuit for the instance checker. See Algorithm 9.1 and Lemma 9.8 for more details.

Downward self-reducibility. Establishing the downward self-reducibility is not obvious. Here we wish to compute G_{k+1} given oracle access to G_k, for every $k \in \mathbb{N}$.[41]

When G_k and G_{k+1} are over the same field, downward self-reducibility follows from the way that the $f_{n,i}$'s are constructed. But when G_k and G_{k+1} are over different fields $\mathbb{F}_{\mathsf{old}}$ and $\mathbb{F}_{\mathsf{new}}$ ($\mathbb{F}_{\mathsf{old}}$ is a subfield of $\mathbb{F}_{\mathsf{new}}$), it is not clear how to evaluate G_{k+1} given an oracle access to G_k. To circumvent this issue, suppose $G_k\colon \mathbb{F}_{\mathsf{old}}^{n+k} \to \mathbb{F}_{\mathsf{old}}$ is a degree $d \le \mathrm{poly}(n)$ polynomial, we wish to extent it to a polynomial $H_k\colon \mathbb{F}_{\mathsf{new}}^{n+k} \to \mathbb{F}_{\mathsf{new}}$.

For this purpose, we construct $n+k+1$ intermediate polynomials $H_0^{\mathsf{int}}, \dots, H_{n+k}^{\mathsf{int}}$, such that $H_i^{\mathsf{int}}\colon \mathbb{F}_{\mathsf{new}}^{i} \times \mathbb{F}_{\mathsf{old}}^{n+k-i} \to \mathbb{F}_{\mathsf{new}}$ is constructed by extending G_k to the domain $\mathbb{F}_{\mathsf{new}}^{i} \times \mathbb{F}_{\mathsf{old}}^{n+k-i}$. Note that $H_{n+k}^{\mathsf{int}} = H_k$. We simply insert the polynomials $H_0^{\mathsf{int}}, H_2^{\mathsf{int}}, \dots, H_{n+k}^{\mathsf{int}}$ between G_k and G_{k+1}. Note that for each $i \in [n+k]$, given oracle access to H_{i-1}^{int}, it is easy to evaluate H_i^{int} by interpolation. Also, G_{k+1} can be evaluated easily given oracle access to H_k, as now they are over the same field $\mathbb{F}_{\mathsf{new}}$, and H_0^{int} can be easily evaluated given oracle access to G_k. To summarize, inserting $H_0^{\mathsf{int}}, \dots, H_{n+k}^{\mathsf{int}}$ between G_k and G_{k+1} restore the downward self-reducibility.

It remains to ensure that adding these H_i^{int}'s does not hurt other properties we want. It is relatively straightforward (but a bit tedious) to verify that paddability, weakly error correctability still holds. To prove that the H_i^{int}'s are also same-length checkable, we use an *extension checker*. See Lemma 9.9 and the proof of Lemma 9.15 for more details.

9.2.2. Additional properties of $\mathscr{F}^{\mathsf{TV}}$ and a proof of Lemma 9.4. For a vector $\vec{x} \in \mathbb{F}_n^n$ and $i \in [n]$, we use $\vec{x}^{i\leftarrow z}$ to denote the vector obtained from $\vec{x}$ by changing x_i to z. We first state the following lemma, which gives details on how the self-reduction Red in Lemma 9.4 is implemented.

Lemma 9.5 (Self-reduction for $\mathscr{F}^{\mathsf{TV}}$). *Let $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i}\colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n\in\mathbb{N}_{\ge 1}, i\in[n]}$ be as in Lemma 9.4. For every $n, i \in \mathbb{N}_{\ge 1}$ such that $i < n$, one can compute an index $J = J_{n,i} \in [n]$ and a type $Q = Q_{n,i} \in \{\exists, \forall, \mathsf{LIN}\}$ 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*
$$f_{n,i}(\vec{x}) = x_j \cdot f_{n,i+1}(\vec{x}^{J\leftarrow 1}) + (1 - x_j) \cdot f_{n,i+1}(\vec{x}^{J\leftarrow 0}).$$

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. $S_{\mathsf{LIN}}(x, y_0, y_1) = xy_1 + (1 - x)y_0$.

[41] This is different than the self-reducibility in Lemma 9.4, where we only have self-reducibility within the sequence $f_{n,1}, f_{n,2}, \dots, f_{n,n}$ for every fixed $n \in \mathbb{N}$.

Now the three cases in Lemma 9.5 can be succinctly written as

$$f_{n,i}(\vec{x}) = S_Q(x_j, f_{n,i+1}(\vec{x}^{J\leftarrow 0}), f_{n,i+1}(\vec{x}^{J\leftarrow 1})). \tag{9.2}$$

We also give a detailed implementation of the instance checker IC in Lemma 9.4 in Algorithm 9.1.

Algorithm 9.1: The instance checker IC from Lemma 9.4

```
Given n, i ∈ ℕ_{≥1} such that i ≤ n, and x⃗ ∈ 𝔽_n as the input;
Given n − i + 1 functions f̃_i, f̃_{i+1}, ..., f̃_n : 𝔽_n^n → 𝔽_n as the oracles;
Let α⃗_i = x⃗;
for j ∈ {i, i+1, ..., n−1} do
    Compute J = J_{n,j} and Q = Q_{n,j} from Lemma 9.5;
    Let w_1, ..., w_{n+1} be the first n + 1 elements in 𝔽_n;
    Set b_ℓ = f̃_{j+1}((α⃗_j)^{J←w_ℓ}) for every ℓ ∈ [n + 1];
    Let 𝓛 = {(w_ℓ, b_ℓ)}_{ℓ∈[n+1]};
    if f̃_j(α⃗_j) ≠ S_Q((α⃗_j)_J, D^intp_{n,n+1}(𝓛, 0), D^intp_{n,n+1}(𝓛, 1)) then
        return ⊥;
    Draw z_j ∈_R 𝔽_n;
    Set α⃗_{j+1} = (α⃗_j)^{J←z_j};
if f̃_n(α⃗_n) = Base_n(α⃗_n) then
    return f̃_i(x⃗);
else
    return ⊥;
```

In the following, we include a proof of Lemma 9.4 to verify the extra properties that is not stated in [59].

We first introduce the following variants of the TQBF (True Quantified Boolean Formula) problem, which is also used in [59].

Definition 9.6. *The TQBFU problem*[42] *takes two matrices $\vec{y}, \vec{z} \in \{0,1\}^{n\times n}$ as input, and the goal is to decide whether the following quantified Boolean formula holds*

$$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \bigwedge_{j\in[n]} \bigvee_{k\in[n]} (y_{j,k} \wedge x_k) \vee (z_{j,k} \wedge \neg x_k), \tag{9.3}$$

where Q_i equals $\exists$ for odd i, and $\forall$ for even i. We use TQBFU_n to denote the TQBFU problem with parameter n (and input length $2n^2$).

We first show that TQBFU is still PSPACE-complete.

Lemma 9.7. *TQBFU is PSPACE-complete.*

Proof. Recall that the TQBF problem is defined as follows: given an n-variable m-clause CNF $\phi(\vec{x})$ as input, the goal is to decide whether $Q_1x_1Q_2\cdots Q_nx_n\phi(\vec{x})$ holds, where Q_i equals $\exists$ for odd i, and $\forall$ for even i. By adding dummy variables or dummy clauses, we can assume that $n = m$.

[42] U stands for universal, since here we have a universal formula in (9.3) that can simulate every n-clause n-variable CNF.

For every $j \in [n]$, letting $C_j(\vec{x})$ be the j-th clause in $\phi(\vec{x})$, we set $y_{j,k}$ to be 1 if C_j contains the variable x_k, and 0 otherwise. Similarly, we set $z_{j,k}$ to be 1 if C_j contains the negated variable $\neg x_k$, and 0 otherwise. Now we can verify that $\mathsf{TQBFU}(\vec{y},\vec{z}) = \mathsf{TQBF}(\phi)$ from (9.3). This proves the PSPACE-completeness of TQBFU as TQBF is PSPACE-complete [56] (see also [7, Theorem 4.13]). ∎

Now we are ready to prove Lemma 9.4 and Lemma 9.5. Our proof follows closely the proof sketch of [59, Lemma 4.1].[43]

Proof of Lemma 9.4 and Lemma 9.5. Let $n \in \mathbb{N}$, and let m be the largest integer such that $6m^3 \le n$. We will use $\{f_{n,i}\}_{i\in[n]}$ to encode the problem TQBFU_m. When $n < 6$, we simply set $f_{n,i}$ to be the zero n-variate polynomial for all $i \in [n]$. So we can assume $m \ge 1$.

We first arithmetize the formula in (9.3) to get the following polynomial $P\colon \mathbb{F}_n^m \times \mathbb{F}_n^{m^2} \times \mathbb{F}_n^{m^2} \to \mathbb{F}_n$

$$P(\vec{x},\vec{y},\vec{z}) = \prod_{j\in[m]} \left[1 - \prod_{k\in[m]} (1 - p(x_k, y_{j,k}, z_{j,k}))\right], \tag{9.4}$$

where $p\colon \mathbb{F}_n^3 \to \mathbb{F}$ is defined as $p(x,y,z) = xy + (1-x)z$. One can verify that $p(x,y,z)$ agrees with $(y\wedge x)\vee(z\wedge\neg x)$ over all Boolean inputs $x,y,z \in \{0,1\}$, and also $P(\vec{x},\vec{y},\vec{z})$ agrees with

$$\bigwedge_{j\in[n]} \bigvee_{k\in[n]} (y_{j,k}\wedge x_k) \vee (z_{j,k} \wedge \neg x_k)$$

on every $\vec{x} \in \{0,1\}^n$ and every $\vec{y},\vec{z} \in \{0,1\}^{n\times n}$. Since p has degree 2, P has degree $2m^2$.

Now we make a list $\mathcal{L}$ consisting of pairs from $\mathbb{N} \times \{\exists, \forall, \mathsf{LIN}\}$:

1. For every integer i from m down to 1:
   (a) We append (i, Q_i) to the end of the list $\mathcal{L}$.
   (b) For every integer j from 1 to $2m^2 + m$, we append (j, LIN) to the end of the list L.
2. We append $n - m\cdot(2m^2+m+1) - 1$ copies of $(1, \mathsf{LIN})$ to the end of the list $\mathcal{L}$. (Note that $m\cdot(2m^2+m+1)+1 \le 6m^3 \le n$ for $m \ge 1$.)

From the construction above, it is easy to see that $|\mathcal{L}| = n-1$. Now we are ready to define our polynomials $\{f_{n,i}\}_{i\in[n]}$.

1. We set $f_{n,n}(\vec{x}) = P(\vec{x}_{\le m+2m^2})$, where $\vec{x}_{\le m+2m^2}$ is the $(m+2m^2)$-length prefix of $\vec{x}$.
2. For every i from 1 to $n-1$, let (J_i, Q_i) be the i-th element of the list $\mathcal{L}$. We set

$$f_{n,n-i}(\vec{x}) = S_{Q_i}(x_{J_i}, f_{n,n-i+1}(\vec{x}^{J_i\leftarrow 0}), f_{n,n-i+1}(\vec{x}^{J_i\leftarrow 1})) \tag{9.5}$$

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

Now we verify each item of Lemma 9.4 separately.

First, Lemma 9.5 and Item (1) of Lemma 9.4 follows immediately from our definition of $f_{n,n-i}$ in (9.5) and Lemma 9.2. The base case (Item (2) of Lemma 9.4) also follows directly from $f_{n,n}(\vec{x}) = P(\vec{x}_{\le m+2m^2})$, the definition of P in (9.4), and Lemma 9.2.

[43] We also refer readers to [7, Section 8.3] and [26, Section 9.1.3] for expositions on the celebrated proof of IP = PSPACE [44, 54].

Item (3) and Item (5) of Lemma 9.4 can be established identically as in [59] (these essentially follows from the same argument as in the proof of $\mathsf{IP} = \mathsf{PSPACE}$. We added some dummy $(1, \mathsf{LIN})$ so that we have exactly n polynomials, but these do not affect the argument.) The instance checker IC in Item (5) is described in Algorithm 9.1.

To see Item (4) of Lemma 9.4, note that $f_{n,n}$ has the same degree of P, which is $2m^2$. For every $i \in [n-1]$ such that $Q_i \in \{\exists, \forall\}$, since $S_\exists$ and $S_\forall$ has degree 2, we have $\deg(f_{n,n-i}) \le 2\deg(f_{n,n-i+1})$. Also, (j_i, Q_i) in L then is followed by a sequence $(1, \mathsf{LIN}), (2, \mathsf{LIN}), \cdots, (2m^2+m, \mathsf{LIN})$, which reduces the degree back to at most $2m^2+m$. Hence, we can see the degree is at most $4m^2+2m \le 6m^3 \le n$ for every $f_{n,i}$. ∎

Finally, as discussed in Section 9.2.1, we next show the instance checker IC in Item (5) of Lemma 9.4 can indeed be implemented by a randomized uniform non-adaptive TC^0 circuit family.

Lemma 9.8. *The instance checker* IC *from Lemma 9.4 can be implemented a randomized uniform non-adaptive* TC^0 *circuit family.*

Proof. The crucial observation here is that we can first draw $z_i, \dots, z_{n-1} \in_{\mathsf{R}} \mathbb{F}_n$ beforehand and run each iteration of the for loop in Algorithm 9.1 in parallel (and return $\perp$ if any of the check on Line 9 fails). Note that for each $j \in \{i, \dots, n\}$ and $\ell \in [n]$, we have

$$(\vec{\alpha}_j)_\ell = \begin{cases} x_\ell & \text{there is no } j' < j \text{ s.t. } J_{n,j'} = \ell \\ z_{j_{\mathsf{max}}} & \text{otherwise,} \end{cases} \tag{9.6}$$

where j_{max} is the maximum $j' < j$ s.t. $J_{n,j'} = \ell$.

Using (9.6), for every $j \in \{i, \dots, n\}$, we can compute $\vec{\alpha}_j$ in uniform TC^0 given $\vec{x}, z_i, \dots, z_{n-1}$. It then follows from Algorithm 9.1, Corollary 9.3, and Item (2) of Lemma 9.2 that IC can be implemented by a randomized uniform non-adaptive TC^0 circuit family. ∎

9.3. Construction of The PSPACE-complete Language.

In this section, we prove Theorem 3.7. We will first construct a PSPACE-complete language L^{PSPACE}, and then prove it satisfies all the desired properties stated in Theorem 3.7.

9.3.1. Extension checker.

We will need the following *extension checker* that checks whether a polynomial $f \colon \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i} \to \mathbb{F}_{\mathsf{new}}$ is the correct extension of another polynomial $g \colon \mathbb{F}_{\mathsf{old}}^m \to \mathbb{F}_{\mathsf{old}}$. Our construction is a simple adaption of the sum-check protocol.

Lemma 9.9. *There is an algorithm* $\mathsf{Ext\text{-}C}$ *such that:*

1. $\mathsf{Ext\text{-}C}$ *takes two integers* $n_1, n_2 \in \mathbb{N}$ *such that* $n_1 < n_2$ *as two parameters. We set* $\mathbb{F}_{\mathsf{old}} = \mathbb{F}_{n_1}$ *and* $\mathbb{F}_{\mathsf{new}} = \mathbb{F}_{n_2}$.
2. $\mathsf{Ext\text{-}C}$ *takes* $m, i, d \in \mathbb{N}$ *such that* $i \le m$ *and* $d \le |\mathbb{F}_{\mathsf{old}}| - 1$ *and* $\vec{z} \in \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$ *as input, and two functions* $f \colon \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i} \to \mathbb{F}_{\mathsf{new}}$ *and* $g \colon \mathbb{F}_{\mathsf{old}}^m \to \mathbb{F}_{\mathsf{old}}$ *as oracles.*
3. *Suppose* g *is a polynomial with degree at most* d *and let* g' *be the unique extension of* g *to the domain* $\mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$. *The following two statements hold:*
   (a) *If* $f = g'$, *then* $\mathsf{Ext\text{-}C}_{n_1,n_2,m,i,d}(\vec{z})^{f,g}$ *outputs* $g'(\vec{z})$ *with probability* 1 *for every* $\vec{z} \in \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$.
   (b) *For every oracle* f *and every* $\vec{z} \in \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$, $\mathsf{Ext\text{-}C}_{n_1,n_2,m,i,d}(\vec{z})^{f,g}$ *outputs an element from* $\{g'(\vec{z}), \perp\}$ *with probability at least* $1 - \frac{m \cdot d}{|\mathbb{F}_{\mathsf{old}}|}$.

4. Ext-C *can be implemented by a randomized uniform non-adaptive* TC^0 *circuit family that queries* g *at most once (but can query* f *many times).*

Algorithm 9.2: The extension checker $\mathsf{Ext\text{-}C}_{n_1,n_2,m,i,d}$

---

1 Given $\vec{x} \in \mathbb{F}_{\mathsf{new}}^i$ and $\vec{y} \in \mathbb{F}_{\mathsf{old}}^{m-i}$ as input, and $f\colon \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i} \to \mathbb{F}_{\mathsf{new}}$ and $g\colon \mathbb{F}_{\mathsf{old}}^m \to \mathbb{F}_{\mathsf{old}}$ as oracles;
2 Draw a random vector $\vec{\beta} \in \mathbb{F}_{\mathsf{old}}^i$;
3 Let $w_1, \dots, w_{d+1}$ be the first $d+1$ elements in $\mathbb{F}_{\mathsf{old}}$;
4 **for** $\mu \in \{1, 2 \dots, i\}$ **do**
5   Let $\vec{\alpha} = \vec{\beta}_{<\mu} \circ \vec{x}_{\geq \mu} \circ \vec{y}$;
6   For every $\eta \in [d+1]$, set $b_\eta = f(\vec{\alpha}^{\mu \leftarrow w_\eta})$;
7   Let $\mathcal{L} = \{(w_\eta, b_\eta)\}_{\eta \in [d+1]}$;
8   **if** $D^{\mathsf{intp}}_{n_2,d+1}(\mathcal{L}, \beta_\mu) \neq f(\vec{\alpha}^{\mu \leftarrow \beta_\mu})$ *or* $D^{\mathsf{intp}}_{n_2,d+1}(\mathcal{L}, \alpha_\mu) \neq f(\vec{\alpha})$ **then**
9   **return** $\perp$;
10 **if** $g(\vec{\beta} \circ \vec{y}) = f(\vec{\beta} \circ \vec{y})$ **then**
11   **return** $f(\vec{x} \circ \vec{y})$;
12 **else**
13   **return** $\perp$;

Proof of Lemma 9.9. The algorithm of $\mathsf{Ext\text{-}C}_{n_1,n_2,m,i,d}$ (we will denote it by Ext-C below for simplicity) is described in Algorithm 9.2.

To see Item (4) of the lemma, all the iterations of the for loop in Algorithm 9.2 can be implemented in parallel. Since $D^{\mathsf{intp}}_{n_2,d+1}$ can be implemented by uniform TC^0, it follows that Ext-C can be implemented by non-adaptive uniform TC^0. It is also clear that Ext-C queries g at most once in Algorithm 9.2 (it only queries g at Line 10).

Now we show that if $f = g'$, then Ext-C outputs $g'(\vec{z})$ (Here $\vec{z} = \vec{x} \circ \vec{y}$) with probability 1 (*i.e.*, Item (3.a) of the lemma). Note that since $f = g'$ is the unique extension of g to the domain $\mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$ and $\deg(f) = \deg(g) \leq d$. It follows from the definition of the b_η's that

$$D^{\mathsf{intp}}_{n_2,d+1}(\{(w_\eta, b_\eta)\}_{\eta \in [d+1]}, \xi) = f(\vec{\alpha}^{\mu \leftarrow \xi})$$

for every $\xi \in \mathbb{F}_{\mathsf{new}}$. Hence the check at Line 8 passes for every $\mu \in [i]$. Moreover, since f is the unique extension of g, the check at Line 10 passes as well. To summarize, it follows that Ext-C outputs $f(\vec{z}) = g'(\vec{z})$ with probability 1.

Next we prove Item (3.b) of the lemma. We first note that if $f(\vec{z}) = g'(\vec{z})$, then since Algorithm 9.2 either outputs $f(\vec{z})$ or $\perp$, Item (3.b) holds with probability 1. So in the following we assume that $f(\vec{z}) \neq g'(\vec{z})$. For every $\mu \in \{0, 1, \dots, i\}$, we let $\vec{\alpha}_\mu = \vec{\beta}_{\leq \mu} \circ \vec{x}_{> \mu} \circ \vec{y}$ (hence, $\vec{\alpha}$ at Line 5 during the μ-th iteration equals $\vec{\alpha}_{\mu-1}$) and let $\mathcal{E}_\mu$ be the event that either $f(\vec{\alpha}_\mu) \neq g'(\vec{\alpha}_\mu)$ or Ext-C returns $\perp$ during the first μ iterations of the for loop in Algorithm 9.2.

We first note that by definition, $\mathcal{E}_\mu$ only depends on $\vec{z}_{\leq \mu}$. Also, from our assumption, we have $\Pr[\mathcal{E}_0] = 1$. We need the following claim.

CLAIM 6. *For every* $\mu \in [i]$, $\Pr[\mathcal{E}_\mu | \mathcal{E}_{\mu-1}] \geq 1 - \frac{d}{|\mathbb{F}_{\mathsf{old}}|}$.

Proof. It suffices to show that conditioning on that Ext-C reaches the μ-th iteration of the for loop and $f(\vec{\alpha}_{\mu-1}) \neq g'(\vec{\alpha}_{\mu-1})$, $\mathcal{E}_\mu$ holds with probability at least $1 - \frac{d}{|\mathbb{F}_{\mathsf{old}}|}$.

Now, let $P\colon \mathbb{F}_{\mathsf{new}} \to \mathbb{F}_{\mathsf{new}}$ be the unique polynomial such that $P(w_\eta) = (b_\eta)$ for every $\eta \in [d{+}1]$ and $\deg(P) \le d$, and $Q\colon \mathbb{F}_{\mathsf{new}} \to \mathbb{F}_{\mathsf{new}}$ be the restriction of g' defined by $Q(v) = g'(\vec{\alpha}_{\mu-1}^{\mu\leftarrow v}) = g'(\vec{\beta}_{\le \mu-1} \circ v \circ \vec{x}_{>\mu} \circ \vec{y})$. Note that $\deg(Q) \le \deg(g') = \deg(g) \le d$. There are two cases:

1. $P \neq Q$. In this case, since β_μ distributed uniformly random from $\mathbb{F}_{\mathsf{old}}$ and is independent from P and Q, we have $P(\beta_\mu) \neq Q(\beta_\mu)$ with probability at least $1 - d/|\mathbb{F}_{\mathsf{old}}|$, since there at most d roots of $P - Q$.
   Next we show that $P(\beta_\mu) \neq Q(\beta_\mu)$ implies that $\mathcal{E}_\mu$ holds. There are two subcases:
   (a) $P(\beta_\mu) \neq f(\vec{\alpha}_\mu)$. In this case, we have
   $$P(\beta_\mu) = D^{\mathsf{intp}}_{n_2,d+1}(\mathcal{L}, \beta_\mu) \neq f(\vec{\alpha}^{\mu\leftarrow\beta_\mu}) = f(\vec{\alpha}_\mu).$$
   Hence Ext-C returns $\perp$ at Line 9, and $\mathcal{E}_\mu$ holds.
   (b) $P(\beta_\mu) \neq Q(\beta_\mu)$. In this case, note that $f(\vec{\alpha}_\mu) = P(\beta_\mu)$ and $g'(\vec{\alpha}_\mu) = Q(\beta_\mu)$, we have
   $$f(\vec{\alpha}_\mu \neq g'(\vec{\alpha}_\mu),$$
   and $\mathcal{E}_\mu$ holds.
   Putting the above two subcases together, we have that $\mathcal{E}_\mu$ holds with probability $1 - d/|\mathbb{F}_{\mathsf{old}}|$ in this case.
2. $P = Q$. In this case, we have
   $$P((\vec{\alpha}_{\mu-1})_\mu) = Q(\vec{\alpha}_{\mu-1})_\mu) = g'(\vec{\alpha}_{\mu-1}) \neq f(\vec{\alpha}_{\mu-1}),$$
   where the last inequality follows from our assumption. So Ext-C returns $\perp$ at Line 9 and $\mathcal{E}_\mu$ holds with probability 1. ◻

Finally, we show that Claim 6 implies Item (3.b) of the lemma. From 6, we have that $\Pr[\mathcal{E}_i] \ge 1 - \frac{i\cdot d}{|\mathbb{F}_{\mathsf{old}}|} \ge 1 - \frac{m\cdot d}{|\mathbb{F}_{\mathsf{old}}|}$. Item (3.b) then follows from the fact that Ext-C always returns $\perp$ under $\mathcal{E}_i$, since either (1) Ext-C returns $\perp$ during the for loop or (2) $f(\vec{\beta} \circ \vec{y}) = f(\vec{\alpha}_i) \neq g'(\vec{\alpha}_i) = g(\vec{\beta} \circ \vec{y})$ and Ext-C returns $\perp$ at Line 13. ◻

9.3.2. The Language L^{PSPACE}. To construct our PSPACE-complete language L^{PSPACE}, we carefully modify the PSPACE-complete language in [59, Theorem 4.3], and combine that with an application of Walsh-Hadamard codes to turn the polynomials into Boolean functions.

Let $\mathscr{F}^{\mathsf{TV}} = \{f_{n,i}\colon \mathbb{F}_n^n \to \mathbb{F}_n\}_{n\in\mathbb{N}_{\ge 1}, i\in[n]}$ be as in Lemma 9.4. First, we list all polynomials in $\mathscr{F}^{\mathsf{TV}}$ in the following order

$$f_{1,1}, f_{2,2}, \ldots, f_{2,1}, f_{3,3}, \ldots, f_{3,1}, \ldots, f_{n,n}, \ldots, f_{n,1}, \ldots . \tag{9.7}$$

For every $k \in \mathbb{N}$, we let g_k be the k-th polynomial in (9.7). We also set n_k and i_k so that $g_k = f_{n_k,i_k}$, and define $\mathscr{G}^{\mathsf{TV}} = \{g_i\}_{i\in[n]}$.

For every $k \in \mathbb{N}$ and $j \in [k]$, we define $h_{k,j}\colon \mathbb{F}_n^n \to \mathbb{F}_n$ as the following polynomial:

- Let $h'_{k,j}\colon \mathbb{F}_n^{n_j} \to \mathbb{F}_n$ be the unique extension of the polynomial $g_j\colon \mathbb{F}_{n_j}^{n_j} \to \mathbb{F}_{n_j}$.
- We set $h_{k,j}(\vec{x}) = h'_{k,j}(\vec{x}_{\le n_j})$. (*i.e.*, $h_{k,j}$ evaluates $h'_{k,j}$ on its first n_j inputs and ignores the rest.)

The following lemma shows that, using the extension checker from Lemma 9.9, the non-adaptive instance checker for $\mathscr{F}^{\mathsf{TV}}$ (Lemma 9.8) can be converted into a non-adaptive instance-checker for the $h_{k,j}$'s.

LEMMA 9.10. *There is a randomized algorithm* h-IC *such that,* h-IC *takes* $k, j \in \mathbb{N}_{\geq 1}$ *such that* $j \leq k$, $\varepsilon \in (0, 1/2)$, *and* $\vec{x} \in \mathbb{F}_n^n$ *(here* $n = n_k$*) as input, and* j *functions* $\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_j : \mathbb{F}_n^n \to \mathbb{F}_n$ *as oracles, and outputs an element in* $\mathbb{F}_n \cup \{\perp\}$. *The following properties hold for* h-IC*:*

1. *If* $\tilde{h}_\ell = h_{k,\ell}$ *for every* $\ell \in [j]$, *then* $\mathsf{h\text{-}IC}_{k,j,\varepsilon}^{\tilde{h}_1,\ldots,\tilde{h}_j}(\vec{x})$ *outputs* $h_{k,j}(\vec{x})$ *with probability* 1 *for every* $\vec{x} \in \mathbb{F}_n^n$.
2. *For every* $\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_j : \mathbb{F}_n^n \to \mathbb{F}_n$ *and every* $\vec{x} \in \mathbb{F}_n^n$, $\mathsf{h\text{-}IC}_{k,j,\varepsilon}^{\tilde{h}_1,\ldots,\tilde{h}_j}(\vec{x}) \in \{h_{k,j}(\vec{x}), \perp\}$ *with probability* $1 - \varepsilon$, *over the internal randomness of* h-IC*.*
3. h-IC *can be implemented by a* $\mathrm{poly}(k \cdot \log \varepsilon^{-1})$*-size randomized uniform non-adaptive* TC^0 *oracle circuit family.*

Proof. Recall that $g_j = f_{n_j,i_j}$ is a polynomial from $\mathbb{F}_{n_j}^{n_j}$ to $\mathbb{F}_{n_j}$. In the following we use i to denote i_j and m to denote n_j for simplicity. We will assume $m \geq 10$ since otherwise we can simply compute $h_{k,j}(\vec{x})$ by interpolating $f_{m,i}(\vec{x})$ directly without using any oracles.

We first define the following oracles $\tilde{f}_i, \tilde{f}_{i+1}, \ldots, \tilde{f}_m : \mathbb{F}_m^m \to \mathbb{F}_m$ to the instance-checker $\mathsf{IC}_{m,i}$ from Lemma 9.4: for every $\ell \in \{i, i+1, \ldots, m\}$, letting k' be such that $g_{k'} = f_{m,\ell}$ (note that $k' \leq j$ from (9.7)), we set

$$\tilde{f}_\ell(\vec{x}) = \tilde{h}_{k'}(\vec{x}, 0, \ldots, 0). \tag{9.8}$$

As a Boolean function, (9.8) implicitly uses $\mathsf{Emd}_{\ell_m \to \ell_n}$ to convert $\vec{x}$ into a vector in $\mathbb{F}_n^m$, and $\mathsf{Emd}_{\ell_n \to \ell_m}$ to interpret $\tilde{h}_{k'}(\vec{x}, 0, \ldots, 0) \in \mathbb{F}_n$ as an element of $\mathbb{F}_m$. Since $\mathsf{Emd}_{\ell_m \to \ell_n}$ and $\mathsf{Emd}_{\ell_n \to \ell_m}$ are both $\mathrm{poly}(n)$-time computable projections, simulating $\tilde{f}_\ell$ via (9.8) does not affect the circuit complexity of $\mathsf{IC}_{m,i}$.

Applying Lemma 9.4, there is a randomized uniform TC^0 oracle circuit D_1 such that:

1. If $\tilde{h}_\ell = h_{k,\ell}$ for every $\ell \in [j]$, then $D_1^{\tilde{h}_1,\ldots,\tilde{h}_j}(\vec{x})$ outputs $f_{m,m}(\vec{x})$ with probability 1 for every $\vec{x} \in \mathbb{F}_m^m$.[44]
2. For every $\tilde{h}_1, \ldots, \tilde{h}_j$, $D_1^{\tilde{h}_1,\ldots,\tilde{h}_j}(\vec{x})$ outputs an element from $\{f_{m,m}(\vec{x}), \perp\}$ with probability at least $9/10$.[45]

We next run $\mathsf{Ext\text{-}C}_{m,n,m,m,m}$ from Lemma 9.9 with oracle access to $r : \mathbb{F}_n^m \to \mathbb{F}_n$ defined by $r(\vec{x}) = \tilde{h}_j(\vec{x}, 0, 0, \ldots, 0)$ and $D_1^{\tilde{h}_1,\ldots,\tilde{h}_j}$, we also modify it slightly so that whenever $D_1^{\tilde{h}_1,\ldots,\tilde{h}_j}$ returns $\perp$, Ext-C returns $\perp$ as well.

Note that $\mathsf{Ext\text{-}C}_{m,n,m,m,m}$ only queries $D_1^{\tilde{h}_1,\ldots,\tilde{h}_j}$ at most once. By a union bound and Lemma 9.9, it holds that Ext-C outputs $h_{k,j}(\vec{x})$ with probability 1 if $\tilde{h}_\ell = h_{k,\ell}$ for every $\ell \in [j]$, and for every possible oracles $\tilde{h}_1, \ldots, \tilde{h}_j$, Ext-C outputs an element from $\{h_{k,j}(\vec{x}), \perp\}$ with probability at least $9/10 - \frac{m^2}{|\mathbb{F}_m|} \geq 2/3$. The last inequality holds since $m \geq 10$ and $|\mathbb{F}_m| \geq 2^m$.

Finally, we can repeat the algorithm above $O(\log \varepsilon^{-1})$ times to amplify the $2/3$ success probability to $1 - \varepsilon$. Our final instance-checker has a randomized uniform non-adaptive TC^0 circuit family since Ext-C does. This completes the proof. ∎

[44] This holds since by (9.8), we have $\tilde{f}_\ell = f_{m,\ell}$ for every $\ell \in \{i, i+1, \ldots, m\}$.

[45] The success probability of $2/3$ in Lemma 9.4 can be boosted to any constant via running IC multiple times with independent randomness. This do no affect the circuit complexity of IC.

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

$$G_k(\vec{x}, \vec{y}) \coloneqq \sum_{j \in [k]} h_{k,j}(x) \cdot y_j, \tag{9.9}$$

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

Since all the $h_{k,j}$ have degree at most n, G_k has degree at most $n+1$.

Construction of field-transferring polynomials $H^{\mathsf{int}}_{k,j}$. We call an integer k *special*, if $\mathbb{F}_{n_k} \neq \mathbb{F}_{n_{k+1}}$. For a special $k \in \mathbb{N}_{\geq 1}$, we next define $n+k+1$ field-transferring polynomials $H^{\mathsf{int}}_{k,0}, H^{\mathsf{int}}_{k,1}, \ldots, H^{\mathsf{int}}_{k,n+k}$.

Since $\mathbb{F}_{n_k} \neq \mathbb{F}_{n_{k+1}}$, from the definition of n_k's and the sequence in (9.7), it must be the case that $n_{k+1} = n_k + 1$. From now on we use n to denote n_k for simplicity. Also, from the definition of $\mathbb{F}_n$ and $\mathbb{F}_{n+1}$, we must have $\mathsf{sz}_{n+1} = 3\mathsf{sz}_n$.

Let $\mathbb{F}_{\mathsf{old}} = \mathbb{F}_n = \mathsf{GF}(2^{\mathsf{sz}_n})$ and $\mathbb{F}_{\mathsf{new}} = \mathbb{F}_{n+1} = \mathsf{GF}(2^{3\mathsf{sz}_n})$. Slightly abusing notation, in the following we use the embedding τ_{ℓ_n} to identify $\mathbb{F}_{\mathsf{old}}$ with the unique subfield of $\mathbb{F}_{\mathsf{new}}$ that is isomorphic to $\mathbb{F}_{\mathsf{old}}$. Formally, for every $u \in \mathbb{F}_{\mathsf{old}}$, we identify it with the element $\tau_{\ell_n}(u) \in \mathbb{F}_{\mathsf{new}}$.

Let $H_k \colon \mathbb{F}_{\mathsf{new}}^{d+k} \to \mathbb{F}_{\mathsf{new}}$ be the unique extension of $G_k \colon \mathbb{F}_{\mathsf{old}}^{n+k} \to \mathbb{F}_{\mathsf{old}}$. For every $j \in \{0, 1, \ldots, n+k\}$, we also let $H^{\mathsf{int}}_{k,j} \colon \mathbb{F}_{\mathsf{new}}^{j} \times \mathbb{F}_{\mathsf{old}}^{n+k-j} \to \mathbb{F}_{\mathsf{new}}$ be the unique extension of G_k to the domain $\mathbb{F}_{\mathsf{new}}^{j} \times \mathbb{F}_{\mathsf{old}}^{n+k-j}$. Note that

$$H^{\mathsf{int}}_{k,j}(\vec{x}, \vec{y}) = H_k(\vec{x}, \vec{y}) \tag{9.10}$$

for every $\vec{x} \in \mathbb{F}_{\mathsf{new}}^{j}$ and $\vec{y} \in \mathbb{F}_{\mathsf{old}}^{n+k-j}$ (*i.e.*, $H^{\mathsf{int}}_{k,j}$ is the restriction of H_k on the domain $\mathbb{F}_{\mathsf{new}}^{j} \times \mathbb{F}_{\mathsf{old}}^{n+k-j}$). Note that $H^{\mathsf{int}}_{k,0}$ is simply G_k with outputs embedded in $\mathbb{F}_{\mathsf{new}}$.

The following claim shows that the the sequence $\{H^{\mathsf{int}}_{k,j}\}$ satisfies TC^0 downward self-reducibility.

Claim 7 (Downward self-reduction for $\{H^{\mathsf{int}}_{k,j}\}$). *There is an algorithm* $\mathsf{H\text{-}Red}$ *satisfying the following:*

1. $\mathsf{H\text{-}Red}$ *takes* $k \in \mathbb{N}_{\geq 1}$, $j \in [n+k]$, *and* $(\vec{y}, \vec{z}) \in \mathbb{F}_{\mathsf{new}}^{j} \times \mathbb{F}_{\mathsf{old}}^{n+k-j}$ *as input, and an oracle* $h \colon \mathbb{F}_{\mathsf{new}}^{j-1} \times \mathbb{F}_{\mathsf{old}}^{n+k-j+1} \to \mathbb{F}_{\mathsf{new}}$, *and outputs an element in* $\mathbb{F}_{\mathsf{new}}$.
2. *For every* $k \in \mathbb{N}_{\geq 1}$ *and* $j \in [n+k]$, $\mathsf{H\text{-}Red}_{k,j}^{H^{\mathsf{int}}_{k,j-1}}$ *computes* $H^{\mathsf{int}}_{k,j}$.
3. $\mathsf{H\text{-}Red}$ *can be implemented by a uniform non-adaptive* TC^0 *oracle circuit family.*

Proof. Note that $\deg(H_k) = \deg(G_k) = n+1$. We set $D = n+1$. In particular, let $(\vec{y}, \vec{z}) \in \mathbb{F}_{\mathsf{new}}^{j} \times \mathbb{F}_{\mathsf{old}}^{n+k-j}$ be an input to $\mathsf{H\text{-}Red}_{k,j}$ (and $H^{\mathsf{int}}_{k,j}$). We define the following polynomial

$$P(\mathbf{x}) = H_k(\vec{y}_{<j}, \mathbf{x}, \vec{z}).$$

Clearly $P(\mathbf{x})$ has degree at most D. Let $w_1, \ldots, w_{D+1}$ be the first $D+1$ elements in $\mathbb{F}_{\mathsf{old}}$. Our algorithm $\mathsf{H\text{-}Red}_{k,j}$ first queries the oracle h to compute $b_i = h((\vec{y}_{<i}, w_i, \vec{z}))$ for every $i \in [D+1]$, and then runs $D^{\mathsf{intp}}_{n+1,D+1}$ with the list $\{(w_i, b_i)\}_{i \in [D+1]}$ ($w_i \in \mathbb{F}_{\mathsf{old}}$ is interpreted as an element of $\mathbb{F}_{\mathsf{new}}$ via τ_{ℓ_n}) and the input y_j, and finally returns the output of $D^{\mathsf{intp}}_{n+1,D+1}$.

Item (1) of the claim follows directly from Corollary 9.3. To see Item (2) holds, we note that when $h = H^{\mathsf{int}}_{k,j-1}$, by the definition of $P(\mathbf{x})$, we have that $b_i = P(w_i)$ for every $i \in [D+1]$. Since $P(\mathbf{x})$ has degree at most D, $\mathsf{H\text{-}Red}_{k,j}$ returns $P(y_j)$, which equals $H_k(\vec{y}, \vec{z}) = H_{k,j}(\vec{y}, \vec{z})$ by definition. ∎

Converting G_k and $H^{\mathsf{int}}_{k,j}$ into Boolean functions via Walsh-Hadamard codes. Next, we convert the polynomials G_k and $H^{\mathsf{int}}_{k,i}$ into Boolean functions by applying Walsh-Hadamard codes.

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

$$F_k(\vec{z},\vec{r}) := \langle \kappa_n^{-1}(G_k(\vec{z})), \vec{r}\rangle, \tag{9.11}$$

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 $e_k = (n_k + k + 1)\cdot \mathsf{sz}_n$ (we write n_k instead of n to emphasize that it is a function of k).

Recall that an integer k is special if G_k and G_{k+1} are over different fields. In this case, we know that $\mathbb{F}_{n+1} = \mathsf{GF}(2^{3\mathsf{sz}_n})$, and $H^{\mathsf{int}}_{k,j}$ is from $\mathbb{F}^j_{n+1} \times \mathbb{F}_n^{n+k-j} \to \mathbb{F}_{n+1}$. Similarly, for every $j \in \{0,1,\ldots,n+k\}$, we define $F^{\mathsf{trans}}_{k,j}\colon \mathbb{F}^j_{n+1} \times \mathbb{F}_n^{n+k-j} \times \{0,1\}^{3\mathsf{sz}_n} \to \{0,1\}$ as

$$F^{\mathsf{trans}}_{k,j}(\vec{z},\vec{r}) := \langle \kappa_{n+1}^{-1}(H^{\mathsf{int}}_{k,j}(\vec{z})), \vec{r}\rangle. \tag{9.12}$$

$F^{\mathsf{trans}}_{k,j}$ can be interpreted as a Boolean function on $\{0,1\}^{e_{k,j}}$, where $e_{k,j} = (n_k + k + j + 3)\cdot \mathsf{sz}_n$.

The following claim is useful.

Claim 8. *For every $k \in \mathbb{N}_{\ge 1}$, it holds that $e_k < e_{k+1}$. Moreover, the following holds for every special k:*

$$e_k < e_{k,0} < e_{k,1} < \ldots < e_{k,n_k+k-1} < e_{k,n_k+k} < e_{k+1}.$$

The language L^{PSPACE}. Now we are ready to define L^{PSPACE} via the following algorithm.

Algorithm 9.3: Algorithm A^{PSPACE} for L^{PSPACE}

```
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;
if k is not speical then
    return F_k(x_{≤e_k});
if m < e_{k,0} then
    return F_k(x_{≤e_k});
Let j be the largest non-negative integer such that e_{k,j} ≤ m;
j ← min(j, n_k + k);
return F^trans_{k,j}(x_{≤e_{k,j}});
```

From Claim 8 and Algorithm 9.3, the following claim is immediate.

Claim 9. *For every $k \in \mathbb{N}_{\ge 1}$, $L^{\mathsf{PSPACE}}_{e_k}$ equals F_k. For every special k and every $j \in \{0,1,\ldots,n_k+k\}$, $L^{\mathsf{PSPACE}}_{e_{k,j}}$ equals $F^{\mathsf{trans}}_{k,j}$.*

9.3.3. Verifying Properties of L^{PSPACE}. Next, we verify that L^{PSPACE} has all the desired properties stated in Theorem 3.7.

Lemma 9.11. *L^{PSPACE} is paddable and non-adaptive TC^0 downward self-reducible.*

Proof. We first note that to show L^{PSPACE} is paddable, it suffices to verify its paddability from input length m to $m+1$. Hence in the following, we prove the paddability and downward self-reducibility for every input length $m \in \mathbb{N}_{\geq 1}$ and $m+1$.

When A_m^{PSPACE} and $A_{m+1}^{\mathsf{PSPACE}}$ (we use A_m^{PSPACE} to denote the restriction of A^{PSPACE} on m-bit inputs) computes the same function on their prefixes, one can simply define $\mathsf{Pad}(x, 1^{m+1}) = x \circ 0$ to establish paddability, and the self-reducibility follows from the fact that $L_{m+1}^{\mathsf{PSPACE}}(x) = L_m^{\mathsf{PSPACE}}(x_{\leq m})$ for every $x \in \{0,1\}^{m+1}$. Hence, according to Algorithm 9.3, there are following four non-trivial cases:[46]

1. A_m^{PSPACE} computes F_k and $A_{m+1}^{\mathsf{PSPACE}}$ computes F_{k+1}.
2. A_m^{PSPACE} computes $F_{k,n_k+k}^{\mathsf{trans}}$, $A_{m+1}^{\mathsf{PSPACE}}$ computes F_{k+1} for a special k.
3. A_m^{PSPACE} computes F_k, $A_{m+1}^{\mathsf{PSPACE}}$ computes $F_{k,0}^{\mathsf{trans}}$ for a special k.
4. A_m^{PSPACE} computes $F_{k,j}^{\mathsf{trans}}$, $A_{m+1}^{\mathsf{PSPACE}}$ computes $F_{k,j+1}^{\mathsf{trans}}$ for a special k and $j \in \{0, 1, \ldots, n_k + k - 1\}$.

Now we discuss these four cases separately. In the rest of the proof we always use n to denote n_k for simplicity. We first note that to verify paddability and self-reduction in Case 1, it suffices to verify that there is a projection that reduces F_k to F_{k+1} and a uniform non-adaptive TC^0 circuit computing F_{k+1} given oracle to F_k. Similarly, to verify paddability and self-reduction in Case 2, 3, and 4, it suffices to establish the desired reductions between (1) F_{k,n_k+k} and F_{k+1}, (2) F_k and $F_{k,0}^{\mathsf{trans}}$, and (3) $F_{k,j}^{\mathsf{trans}}$ and $F_{k,j+1}^{\mathsf{trans}}$.

Case 1 and Case 2. We will handle these two cases together. To do so, we begin by setting up some notation. We first set $\mathbb{F}_{\mathsf{new}} = \mathbb{F}_{n+1}$. We also set G_{old} and F_{old} depending on whether we are in Case 1 and Case 2 as follows:

1. In Case 1, we set $G_{\mathsf{old}} = G_k$ and $F_{\mathsf{old}} = F_k$;
2. In Case 2, we set $G_{\mathsf{old}} = H_{k,n_k+k}^{\mathsf{int}}$ and $F_{\mathsf{old}} = F_{k,n_k+k}^{\mathsf{trans}}$.

Our goal now (for both cases) is to verify the paddability from F_{old} to F_{k+1}, and the downward self-reducibility from F_{k+1} to F_{old}.

From Algorithm 9.3, we can see that the polynomials G_{old} and G_{k+1} are over the same field $\mathbb{F}_{\mathsf{new}}$. We first verify the paddability. From the definition of G_k and G_{k+1} in (9.9) (Case 1) and the definition of $H_{k,n+k}^{\mathsf{int}}$ in (9.10) (Case 2), we have

$$G_{\mathsf{old}}(\vec{x}, \vec{y}) = G_{k+1}(\vec{x}, \vec{y}, (0)_{\mathbb{F}_{\mathsf{new}}})$$

for every $\vec{x} \in \mathbb{F}_{\mathsf{new}}^n$ and $\vec{y} \in \mathbb{F}_{\mathsf{new}}^k$. Hence, by the definition of F_{old} and F_{k+1}, we have

$$F_{\mathsf{old}}(\vec{x}, \vec{y}, \vec{z}) = F_{k+1}(\vec{x}, \vec{y}, (0)_{\mathbb{F}_{\mathsf{new}}}, \vec{z})$$

for every $\vec{x} \in \mathbb{F}_{\mathsf{new}}^n$, $\vec{y} \in \mathbb{F}_{\mathsf{new}}^k$ and $\vec{z} \in \{0,1\}^{\log_2 |\mathbb{F}_{\mathsf{new}}|}$. Hence, the projection $(\vec{x}, \vec{y}, \vec{z}) \mapsto (\vec{x}, \vec{y}, (0)_{\mathbb{F}_{\mathsf{new}}}, \vec{z})$ is the required reduction from F_{old} to F_{k+1}.

Next we verify the downward self-reducibility, for which we have to show how to compute F_{k+1} using a uniform non-adaptive TC^0 circuit with an F_{old} oracle. We first note that by the definition of G_k and G_{k+1} in (9.9) (Case 1) and the definition of $H_{k,n_k+k}^{\mathsf{int}}$ in (9.10) (Case 2), we have

$$G_{k+1}(\vec{x}, \vec{y}) = G_{\mathsf{old}}(\vec{x}, \vec{y}_{\leq k}) + g_{k+1}(\vec{x}) \cdot y_{k+1} \tag{9.13}$$

for every $\vec{x} \in \mathbb{F}_{\mathsf{new}}^{n_{k+1}}$ and $\vec{y} \in \mathbb{F}_{\mathsf{new}}^{k+1}$.

[46]For convenience, we will simply say that A_m^{PSPACE} computes F_k (resp. $F_{k,j}^{\mathsf{trans}}$) when it computes F_k (resp. $F_{k,j}^{\mathsf{trans}}$) on its prefix of length e_k (resp. $e_{k,j}$).

We first show how to compute G_{k+1} with oracle access to G_{old}. From (9.13), it suffices to compute $g_{k+1}(\vec{x})$ with oracle access to G_{old}. Recall that $g_{k+1} = f_{n_{k+1},i_{k+1}}$. We first note that if $i_{k+1} = n_{k+1}$, then by Item (2) of Lemma 9.4, we can compute $g_{k+1}(\vec{x})$ by a uniform TC^0 circuit directly without oracle access to G_{old}.

So we can assume that $i_{k+1} < n_{k+1}$. In this case, we also have $n = n_k = n_{k+1}$ and $i_k = i_{k+1} + 1$ from (9.7). In other words, k is not special and we are in Case 1. (So $G_{\mathsf{old}} = G_k$.) Note that

$$G_k(\vec{x}, \vec{z}) = f_{n,i_k}(\vec{x}) \quad \text{for } \vec{z} = (0, 0, \ldots, 0, 1) \in \mathbb{F}_{\mathsf{new}}^k \text{ and every } \vec{x} \in \mathbb{F}_{\mathsf{new}}^n. \tag{9.14}$$

Therefore, to compute $G_{k+1}(\vec{x}, \vec{y})$ according to (9.13), we first compute $G_k(\vec{x}, \vec{y}_{\le k})$ via an oracle call to G_k, and then compute $g_{k+1}(\vec{x}) = f_{n,i_{k+1}}(\vec{x})$ by applying the algorithm $\mathsf{Red}_{n,i_{k+1}}$ with the oracle to f_{n,i_k} simulated by G_k using (9.14). Finally, we compute $G_k(\vec{x}, \vec{y}_{\le k}) + g_{k+1}(\vec{x}) \cdot y_{k+1}$ using the algorithm from Lemma 9.2. A straightforward implementation gives a uniform non-adaptive TC^0 circuit computing G_{k+1} given oracle to G_k.

Finally, note that a single query to G_{old} can be simulated by $\log |\mathbb{F}_{\mathsf{new}}|$ queries to F_{old} and recall the definition of F_{k+1} in (9.11), we can obtain the desired oracle circuit computing F_{k+1} given oracle to F_{old}.

Notation. In the next two cases, k is special and we recall some notation for convenience. Since k is special, we have that $n_{k+1} = n_k + 1 = n+1$ and $\mathsf{sz}_{n+1} = 3 \cdot \mathsf{sz}_n$. We let $\mathbb{F}_{\mathsf{old}} = \mathbb{F}_n$ and still set $\mathbb{F}_{\mathsf{new}} = \mathbb{F}_{n+1}$.

Case 3. Recall the definition of $H_{k,0}^{\mathsf{int}}$ in (9.10) and that $H_{k,0}^{\mathsf{int}} \colon \mathbb{F}_{\mathsf{old}}^{n+k} \to \mathbb{F}_{\mathsf{new}}$ is simply $G_k \colon \mathbb{F}_{\mathsf{old}}^{n+k} \to \mathbb{F}_{\mathsf{old}}$ with outputs embedded in $\mathbb{F}_{\mathsf{new}}$. To compute $G_k(\vec{z})$ given an oracle to $H_{k,0}^{\mathsf{int}}$, we simply apply $\mathsf{Emd}_{\ell_n}^{-1}$ to the Boolean encoding of $H_{k,0}^{\mathsf{int}}(\vec{z})$. Similarly, to compute $H_{k,0}^{\mathsf{int}}(\vec{z})$ given an oracle to G, we simply apply Emd_{ℓ_n} to the Boolean encoding of $G(\vec{z})$. Finally, using a similar argument as in Case 1 and 2, we can lift these reductions between G_k and $H_{k,0}^{\mathsf{int}}$ into the required reductions between F_k and $F_{k,0}^{\mathsf{trans}}$. This completes the proof for this case.

Case 4. Similar to the three cases above, it suffices to establish the paddability from $H_{k,j}^{\mathsf{int}}$ to $H_{k,j+1}^{\mathsf{int}}$ and the downward self-reducibility from $H_{k,j+1}^{\mathsf{int}}$ to $H_{k,j}^{\mathsf{int}}$. Note that the required downward self-reducibility follows directly from Claim 7. To see the paddability, note that

$$H_{k,j}^{\mathsf{int}}(\vec{y}_{\le j}, y_{j+1}, \vec{z}) = H_{k,j+1}^{\mathsf{int}}(\vec{y}_{\le j}, \tau_{\ell_n}(y_{j+1}), \vec{z}) \qquad \square$$

for every $\vec{y} \in \mathbb{F}_{\mathsf{new}}^j$ and $\vec{z} \in \mathbb{F}_{\mathsf{old}}^{n+k-j}$. Recall that $\tau_{\ell_n}(y_{j+1})$ can be computed by applying the polynomial-time computable projection Emd_ℓ (see Lemma 9.1) on the Boolean encoding of y_{j+1}. Hence $(\vec{y}_{\le j}, y_{j+1}, \vec{z}) \mapsto (\vec{y}_{\le j}, \tau_{\ell_n}(y_{j+1}), \vec{z})$ is the desired projection padding from $H_{k,j}^{\mathsf{int}}$ to $H_{k,j+1}^{\mathsf{int}}$. This completes the whole proof.

Next we show the PSPACE-completeness of L^{PSPACE}.

LEMMA 9.12. *L^{PSPACE} is PSPACE-complete.*

Proof. We first note that $L^{\mathsf{PSPACE}} \in \mathsf{PSPACE}$ since every downward self-reducible language is in PSPACE (see, *e.g.*, [7, Exercise 8.9]).

Let $L \in \mathsf{SPACE}$, and let $(A_L^{\mathsf{len}}, A_L^{\mathsf{red}})$ be the pair of algorithms in Lemma 9.4. The following is a polynomial-time reduction R_L from L to L^{PSPACE}:

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 $\vec{y} \in \mathbb{F}_m^k$ be such that $y_k = 1$ and $y_j = 0$ for $j \in [k-1]$, 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 $L_{e_k}^{\mathsf{PSPACE}}(\vec{z}, \vec{y}, \vec{u})$.

By Lemma 9.4, 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

$$\left(\kappa_m^{-1}(f_{m,1}(\vec{z}))\right)_1 = L(x). \tag{9.15}$$

Now, by the definition of G_k in (9.9), 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 9 and (9.15), we have

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

Therefore, L^{PSPACE} is PSPACE-complete. ∎

Next we prove that L^{PSPACE} is weakly error correctable. We need the following local decoding procedure for Reed-Muller codes from [25].

LEMMA 9.13. *Let $n, m, d \in \mathbb{N}$ such that $m, d \le 2n^2$. Let $\mathbb{F}_{\mathsf{old}} = \mathbb{F}_n$ and $\mathbb{F}_{\mathsf{new}} = \mathbb{F}^{(\ell_n+1)}$. For every $i \in \{0, 1 \ldots, m\}$, there is a randomized algorithm $\mathsf{RM\text{-}Dec}_{n,m,i}$ satisfying the following:*

1. *$\mathsf{RM\text{-}Dec}_{n,m,i}$ takes $\vec{x} \in \mathbb{F}_{\mathsf{new}}^i$ and $\vec{y} \in \mathbb{F}_{\mathsf{old}}^{m-i}$ as input, and a function $f\colon \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i} \to \mathbb{F}_{\mathsf{new}}$ as oracle, and outputs an element of $\mathbb{F}_{\mathsf{new}}$.*
2. *If there is a degree-d polynomial $P\colon \mathbb{F}_{\mathsf{new}}^m \to \mathbb{F}_{\mathsf{new}}$ that agrees with f on a 0.9 fraction of the inputs from $\mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$, then*

$$\Pr[\mathsf{RM\text{-}Dec}_{n,m,i}^f(\vec{x}, \vec{y}) = P(\vec{x}, \vec{y})] \ge 2/3$$

   for every $\vec{x} \in \mathbb{F}_{\mathsf{new}}^i$ and $\vec{y} \in \mathbb{F}_{\mathsf{old}}^{m-i}$.
3. *$\mathsf{RM\text{-}Dec}$ can be implemented by a randomized uniform non-adaptive NC^3 oracle circuit family.*

For completeness, we provide a proof of Lemma 9.13 in Appendix A.

LEMMA 9.14. *L^{PSPACE} is non-adaptive NC^3 weakly error correctable.*

Proof. Let $m \in \mathbb{N}$ be an input length. If $m < e_1$, then according to Algorithm 9.3, L_m^{PSPACE} is the all-zero function and the lemma holds trivially. So we assume $m \ge e_1$.

Now there are two cases: (1) A_m^{PSPACE} computes F_k on its length-e_k prefix for some $k \in \mathbb{N}$ and (2) A_m^{PSPACE} computes $F_{k,j}^{\mathsf{trans}}$ on its length-$e_{k,j}$ prefix for some special $k \in \mathbb{N}$ and $j \in \{0, 1, \ldots, n_k + k\}$. To prove the lemma, it suffices to show weakly error correctability for F_k in Case 1 and and $F_{k,j}^{\mathsf{trans}}$ in Case 2.

Case 2. We will first focus on Case 2 and then discuss how to deal with Case 1. In the following, we use n to denote n_k for simplicity, and we let $\mathbb{F}_{\mathsf{old}} = \mathbb{F}_n$ and $\mathbb{F}_{\mathsf{new}} = \mathbb{F}^{(\ell_n+1)}$. Recall that $H_k\colon \mathbb{F}_{\mathsf{new}}^{n+k} \to \mathbb{F}_{\mathsf{new}}$ is a degree-$(n+1)$ polynomial, and $H_{k,j}^{\mathsf{int}}$ is the restriction of H_k to the domain $\mathbb{F}_{\mathsf{old}}^j \times \mathbb{F}_{\mathsf{new}}^{n+k-j}$. Also, $F_{k,j}^{\mathsf{trans}}\colon \mathbb{F}_{\mathsf{new}}^j \times \mathbb{F}_{\mathsf{old}}^{n+k-j} \times \{0,1\}^{3\mathsf{sz}_n}$ (9.12) is obtained by encoding the output of $H_{k,j}^{\mathsf{int}}$ via Walsh-Hadamard codes as follows

$$F_{k,j}^{\mathsf{trans}}(\vec{z}, \vec{r}) := \langle \kappa_{n+1}^{-1}(H_{k,j}^{\mathsf{int}}(\vec{z})), \vec{r} \rangle.$$

Let $f\colon \mathbb{F}_{\mathsf{old}}^j \times \mathbb{F}_{\mathsf{new}}^{n+k-j} \times \{0,1\}^{3\mathsf{sz}_n} \times \{0,1\}$ be an oracle that agrees with $F_{k,j}^{\mathsf{trans}}$ on a 0.99 fraction of inputs. (f and F^{trans} can be interpreted as Boolean functions via κ_n and κ_{n+1}.) We first show that there is a non-adaptive TC^0 oracle circuit $D_1 \colon \mathbb{F}_{\mathsf{new}}^j \times \mathbb{F}_{\mathsf{old}}^{n+k-j} \to \mathbb{F}_{\mathsf{new}}$ such that D_1^f agrees with $H_{k,j}^{\mathsf{int}}$ on a 0.9 fraction of inputs.

This can be done by the local decoding algorithm of Walsh-Hadamard codes [27] (see also [7, Theorem 19.18]). By a Markov inequality, for at least a 0.95 fraction of $\vec{x} \in \mathbb{F}_{\mathsf{new}}^j \times \mathbb{F}_{\mathsf{old}}^{n+k-j}$, $f(\vec{x},\vec{z}) = F_{k,j}^{\mathsf{trans}}(\vec{x},\vec{z})$ holds for at least a 0.8 fraction of $\vec{z} \in \{0,1\}^{3\mathsf{sz}_n}$. We call such $\vec{x}$ good. We first consider the following randomized oracle circuit D_2:

1. Let $c_1 \in \mathbb{N}$ be a sufficiently large constant. Draw $z_1, \ldots, z_{c_1} \in_{\mathsf{R}} \{0,1\}^{3\mathsf{sz}_n}$ independently. For every $j \in [3\mathsf{sz}_n]$, let $e_j \in \{0,1\}^{3\mathsf{sz}_n}$ be the string that $(e_j)_j = 1$ and $(e_j)_\ell = 0$ for $\ell \neq j$.
2. Given $\vec{x} \in \mathbb{F}_{\mathsf{new}}^j \times \mathbb{F}_{\mathsf{old}}^{n+k-j}$ as input, for every $j \in [3\mathsf{sz}_n]$, set b_j as the majority of $\{f(\vec{x}, z_\ell) \oplus f(\vec{x}, z_\ell \oplus e_j)\}_{\ell \in [c_1]}$.
3. Output the string $b_1, b_2, \ldots, b_{3\mathsf{sz}_n}$. (Interpreted as an element of $\mathbb{F}_{\mathsf{new}}$ via κ_{n+1}.)

A standard argument (see [7, Theorem 19.18]) shows that for every good $\vec{x}$, $D_2^f(\vec{x}) = H_{k,j}^{\mathsf{int}}(\vec{x})$ with probability at least $1 - 2^{-\Omega(c_1)} \geq 0.99$ (since c_1 is sufficiently large). Hence, by an averaging argument, we can fix the randomness in D_2 to obtain a (deterministic) oracle circuit D_1 such that D_1^f agrees with $H_{k,j}^{\mathsf{int}}$ on a 0.9 fraction of inputs. Also, we can see that D_1 (and thus also D_2) can be implemented by a non-adaptive TC^0 circuit.

Next, we apply Lemma 9.13 with parameter $(n, m, d, i) = (n, n+k, n+1, j)$ and polynomial $P = H_k$.[47] It follows that

$$\Pr\left[\mathsf{RM\text{-}Dec}_{n,n+k,j}^{D_1^f}(\vec{x}) = H_{k,j}^{\mathsf{int}}(\vec{x})\right] \geq 2/3$$

for every $\vec{x} \in \mathbb{F}_{\mathsf{new}}^j \times \mathbb{F}_{\mathsf{old}}^{n+k-j}$.

The success probability above can be amplified to $1 - \frac{1}{2|\mathbb{F}_{\mathsf{new}}|^{n+k}}$ by repeating the algorithm $\mathrm{poly}(m, \mathsf{sz}_n) \leq \mathrm{poly}(n)$ times with independently randomness, and taking a majority of the outputs. We denote the resulting randomized oracle algorithm by D_3. By a union bound over every input in $\mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$ and an averaging principle, we can fix the randomness in D_3 to obtain a nonadaptive NC^3 oracle circuit D_4[48] such that $D_4^{D_1^f}$ agrees with $H_{k,j}^{\mathsf{int}}$ on every input. Since both D_1 and D_4 are non-adaptive NC^3 oracle circuits, we can collapse them in $D_4^{D_1^f}$ to obtain a non-adaptive NC^3 oracle circuit E_1 such that $E_1^f = H_{k,j}^{\mathsf{int}}$. Finally, using the definition of $F_{k,j}^{\mathsf{trans}}$, from E_1 we can construct a non-adaptive NC^3 oracle circuit E_2 such that $E_2^f = F_{k,j}^{\mathsf{trans}}$. This completes the proof for Case 2.

Case 1. Here F_k is obtained by encoding the output of $G_k \colon \mathbb{F}_n^{n+k} \to \mathbb{F}_n$ via Walsh-Hadamard codes. We note that this is identical to the subcase of Case 2 where $j = n+k$ (and $H_{k,j}^{\mathsf{int}}$ is from $\mathbb{F}_{\mathsf{new}}^{n+k}$ to $\mathbb{F}_{\mathsf{new}}$), and we can establish the weakly error correctability for F_k in exactly the same way. ∎

LEMMA 9.15. *L^{PSPACE} is non-adaptive TC^0 same-length checkable.*

Proof. Let $m \in \mathbb{N}$ be an input length. Similar to the proof of Lemma 9.14 we assume $m \geq e_1$, and there are two cases (1) A_m^{PSPACE} computes F_k on its length-e_k

[47] From the definition of n_k (see (9.7)), it holds that $n_k + k \leq 2 \cdot n_k^2$ for $k \in \mathbb{N}_{\geq 1}$.

[48] NC^3 is closed under taking a majority and $\mathsf{RM\text{-}Dec}_{n,n+k,j}$ can be implemented by non-adaptive NC^3 oracle circuits by Lemma 9.13.

prefix for some $k \in \mathbb{N}$ and (2) A_m^{PSPACE} computes $F_{k,j}^{\mathsf{trans}}$ on its length-$e_{k,j}$ prefix for some special $k \in \mathbb{N}$ and $j \in \{0, 1, \ldots, n_k + k\}$. To prove the lemma, it suffices to establish the same-length checkability for F_k in Case 1 and for $F_{k,j}^{\mathsf{trans}}$ in Case 2.

As before, in the following we will also use n to denote n_k.

Case 1: Instance checker for G_k. We focus on Case 1 first. We first show how to establish an instance checker $\mathsf{G\text{-}IC}$ for G_k.

Recall that

$$G_k(\vec{x}, \vec{y}) = \sum_{j \in [k]} h_{k,j}(\vec{x}) \cdot y_j \tag{9.16}$$

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

Note that for every $j \in [k]$, by setting $\vec{r^j}$ such that $r_j^j = 1$ and $r_\ell^j = 0$ for $\ell \neq j$, we have

$$h_{k,j}(\vec{x}) = G_k(\vec{x}, \vec{r^j}) \quad \text{for every } \vec{x} \in \mathbb{F}_n^n, \tag{9.17}$$

meaning that the oracle access to $h_{k,j}$ can be simulated by the oracle access to G_k. $\mathsf{G\text{-}IC}$ works as follows:

1. Given $\vec{x} \in \mathbb{F}_n^n$ and $\vec{y} \in \mathbb{F}_n^k$ as input, and access to an oracle $\tilde{G} \colon \mathbb{F}_n^{n+k} \to \mathbb{F}_n$ that is supposed to compute G_k.
2. For every $j \in [k]$, letting $\varepsilon = 1/3k$, for every $j \in [k]$, $\mathsf{G\text{-}IC}$ runs $\mathsf{h\text{-}IC}_{k,j,\varepsilon}$ (from Lemma 9.10) on input $\vec{x}$ with oracle access to $\tilde{h}_1, \ldots, \tilde{h}_{j-1}$ simulated by $\tilde{G}$ via (9.17) to obtain an output $u_j \in \mathbb{F}_n \cup \{\perp\}$.[49]
3. If any of the u_j equals $\perp$, we output $\perp$. Otherwise, we output $\sum_{j \in [k]} u_j \cdot y_j$.

Since $\mathsf{h\text{-}IC}_{k,j,\varepsilon}$ can be implemented by a randomized uniform non-adaptive TC^0 oracle circuit, so can $\mathsf{G\text{-}IC}$.

Now we show that when $\tilde{G} = G_k$, $\mathsf{G\text{-}IC}$ outputs $G_k(\vec{x}, \vec{y})$ with probability 1. Note that for every $j \in [k]$, since $\tilde{G} = G_k$, we have $\tilde{h}_\ell = h_{k,\ell}$ for every $\ell \in [j-1]$ from (9.17). Applying Lemma 9.10, it holds that with probability 1, $u_j = h_{k,j}(\vec{x})$ for every $j \in [k]$. Therefore, with probability 1, $\mathsf{G\text{-}IC}$ outputs $\sum_{j \in [k]} u_j \cdot y_j$, 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, $\mathsf{G\text{-}IC}^{\tilde{G}}$ outputs either $G_k(\vec{x}, \vec{y})$ or $\perp$. We first note that by Lemma 9.10 and a union bound, with probability at least 2/3, $u_j \in \{h_{k,j}(\vec{x}), \perp\}$ for every $j \in [k]$, which implies that $\mathsf{G\text{-}IC}$ 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 $\mathsf{G\text{-}IC}$ for G_k.

Case 1: Instance checker for F_k. Next we show how to construct the desired instance checker $\mathsf{F\text{-}IC}$ for F_k:

1. Given $\vec{x} \in \mathbb{F}_n^{n+k}$ and $\vec{z} \in \{0,1\}^{\mathsf{sz}_n}$ as input, and access to an oracle $\tilde{F} \colon \mathbb{F}_n^{n+k} \times \{0,1\}^{\mathsf{sz}_n} \to \{0,1\}$ that is supposed to compute F_k.
2. $\mathsf{F\text{-}IC}$ simulates $\mathsf{G\text{-}IC}$ on input $\vec{x}$ given oracle access to the function[50]
$$\vec{x} \mapsto \tilde{F}(\vec{x}, \vec{e}_1) \circ \tilde{F}(\vec{x}, \vec{e}_2) \circ \cdots \circ \tilde{F}(\vec{x}, \vec{e}_{\mathsf{sz}_n}),$$
to obtain an output $u \in \mathbb{F}_n \cup \{\perp\}$.
3. $\mathsf{F\text{-}IC}$ outputs $\perp$ if u equals $\perp$ and outputs $\langle \kappa_n^{-1}(u), \vec{z} \rangle$ (inner product is over $\mathsf{GF}(2)$) otherwise.

[49] That is, $\tilde{h}_\ell(\vec{x}) = \tilde{G}(\vec{x}, \vec{r^\ell})$ for every $\ell \in [j-1]$.

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

Since we encode an element of $\mathbb{F}_n$ via κ_n, when $\tilde{F} = F_k$, $\mathsf{G\text{-}IC}$ 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 $\mathsf{G\text{-}IC}$, we know that $\mathsf{G\text{-}IC}$ outputs an element in $\{G_k(\vec{x}), \perp\}$ with probability at least $2/3$. This implies that $\mathsf{F\text{-}IC}$ outputs an element in $\{F_k(\vec{x}, \vec{z}), \perp\}$ with probability at least $2/3$ as well. Therefore, $\mathsf{F\text{-}IC}$ is an instance checker for F_k. Since $\mathsf{G\text{-}IC}$ can be implemented by a randomized uniform non-adaptive TC^0 oracle circuit, so can $\mathsf{F\text{-}IC}$.

Case 2: Instance checker for $H^{\mathsf{int}}_{k,j}$. We note that similar to Case 1, it suffices to construct an instance checker $\mathsf{H\text{-}IC}$ for $H^{\mathsf{int}}_{k,j}$.

In this case k is special. Let $\mathbb{F}_{\mathsf{old}} = \mathbb{F}_n$ and $\mathbb{F}_{\mathsf{new}} = \mathbb{F}_{n+1} = \mathbb{F}^{(\ell_n+1)}$. Recall that $H^{\mathsf{int}}_{k,j} \colon \mathbb{F}^j_{\mathsf{new}} \times \mathbb{F}^{n+k-j}_{\mathsf{old}} \to \mathbb{F}_{\mathsf{new}}$ is the unique extension of $G_k \colon \mathbb{F}^{n+k}_{\mathsf{old}} \to \mathbb{F}_{\mathsf{old}}$ to the domain $\mathbb{F}^j_{\mathsf{new}} \times \mathbb{F}^{n+k-j}_{\mathsf{old}}$. $\mathsf{H\text{-}IC}$ works as follows:

1. $\mathsf{H\text{-}IC}$ takes $\vec{z} \in \mathbb{F}^j_{\mathsf{new}} \times \mathbb{F}^{n+k-j}_{\mathsf{old}}$ as input, and an oracle $\tilde{H} \colon \mathbb{F}^j_{\mathsf{new}} \times \mathbb{F}^{n+k-j}_{\mathsf{old}} \to \mathbb{F}_{\mathsf{new}}$ that is supposed to compute $H^{\mathsf{int}}_{k,j}$. We let $\tilde{G} \colon \mathbb{F}^{n+k}_{\mathsf{old}} \to \mathbb{F}_{\mathsf{old}}$ be the simulated oracle to $\mathsf{G\text{-}IC}$ such that if $\tilde{H} = H^{\mathsf{int}}_{k,j}$ then $\tilde{G} = G_k$.[51]
2. $\mathsf{H\text{-}IC}$ runs $\mathsf{Ext\text{-}C}_{n,n+1,n+k,j,n+1}(\vec{z})$ with $\tilde{H}$ and $\mathsf{G\text{-}IC}^{\tilde{G}}$ as oracles.

Since both $\mathsf{G\text{-}IC}$ and $\mathsf{Ext\text{-}C}$ can be implemented by a randomized uniform TC^0 circuit family, so can $\mathsf{H\text{-}IC}$. Moreover, it is straightforward to verify that H^{int} is an instance-checker for $H^{\mathsf{int}}_{k,j}$, using the fact that $\mathsf{G\text{-}IC}$ is an instance-checker for G_k and $H^{\mathsf{int}}_{k,j}$ is the unique extension of G_k to $\mathbb{F}^j_{\mathsf{new}} \times \mathbb{F}^{n+k-j}_{\mathsf{old}}$ (so we can apply Lemma 9.9). This completes the proof for Case 2. □

Appendix A. Low-depth Decoders for Reed-Muller Codes.

In this section, we prove Lemma 9.13 (restated below).

Reminder of Lemma 9.13. *Let $n, m, d \in \mathbb{N}$ such that $m, d \le 2n^2$. Let $\mathbb{F}_{\mathsf{old}} = \mathbb{F}_n$ and $\mathbb{F}_{\mathsf{new}} = \mathbb{F}^{(\ell_n+1)}$. For every $i \in \{0, 1 \ldots, m\}$, there is a randomized algorithm $\mathit{RM\text{-}Dec}_{n,m,i}$ satisfying the following:*

1. *$\mathit{RM\text{-}Dec}_{n,m,i}$ takes $\vec{x} \in \mathbb{F}^i_{\mathsf{new}}$ and $\vec{y} \in \mathbb{F}^{m-i}_{\mathsf{old}}$ as input, and a function $f \colon \mathbb{F}^i_{\mathsf{new}} \times \mathbb{F}^{m-i}_{\mathsf{old}} \to \mathbb{F}_{\mathsf{new}}$ as oracle, and outputs an element of $\mathbb{F}_{\mathsf{new}}$.*
2. *If there is a degree-d polynomial $P \colon \mathbb{F}^m_{\mathsf{new}} \to \mathbb{F}_{\mathsf{new}}$ that agrees with f on a 0.9 fraction of the inputs from $\mathbb{F}^i_{\mathsf{new}} \times \mathbb{F}^{m-i}_{\mathsf{old}}$, then*
$$\Pr[\mathit{RM\text{-}Dec}^f_{n,m,i}(\vec{x}, \vec{y}) = P(\vec{x}, \vec{y})] \ge 2/3$$
for every $\vec{x} \in \mathbb{F}^i_{\mathsf{new}}$ and $\vec{y} \in \mathbb{F}^{m-i}_{\mathsf{old}}$.
3. *$\mathit{RM\text{-}Dec}$ can be implemented by a randomized uniform non-adaptive NC^3 oracle circuit family.*

We first need the following standard unique decoding for Reed-Solomon (RS) codes from [63] (see also [7, Theorem 19.15]).

Lemma A.1 ([63]). *For $n, d, m \in \mathbb{N}$, there is an algorithm $\mathsf{RS\text{-}Dec}_{n,d,m}$ that takes a list $(a_1, b_1), \ldots, (a_m, b_m) \in \mathbb{F}_n \times \mathbb{F}_n$ as input and satisfies the following:*

1. *If there is a degree-d polynomial $G \colon \mathbb{F}_n \to \mathbb{F}_n$ satisfying $G(a_i) = b_i$ for at least $t > \frac{m}{2} + \frac{d}{2}$ of the numbers $i \in [m]$, then $\mathsf{RS\text{-}Dec}_{n,d,m}$ outputs G.*
2. *$\mathsf{RS\text{-}Dec}$ can be implemented by uniform NC^3.*

[51] In more details, for $\vec{z} \in \mathbb{F}^{n+k}_{\mathsf{old}}$, we set $\tilde{G}(\vec{z}) = \tilde{H}(\vec{z}_{\le j}, \vec{z}_{>j})$, where $\vec{z}_{\le j}$ is interpreted as a vector in $\mathbb{F}^i_{\mathsf{new}}$ via the embedding τ_{ℓ_n}.

Proof. To see that $\mathsf{RS\text{-}Dec}$ can be implemented by NC^3. We note that the computational bottleneck of the algorithm from [63] (see also the proof of [7, Theorem 19.15]) is finding a solution (that is guaranteed to exist under the assumption on t) to a system of linear equations and computing the division of two polynomials over $\mathbb{F}_n$. Now we show that both tasks can be done in uniform NC^3:

1. Chistov [22] (see also [40, Section 32]) gave a uniform $O(\log^2 n)$-depth arithmetic circuit family that computes the determinant of a square matrix over every field. Replacing all field operations by corresponding TC^0 circuits from Lemma 9.2, we get a uniform NC^3 circuit family that computes the determinant over $\mathbb{F}_n$. Also, Mulmuley [45] gave a uniform $O(\log^2 n)$-depth arithmetic circuit family that computes the rank of a matrix over every field. Similarly, we get a uniform NC^3 circuit family that computes the rank over $\mathbb{F}_n$.
   Using the reduction of [15, Theorem 5] (see also [40, Section 34]) from solving a system of linear equations to computing both rank and determinant, we obtain a uniform NC^3 circuit family for solving a system of linear equations over $\mathbb{F}_n$.
2. We show that computing the division of two polynomials $f, g \in \mathbb{F}_n[\mathbf{x}]$ can be reduced to solving a system of linear equations.
   Without loss of generality, we can assume that $1 \le \deg(g) \le \deg(f)$. Our goal now is to find a degree-$(\deg(f) - \deg(g))$ polynomial $q \in \mathbb{F}_n[\mathbf{x}]$ and another polynomial $r \in \mathbb{F}_n[\mathbf{x}]$ with degree at most $\deg(g) - 1$ such that

$$f(\mathbf{x}) = g(\mathbf{x})q(\mathbf{x}) + r(\mathbf{x}). \tag{A.1}$$

   We create a system of linear equations with $\deg(f) + 1$ unknown variables corresponding to coefficients in $q(\mathbf{x})$ and $r(\mathbf{x})$, and $\deg(f) + 1$ equations corresponding to taking the coefficients of $\mathbf{x}^i$ on each side of (A.1) for every $i \in \{0, 1, \ldots, \deg(f)\}$. Then we can apply the aforementioned NC^3 algorithm for solving a system of linear equations over $\mathbb{F}_n$. ∎

Proof of Lemma 9.13. Let $M = 20d$ and $\mathbb{D} = \mathbb{F}_{\mathsf{new}}^i \times \mathbb{F}_{\mathsf{old}}^{m-i}$. Given an input $\vec{z} \in \mathbb{D}$, $\mathsf{RM\text{-}Dec}_{n,m,i}$ draws $\vec{u} \in_{\mathsf{R}} \mathbb{D}$, and queries the oracle f on the input-set $L_{\vec{z},\vec{u}} = \{(\vec{z} + w_i \cdot \vec{u})\}_{i \in [M]}$, where w_i is the i-th non-zero element in $\mathbb{F}_{\mathsf{old}}$. We note that since $\vec{u} \in \mathbb{D}$ and $w_i \in \mathbb{F}_{\mathsf{old}}$, we have $L_{\vec{z},\vec{u}} \subseteq \mathbb{D}$. Our algorithm $\mathsf{RM\text{-}Dec}_{n,m,i}$ then runs $\mathsf{RS\text{-}Dec}_{n,d,M}$ on the list of pairs $\{(\vec{y}, f(\vec{y})) \colon \vec{y} \in L_{\vec{z},\vec{u}}\}$ to obtain a polynomial $Q \colon \mathbb{F}_{\mathsf{new}} \to \mathbb{F}_{\mathsf{new}}$ and outputs $Q(0)$.

For every $i \in [M]$, since $\vec{u}$ is drawn uniformly random from $\mathbb{D}$, it follows that $\vec{z} + w_i \cdot \vec{u}$ is also distributed uniformly random over $\mathbb{D}$. Let $E_{\vec{z},\vec{u}}$ be the number of $\vec{y} \in L_{\vec{z},\vec{u}}$ such that $f(\vec{y}) = Q(\vec{y})$. Since f agrees on a 0.9 fraction with a degree-d polynomial P on $\mathbb{D}$, by the linearity of expectation, it follows that

$$\mathop{\mathbb{E}}_{\vec{u} \in_{\mathsf{R}} \mathbb{D}}[E_{\vec{z},\vec{u}}] \ge 0.9 \cdot M.$$

Therefore, by a Markov inequality, with probability at least $2/3$ over the choice of $\vec{u}$, we have $E_{\vec{z},\vec{u}} \ge 0.7 \cdot M$. Note that

$$0.7 \cdot M > 0.5M + d/2$$

by our choice $M = 20d$. Let $Q \colon \mathbb{F}_{\mathsf{new}} \to \mathbb{F}_{\mathsf{new}}$ be such that $Q(x) = P(\vec{z} + x \cdot \vec{u})$. Note that Q has degree d as well. Hence, by Lemma A.1, it follows that $\mathsf{RS\text{-}Dec}_{n,d,M}$ recovers Q and outputs $Q(0) = P(\vec{z})$ with probability at least $2/3$ over the choice of $\vec{u}$.

Finally, note that outputting $Q(0)$ means outputs the constant term in the polynomial Q and $\mathsf{RS\text{-}Dec}_{n,d,M}$ can be implemented by uniform NC^3, it follows that $\mathsf{RM\text{-}Dec}_{n,m,i}$ can be implemented by randomized non-adaptive NC^3 oracle circuits. ∎

Appendix B. An Xor Lemma from Average-Case Hardness against $\mathsf{MAJ} \circ \mathscr{C}$ Circuits.

In this section, we provide a self-contained proof of Lemma 3.14. Our proof below follows a similar structure of the proof of [18, Lemma 3.8].

Reminder of Lemma 3.14 *Let $\mathscr{C}$ be a typical 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.*

Proof. Let $c \geq 1$ be a large enough constant. Fix $n \in \mathbb{N}$, $f \in \mathcal{F}_{n,1}$, and $\delta \in (0, 0.01)$. We will prove the following contrapositive of the lemma.

Claim 10. *For every $k \in \mathbb{N}$, $\varepsilon_k = (1-\delta)^{k-1}\left(\frac{1}{2} - \delta\right)$, and $\ell_k = c \cdot \frac{\log \delta^{-1}}{\varepsilon_k^2}$, if $f^{\oplus k}$ can be $(\frac{1}{2} + \varepsilon_k)$-approximated by a $\mathscr{C}$ circuit of size s, then f can be $(1-5\delta)$-approximated by an $\mathsf{MAJ}_{\ell_k} \circ \mathscr{C}$ circuit of size $s \cdot \ell_k + 1$.*

Note that Claim 10 holds trivially when $k = 1$. In the following we prove Claim 10 by an induction on k.

Let $k \in \mathbb{N}$ be such that $k \geq 2$. For an input x to $f^{\oplus k}$, we write $x = yz$ such that $|y| = n, |z| = (k-1)n$. Letting C be a size-s $\mathscr{C}$ circuit that $(1/2 + \varepsilon_k)$-approximates $f^{\oplus k}$ and assuming that Claim 10 holds for $k-1$, we consider the following two cases.

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

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

Then, we fix one such y, and note that since $\mathscr{C}$ is typical, either circuit $C'(z) := C(y,z)$ or $\neg C'(z)$ is a size-s $\mathscr{C}$ circuit that $(1/2 + \varepsilon_{k-1})$-approximates $f^{\oplus(k-1)}$. Hence, from our induction hypothesis, f can be $(1-5\delta)$-approximated by an $\mathsf{MAJ}_{\ell_{k-1}} \circ \mathscr{C}$ circuit of size $s \cdot \ell_{k-1} + 1$. This proves Claim 10 for k since $\ell_{k-1} \leq \ell_k$.

Case 2. Otherwise, for all $y \in \{0,1\}^n$, it holds that

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

From now on, we will use ε to denote ε_k for simplicity. We define

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

From the definition of T_y and (B.1), it follows that for every $y \in \{0,1\}^n$, we have

$$\left|T_y - \frac{1}{2}\right| \leq \frac{\varepsilon}{1-\delta}. \tag{B.2}$$

Also, since C $(\frac{1}{2}+\varepsilon)$-approximates $f^{\oplus k}$, we have

$$\mathop{\mathbb{E}}_{y}[T_y] \ge 1/2+\varepsilon. \tag{B.3}$$

We need the following claim first.

Claim 11. *For at least a $1-4\delta$ fraction of $y \in \{0,1\}^n$, it holds that $T_y > 1/2+\varepsilon/2$.*

Proof. For every $y \in \{0,1\}^n$, we set

$$U_y = \frac{\varepsilon}{1-\delta} - \left(T_y - \frac{1}{2}\right).$$

From (B.2) and (B.3), we have $U_y \ge 0$ for all y and $\mathbb{E}_y[U_y] \le \frac{\varepsilon}{1-\delta} - \varepsilon \le 2\delta\varepsilon$, where the last inequality follows from our assumption that $\delta \in (0, 0.01)$.

By a Markov inequality, we have

$$\Pr_y[U_y \ge 1/2\varepsilon] \le \frac{\mathbb{E}_y[U_y]}{1/2\varepsilon} \le 4\delta.$$

The claim then follows from the fact that $U_y < 1/2\varepsilon$ implies $T_y > 1/2+\varepsilon/2$. ◻

Recall that $\ell_k = c \cdot \frac{\log \delta^{-1}}{\varepsilon_k^2}$, where c is sufficiently large universal constant. In the following we will use ℓ_k to denote ℓ for simplicity.

Now for each $i \in [\ell]$, we draw $Z_i \in_{\mathsf{R}} \{0,1\}^{n(k-1)}$ independently. We then define

$$\widetilde{T}_y := \mathop{\mathbb{E}}_{i \leftarrow [\ell]} \left[f(y) = C(y, Z_i) \oplus f^{\oplus(k-1)}(Z_i) \right]. \tag{B.4}$$

Since c is large enough, by a Chernoff bound, it follows that for every $y \in \{0,1\}^n$,

$$\Pr_{\{Z_i\}} \left[\left| T_y - \widetilde{T}_y \right| \ge \varepsilon/6 \right] \le \delta.$$

By an averaging principle, we can fix an assignment to all the Z_i's so that

$$\left| T_y - \widetilde{T}_y \right| < \varepsilon/6 \tag{B.5}$$

holds for at least a $1-\delta$ fraction of $y \in \{0,1\}^n$.

Combining (B.5) and Claim 11, it follows that for at least a $1-5\delta$ fraction of $y \in \{0,1\}^n$, we have $\widetilde{T}_y > 1/2+\varepsilon/3$. We then construct an $\mathsf{MAJ}_\ell \circ \mathscr{C}$ E by applying MAJ_ℓ to $\{C(y,Z_i) \oplus f^{\oplus(k-1)}(Z_i)\}_{i\in[\ell]}$. Note that since $f^{\oplus(k-1)}(Z_i)$ is a constant and $\mathscr{C}$ is typical, each $C(y,Z_i) \oplus f^{\oplus(k-1)}(Z_i)$ is a $\mathscr{C}$ circuit of size at most s. Also, by (B.4), $E(y) = f(y)$ if $\widetilde{T}_y > 1/2+\varepsilon/3$.

To summarize, E is an $\mathsf{MAJ}_\ell \circ \mathscr{C}$ circuit of size at most $\ell \cdot s + 1$ that $(1-5\delta)$-approximates f. This proves Claim 10 for k. The lemma then follows from an induction on k. ◻

Appendix C. PRG Construction for Low-Depth Circuits.

In this section, we prove Theorem 3.3. Our proof is a simple combination of the local-list deocdable codes in [32] and the Nisan-Wigderson PRG construction [47].

We first state the needed black-box hardness amplification result from [32].

Theorem C.1 ([32, Theorem 8]). *There is a universal constant $c \in \mathbb{N}_{\ge 1}$ such that there are two oracle algorithms* Amp *and* Dec *satisfying the following:*

1. *Amp takes $n \in \mathbb{N}$, $\varepsilon \in (0, 1/2)$, and $x \in \{0,1\}^{cn}$ as input, a function $f \in \mathcal{F}_{n,1}$ as an oracle, and outputs a single output bit in $2^{O(n)}$ time.*
2. *Dec takes $n \in \mathbb{N}$, $\varepsilon \in (0, 1/2)$, and $x \in \{0,1\}^n$ as input, an $O(\log \varepsilon^{-1})$-bit string α as advice, a function $h \in \mathcal{F}_{cn,1}$ as an oracle, and outputs a single bit. Moreover, Dec can be implemented by a TC^0 oracle circuit of size $\mathrm{poly}(n, \varepsilon^{-1})$.*
3. *For every large enough $n \in \mathbb{N}$ and for every $\varepsilon \in (2^{-\sqrt{n}/c}, 1/2)$, for every pair of $f \in \mathcal{F}_{n,1}$ and $h \in \mathcal{F}_{cn,1}$ such that*

$$\Pr_{x \in_{\mathsf{R}} \{0,1\}^{cn}} [h(x) = \mathsf{Amp}^f(n, \varepsilon, x)] \geq 1/2 + \varepsilon,$$

there is an advice string $\alpha \in \{0,1\}^{O(\log \varepsilon^{-1})}$ such that

$$f(x) = \mathsf{Dec}^h(n, \varepsilon, x, \alpha)$$

for every $x \in \{0,1\}^n$.

We also need the following refined analysis of the Nisan-Wigderson PRG construction [47]. Let $\mathscr{F}$ be a collection of function, we use $\mathscr{F} \circ \mathsf{Junta}_a$ to denote the collection of function $g \in \mathcal{F}_{n,1}$ for some $n \in \mathbb{N}$ such that $g(x) = f(J_1(x), J_2(x), \ldots, J_\ell(x))$ for every $x \in \{0,1\}^n$, where each $J_i(x)$ is a function that depends on at most a bits of x and $f \in \mathscr{F} \cap \mathcal{F}_{\ell,1}$ for some ℓ.

LEMMA C.2. *Let $\mathscr{C}$ be a typical circuit class. There is a universal constant $c \in \mathbb{N}_{\geq 1}$ and an algorithm G such that:*
1. *G takes $\ell, m \in \mathbb{N}$ such that $\log m \leq \ell \leq m$, $Y \in \{0,1\}^{2^\ell}$ and $z \in \{0,1\}^t$ as input, where $t = c \cdot \ell^2$, and outputs an m-bit string. G is also computable in $2^{O(\ell)}$ time.*
2. *For every $\ell, m, \in \mathbb{N}$, $Y \in \{0,1\}^{2^\ell}$ and $\varepsilon \in (0, 0.5)$, let $\mathscr{F} \subseteq \mathcal{F}_{m,1}$ be a collection of functions, if $\mathsf{func}(Y)$ cannot be $(1/2+\varepsilon/m)$-approximated by $\mathscr{F} \circ \mathsf{Junta}_{\log m}$, then $G_{\ell,m}(Y, \cdot)$ is a PRG fooling all functions in $\mathscr{F}$ with error ε.*

Before proving Lemma C.2, we show it together with Theorem C.1 implies Theorem 3.3 (restated below).

Reminder of Theorem 3.3. *Let $\delta \in (0, 1)$ be a constant. There are universal constants $c \in (0, 1)$ and $g > 1$, and an algorithm G such that:*
1. *G takes two integers ℓ and m such that $\ell \leq m \leq 2^{\ell^{c\delta}}$, together with two strings $u \in \{0,1\}^{2^\ell}$ and $z \in \{0,1\}^{\ell^g}$ as inputs, and outputs an m-bit string. G is also computable in $2^{O(\ell)}$ time.*
2. *For every large enough $\ell \in \mathbb{N}$, if $f \in \mathcal{F}_{\ell,1}$ does not have ℓ^δ-depth circuits, then $G_{\ell,m}(\mathsf{tt}(f), \cdot)$ is a PRG for $\ell^{c\delta}$-depth m-input circuits with error $1/m$ and seed length ℓ^g.*

Proof. We set $c = 1/3$ and $g = 3$. Let $f \in \mathcal{F}_{\ell,1}$ be such that f does not have ℓ^δ-depth circuits. Let c_1 be the universal constant in Theorem C.1. We also set $\varepsilon = 1/m^2$.

In the following we assume that ℓ is large enough. Note that $\varepsilon \geq 2^{-2\ell^{\delta/3}} \geq 2^{-2\ell^{1/3}} \geq 2^{-\sqrt{\ell}/c_1}$. We set $g = \mathsf{Amp}^f(\ell, \varepsilon, \cdot)$ to be a function in $\mathcal{F}_{c_1\ell,1}$. From Theorem C.1, it follows that g cannot be $(1/2+\varepsilon)$-approximated by $\ell^{\delta/2}$-depth circuit, since otherwise f can be computed by a circuit of depth $O(\log \ell + \log \varepsilon^{-1}) + O(\ell^{\delta/2}) = o(\ell^\delta)$, contradicting to our hardness assumption on f.

Let G^{nw} be the algorithm in Lemma C.2. We set $G_{\ell,m}(\mathsf{tt}(f), z) = G^{\mathsf{nw}}_{c_1\ell,m}(\mathsf{tt}(g), z)$, where z is of length $O(\ell^2)$, which is at most $\ell^g = \ell^3$ since ℓ is large enough.

Now we set $\mathscr{F}$ be the set of all m-input $\ell^{\delta/3}$-depth circuits. Applying Lemma C.2 with $\mathscr{F}$ and note that all $\mathscr{F} \circ \mathsf{Junta}_{\log m}$ functions have circuits of depth at most $\ell^{\delta/3} + m = O(\ell^{\delta/3})$, it follows that $G_{\ell,m}(\mathsf{tt}(f), \cdot)$ is a PRG for m-input circuits of depth $\ell^{\delta/3} = \ell^{c\delta}$ with error $1/m$. Combining the running time of Amp in Theorem C.1 and G^{nw} in Lemma C.2, it follows that G is computable in $2^{O(\ell)}$ time. ∎

To prove Lemma C.2, we need the following construction of sets with small pairwise intersections, a.k.a. *designs*.

Lemma C.3 ([58, Lemma 2.5]). *There is a universal constant $c \in \mathbb{N}_{\geq 1}$ such that, for all integers m, ℓ such that $\log m \leq \ell \leq m$, there is a family of m sets $S_1, S_2, \ldots, S_m \subseteq [t]$ (denoted as an $(m, t, \ell, \log m)$-design), such that*

1. *$t = c \cdot \ell^2$;*
2. *for every i, $|S_i| = \ell$;*
3. *for every $i \neq j$, $|S_i \cap S_j| \leq \log m$.*

Moreover, the family is constructible in deterministic $\mathrm{poly}(m)$ time.

Now we are ready to prove Lemma C.2.

Proof. Let c be the universal constant in Lemma C.3. Given $m, \ell \in \mathbb{N}$ such that $\log m \leq \ell \leq m$, $Y \in \{0,1\}^{2^\ell}$, and $z \in \{0,1\}^t$, where $t = c \cdot \ell^2$. Let $S_1, S_2, \ldots, S_m$ be the $(m, t, \ell, \log m)$-design specified in Lemma C.3, and let $f = \mathsf{func}(Y)$. We define G as

$$G_{\ell,m}(Y, z) = f(z|_{S_1}) \circ f(z|_{S_2}) \circ \cdots \circ f(z|_{S_m}),$$

where $z|_S$ is the $|S|$-bit string obtained by taking the bits in z with indices in S.

We let $G(\cdot) = G_{m,\ell}(Y, \cdot)$ for simplicity. Suppose for the sake of contradiction that G is a not a PRG for a function $C \in \mathscr{F}$ with error ε. In other words, we have

$$\left| \mathop{\mathbb{E}}_{z \in_{\mathsf{R}} \{0,1\}^m}[C(z)] - \mathop{\mathbb{E}}_{z \in_{\mathsf{R}} \{0,1\}^t}[C(G(z))] \right| > \varepsilon.$$

In the following we will use bold font letters such as $\mathbf{X}$ to denote random variables. We also use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$. Let $\mathbf{w} \sim \mathcal{U}_t$. A standard hybrid argument implies that there is some $i \in [m]$ such that C distinguishes the following two distributions with advantage at least ε/m:

$$\begin{aligned} \mathbf{D}_{i-1} =& f(\mathbf{w}|_{S_1}) \circ f(\mathbf{w}|_{S_2}) \circ \cdots \circ f(\mathbf{w}|_{S_{i-1}}) \circ \mathcal{U}_{m-i+1}, \text{ and} \\ \mathbf{D}_i =& f(\mathbf{w}|_{S_1}) \circ f(\mathbf{w}|_{S_2}) \circ \cdots \circ f(\mathbf{w}|_{S_i}) \circ \mathcal{U}_{m-i}. \end{aligned}$$

Note that $\mathscr{C}$ is closed under negations since it is typical, we may assume that

$$\Pr[C(\mathbf{D}_{i-1}) = 1] \geq \Pr[C(\mathbf{D}_i) = 1] + \varepsilon/m.$$

We now construct a randomized $\mathscr{F} \circ \mathsf{Junta}_{\log m}$ function $\mathbf{C}'$ that $(1/2 + \varepsilon/m)$-approximates f, contradicting the hardness of Y. Given a random input $\mathbf{x} \in_{\mathsf{R}} \{0,1\}^\ell$, we fix a random seed $\mathbf{w}$ as follows. We let $\mathbf{w}|_{S_i} = \mathbf{x}$ and the other bits of $\mathbf{w}$ are independent and uniform random bits. It is easy to see that $\mathbf{w}$ distributes uniformly random over $\{0,1\}^t$. We also pick $\mathbf{z} \in_{\mathsf{R}} \{0,1\}^{m-i+1}$, to form an input

$$\mathbf{input} = f(\mathbf{w}|_{S_1}) \circ f(\mathbf{w}|_{S_2}) \circ \cdots \circ f(\mathbf{w}|_{S_{i-1}}) \circ \mathbf{z}.$$

Then we let $\mathbf{C}'(\mathbf{x}) = C(\mathbf{input}) \oplus \mathbf{z}_i$.

We show that $\mathbf{C}'$ computes f correctly with probability at least $\geq 1/2 + \varepsilon/m$, where the probability space is over both the random input $\mathbf{x}$ and the internal randomness of $\mathbf{C}'$ (*i.e.*, $\mathbf{w}|_{[t]\setminus S_i}$ and $\mathbf{z}$). Let

$$p_{\mathsf{right}} = \Pr[C(\mathbf{input}) = 1 \mid \mathbf{z}_i = f(\mathbf{x})], \quad \text{and} \quad p_{\mathsf{wrong}} = \Pr[C(\mathbf{input}) = 1 \mid \mathbf{z}_i \neq f(\mathbf{x})].$$

By the definition of $\mathbf{D}_i$ and $\mathbf{D}_{i-1}$, we have

$$\Pr[C(\mathbf{D}_i) = 1] = p_{\mathsf{right}}, \quad \text{and} \quad \Pr[C(\mathbf{D}_{i-1}) = 1] = \frac{1}{2}(p_{\mathsf{wrong}} + p_{\mathsf{right}}),$$

and

$$\begin{aligned}\Pr[\mathbf{C}'(\mathbf{x}) = f(\mathbf{x})] &= \frac{1}{2}p_{\mathsf{wrong}} + \frac{1}{2}(1 - p_{\mathsf{right}})\\ &= \frac{1}{2} + \Pr[C(\mathbf{D}_{i-1}) = 1] - \Pr[C(\mathbf{D}_i) = 1]\\ &\geq \frac{1}{2} + \varepsilon/m.\end{aligned}$$

By an averaging principle, we can fix the internal randomness of $\mathbf{C}'$ to obtain a deterministic circuit C' that $(1/2 + \varepsilon/m)$-approximates f. Since for each $j < i$, $|S_j \cap S_i| \leq \log m$, each bit of **input** depends on at most $\log m$ bits in $\mathbf{x}$. It follows that C' is in $\mathscr{F} \circ \mathsf{Junta}_{\log m}$, contradicting the hardness of f. ◻

Appendix D. Either $\mathsf{NQP} \not\subset \mathsf{NQP}$ or $\mathsf{MCSP} \not\subset \mathsf{ACC}^0$.

In this section, we prove Corollary 1.2. We will need the following lemma that is implicit in [16].

Lemma D.1 ([16]). *For every large enough $n, s \in \mathbb{N}$ such that $s \geq n$, and for every n-input s-size circuit C, there is a $(\mathsf{TC}^0)^{\mathsf{MCSP}}$ circuit C of $\mathrm{poly}(s)$ size that 0.99-approximates C.*

In the following we provide a proof for Lemma D.1 for completeness. We first need the following lemma from [48].[52]

Lemma D.2 ([48, Corollary 66]). *There exists a constant $c \geq 1$ such that, for every large enough $n, s \in \mathbb{N}$ such that $s \geq n$, and for every n-input s-size circuit C, there is an $(\mathsf{AC}^0)^{\mathsf{MCSP}}$ circuit C of $\mathrm{poly}(s)$ size that $(1/2 + 1/s^c)$-approximates C.*

Lemma D.1 is then proved by combining Lemma D.2 and Lemma 3.14.

Proof of Lemma D.1. Let C be an n-input s-size circuit, and let $k \in \mathbb{N}$ be a parameter to be specified later. Note that $C^{\oplus k}$ is a (kn)-input $10ks$-size circuit. By Lemma D.2, there is a universal constant $c \geq 1$ such that $C^{\oplus k}$ can be $(1/2+1/(10ks)^c)$-approximated by a $\mathrm{poly}(ks)$-size $(\mathsf{AC}_d)^{\mathsf{MCSP}}$ circuit for a constant $d \in \mathbb{N}$. We also set $\delta = 0.01/5$.

Now, we set $k = c_1 \cdot \log s$ for a large enough constant c_1 so that $\varepsilon_k = (1-\delta)^{k-1} \cdot (1/2 - \delta) \leq 1/(10ks)^c$. Applying (the contrapositive of) Lemma 3.14 and note that $(\mathsf{AC}_d)^{\mathsf{MCSP}}$ is a typical circuit class, it follows that there is an $\mathsf{MAJ} \circ (\mathsf{AC}_d)^{\mathsf{MCSP}}$ circuit that $(1-5\delta)$-approximates f and has size $O(\mathrm{poly}(ks) \cdot \log \delta^{-1} \cdot \varepsilon_k^{-2})$. From our choice

[52] In [48], it is stated as $(1/2 + 1/s^c)$-approximating C AC^0-reduces via tt-reductions to MCSP, which can be interpreted as a *non-adaptive* $(\mathsf{AC}^0)^{\mathsf{MCSP}}$ circuit that $(1/2+1/s^c)$-approximates C. We do not state this non-adaptive property in Lemma D.2 since it is not important for our proof. We also note in [48, Corollary 66], one can always consider $f \in \mathsf{Circuit}[n]$ by adding dummy inputs, and therefore here we can take c to be universal constant. The proof of [48, Corollary 66] builds on [16].

of δ and k, we have $1 - 5\delta = 0.99$ and $\log \delta^{-1} \cdot \varepsilon_k^{-2} \leq \mathrm{poly}(s)$, which completes the proof. ∎

Now we are ready to prove Corollary 1.2.

Reminder of Corollary 1.2 *Either* $\mathsf{NQP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$ *or* $\mathsf{MCSP} \notin \mathsf{ACC}^0$.

Proof. For the sake of contradiction, suppose $\mathsf{NQP} \subset \mathsf{P}_{/\,\mathrm{poly}}$ and $\mathsf{MCSP} \in \mathsf{ACC}^0$.

By [30, Corollary 5.1], we have that $\mathsf{MAJ} \in (\mathsf{AC}^0)^{\mathsf{MCSP}}$ and therefore $(\mathsf{TC}^0)^{\mathsf{MCSP}} \subseteq (\mathsf{AC}^0)^{\mathsf{MCSP}} \subseteq \mathsf{ACC}^0$, since $\mathsf{MCSP} \in \mathsf{ACC}^0$.

From the assumption $\mathsf{NQP} \subset \mathsf{P}_{/\,\mathrm{poly}}$, Lemma D.1, and the inclusion $(\mathsf{TC}^0)^{\mathsf{MCSP}} \subseteq \mathsf{ACC}^0$, it follows that NQP can be 0.99-approximated by ACC^0, a contradiction to Theorem 1.1. Hence, we must have either $\mathsf{NQP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$ or $\mathsf{MCSP} \notin \mathsf{ACC}^0$. ∎

Acknowledgment. This work is supported by NSF CCF-1741615 (CAREER: Common Links in Algorithms and Complexity). This work was done while the author was visiting the Simons Institute for the Theory of Computing.

I would like to thank my advisor, Ryan Williams, for his continuing support and countless valuable discussions during this work, for his suggestion to use a random self-reducible NC^1-complete problem to simplify the proof, also for many comments on an early draft of this paper.

I am grateful to Roei Tell for several detailed discussions on the proof and helpful suggestions on the presentation. I am also grateful to Chi-Ning Chou for suggestions on an early draft of this paper, and Mrinal Kumar for discussions on the complexity of the local-list decoder of Reed Solomon codes. I also would like to thank Hanlin Ren for catching an issue in the previous construction of the PSPACE-complete language.

I would like to thank the anonymous SICOMP reviewers whose detailed comments significantly improved the presentation of the paper. I also want to thank Josh Alman, Chi-Ning Chou, Shuichi Hirahara, Xuangui Huang, Nutan Limaye, Igor Carboni Oliveira, Zhao Song and Emanuele Viola for helpful discussions during this work, and the anonymous FOCS reviewers for useful comments.

REFERENCES

[1] S. Aaronson and A. Wigderson, *Algebrization: A new barrier in complexity theory*, TOCT, 1 (2009), pp. 2:1–2:54, https://doi.org/10.1145/1490270.1490272, http://doi.acm.org/10.1145/1490270.1490272.

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

[3] M. 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, 1990, pp. 1–20.

[4] M. Ajtai and M. Ben-Or, *A theorem on probabilistic constant depth computations*, in Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA, 1984, pp. 471–474, https://doi.org/10.1145/800057.808715, https://doi.org/10.1145/800057.808715.

[5] E. Allender, *The new complexity landscape around circuit minimization*, in Language and Automata Theory and Applications - 14th International Conference, LATA 2020, Milan, Italy, March 4-6, 2020, Proceedings, vol. 12038 of Lecture Notes in Computer Science, Springer, 2020, pp. 3–16, https://doi.org/10.1007/978-3-030-40608-0_1, https://doi.org/10.1007/978-3-030-40608-0_1.

[6] J. Alman and L. Chen, *Efficient construction of rigid matrices using an NP oracle*, in 60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019, IEEE Computer Society, 2019, pp. 1034–1055, https://doi.org/10.1109/FOCS.2019.00067, https://doi.org/10.1109/FOCS.2019.00067.

[7] S. Arora and B. Barak, *Computational Complexity - A Modern Approach*, Cambridge University Press, 2009, http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.
[8] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and the hardness of approximation problems*, J. ACM, 45 (1998), pp. 501–555, https://doi.org/10.1145/278298.278306, http://doi.acm.org/10.1145/278298.278306.
[9] S. Arora and S. Safra, *Probabilistic checking of proofs: A new characterization of NP*, J. ACM, 45 (1998), pp. 70–122, https://doi.org/10.1145/273865.273901, http://doi.acm.org/10.1145/273865.273901.
[10] L. Babai, *Random oracles separate PSPACE from the polynomial-time hierarchy*, Inf. Process. Lett., 26 (1987), pp. 51–53, https://doi.org/10.1016/0020-0190(87)90036-6, https://doi.org/10.1016/0020-0190(87)90036-6.
[11] T. P. Baker, J. Gill, and R. Solovay, *Relativizations of the P =? NP question*, SIAM J. Comput., 4 (1975), pp. 431–442, https://doi.org/10.1137/0204037, https://doi.org/10.1137/0204037.
[12] D. A. M. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1*, J. Comput. Syst. Sci., 38 (1989), pp. 150–164, https://doi.org/10.1016/0022-0000(89)90037-8, https://doi.org/10.1016/0022-0000(89)90037-8.
[13] E. Ben-Sasson and E. Viola, *Short pcps with projection queries*, in Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I, 2014, pp. 163–173, https://doi.org/10.1007/978-3-662-43948-7_14, https://doi.org/10.1007/978-3-662-43948-7_14.
[14] A. Bhangale, P. Harsha, O. Paradise, and A. Tal, *Rigid matrices from rectangular pcps or: Hard claims have complex proofs*, in 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020, IEEE, 2020, pp. 858–869, https://doi.org/10.1109/FOCS46700.2020.00084, https://doi.org/10.1109/FOCS46700.2020.00084.
[15] A. Borodin, J. von zur Gathen, and J. E. Hopcroft, *Fast parallel matrix and GCD computations*, Inf. Control., 52 (1982), pp. 241–256, https://doi.org/10.1016/S0019-9958(82)90766-5, https://doi.org/10.1016/S0019-9958(82)90766-5.
[16] M. L. Carmosino, R. Impagliazzo, V. Kabanets, and A. Kolokolova, *Learning algorithms from natural proofs*, in 31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan, 2016, pp. 10:1–10:24, https://doi.org/10.4230/LIPIcs.CCC.2016.10, https://doi.org/10.4230/LIPIcs.CCC.2016.10.
[17] L. Chen, *Non-deterministic quasi-polynomial time is average-case hard for ACC circuits*, in 60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019, IEEE Computer Society, 2019, pp. 1281–1304, https://doi.org/10.1109/FOCS.2019.00079, https://doi.org/10.1109/FOCS.2019.00079.
[18] L. Chen, X. Lyu, and R. R. Williams, *Almost-everywhere circuit lower bounds from non-trivial derandomization*, in 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020, IEEE, 2020, pp. 1–12, https://doi.org/10.1109/FOCS46700.2020.00009, https://doi.org/10.1109/FOCS46700.2020.00009.
[19] L. Chen and H. Ren, *Strong average-case lower bounds from non-trivial derandomization*, in Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020, ACM, 2020, pp. 1327–1334, https://doi.org/10.1145/3357713.3384279, https://doi.org/10.1145/3357713.3384279.
[20] R. Chen, I. C. Oliveira, and R. Santhanam, *An average-case lower bound against ACC^0*, in LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings, 2018, pp. 317–330, https://doi.org/10.1007/978-3-319-77404-6_24, https://doi.org/10.1007/978-3-319-77404-6_24.
[21] S. Chen and P. A. Papakonstantinou, *Depth-reduction for composites*, in IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA, 2016, pp. 99–108, https://doi.org/10.1109/FOCS.2016.20, https://doi.org/10.1109/FOCS.2016.20.
[22] A. L. Chistov, *Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic*, in International Conference on Fundamentals of Computation Theory, Springer, 1985, pp. 63–69.
[23] L. Fortnow and R. Santhanam, *Hierarchy theorems for probabilistic polynomial time*, in 45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings, 2004, pp. 316–324, https://doi.org/10.1109/FOCS.2004.33, https://doi.org/10.1109/FOCS.2004.33.

[24] M. L. Furst, J. B. Saxe, and M. Sipser, *Parity, circuits, and the polynomial-time hierarchy*, Mathematical Systems Theory, 17 (1984), pp. 13–27, https://doi.org/10.1007/BF01744431, https://doi.org/10.1007/BF01744431.

[25] P. Gemmell, R. J. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson, *Self-testing/correcting for polynomials and for approximate functions*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA, ACM, 1991, pp. 32–42, https://doi.org/10.1145/103418.103429, https://doi.org/10.1145/103418.103429.

[26] O. Goldreich, *Computational complexity - a conceptual perspective*, Cambridge University Press, 2008.

[27] O. Goldreich and L. A. Levin, *A hard-core predicate for all one-way functions*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA, 1989, pp. 25–32, https://doi.org/10.1145/73007.73010, https://doi.org/10.1145/73007.73010.

[28] O. Goldreich, N. Nisan, and A. Wigderson, *On yao's xor-lemma*, Electron. Colloquium Comput. Complex., (1995), https://eccc.weizmann.ac.il/eccc-reports/1995/TR95-050/index.html.

[29] S. Goldwasser, D. Gutfreund, A. Healy, T. Kaufman, and G. N. Rothblum, *Verifying and decoding in constant depth*, in Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007, 2007, pp. 440–449, https://doi.org/10.1145/1250790.1250855, https://doi.org/10.1145/1250790.1250855.

[30] A. Golovnev, R. Ilango, R. Impagliazzo, V. Kabanets, A. Kolokolova, and A. Tal, *$\mathsf{AC}_0[p]$ lower bounds against MCSP via the coin problem*, in 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece, vol. 132 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 66:1–66:15, https://doi.org/10.4230/LIPIcs.ICALP.2019.66, https://doi.org/10.4230/LIPIcs.ICALP.2019.66.

[31] A. Grinberg, R. Shaltiel, and E. Viola, *Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs*, in 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018, 2018, pp. 956–966, https://doi.org/10.1109/FOCS.2018.00094, https://doi.org/10.1109/FOCS.2018.00094.

[32] D. Gutfreund and G. N. Rothblum, *The complexity of local list decoding*, in Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings, 2008, pp. 455–468, https://doi.org/10.1007/978-3-540-85363-3_36, https://doi.org/10.1007/978-3-540-85363-3_36.

[33] J. Håstad, *Almost optimal lower bounds for small depth circuits*, Advances in Computing Research, 5 (1989), pp. 143–170.

[34] A. Healy and E. Viola, *Constant-depth circuits for arithmetic in finite fields of characteristic two*, in STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings, 2006, pp. 672–683, https://doi.org/10.1007/11672142_55, https://doi.org/10.1007/11672142_55.

[35] W. Hesse, E. Allender, and D. A. M. Barrington, *Uniform constant-depth threshold circuits for division and iterated multiplication*, J. Comput. Syst. Sci., 65 (2002), pp. 695–716, https://doi.org/10.1016/S0022-0000(02)00025-9, https://doi.org/10.1016/S0022-0000(02)00025-9.

[36] R. Impagliazzo, R. Jaiswal, V. Kabanets, and A. Wigderson, *Uniform direct product theorems: Simplified, optimized, and derandomized*, SIAM J. Comput., 39 (2010), pp. 1637–1665, https://doi.org/10.1137/080734030, https://doi.org/10.1137/080734030.

[37] R. Impagliazzo, V. Kabanets, and A. Wigderson, *In search of an easy witness: exponential time vs. probabilistic polynomial time*, J. Comput. Syst. Sci., 65 (2002), pp. 672–694, https://doi.org/10.1016/S0022-0000(02)00024-7, https://doi.org/10.1016/S0022-0000(02)00024-7.

[38] R. Impagliazzo and A. Wigderson, P = BPP *if* E *requires exponential circuits: Derandomizing the XOR lemma*, in Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997, ACM, 1997, pp. 220–229, https://doi.org/10.1145/258533.258590, https://doi.org/10.1145/258533.258590.

[39] J. Kilian, *Founding cryptography on oblivious transfer*, in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, 1988, pp. 20–31, https://doi.org/10.1145/62212.62215, https://doi.org/10.1145/62212.62215.

[40] D. C. Kozen, *Design and Analysis of Algorithms*, Texts and Monographs in Computer

Science, Springer, 1992, https://doi.org/10.1007/978-1-4612-4400-4, https://doi.org/10.1007/978-1-4612-4400-4.

[41] L. A. Levin, *One-way functions and pseudorandom generators*, Comb., 7 (1987), pp. 357–363, https://doi.org/10.1007/BF02579323, https://doi.org/10.1007/BF02579323.

[42] Z. Lu, I. C. Oliveira, and R. Santhanam, *Pseudodeterministic algorithms and the structure of probabilistic time*, in STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021, S. Khuller and V. V. Williams, eds., ACM, 2021, pp. 303–316, https://doi.org/10.1145/3406325.3451085, https://doi.org/10.1145/3406325.3451085.

[43] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan, *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868, https://doi.org/10.1145/146585.146605, https://doi.org/10.1145/146585.146605.

[44] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan, *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868, https://doi.org/10.1145/146585.146605, http://doi.acm.org/10.1145/146585.146605.

[45] K. Mulmuley, *A fast parallel algorithm to compute the rank of a matrix over an arbitrary field*, Comb., 7 (1987), pp. 101–104, https://doi.org/10.1007/BF02579205, https://doi.org/10.1007/BF02579205.

[46] C. Murray and R. R. Williams, *Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP*, in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018, 2018, pp. 890–901, https://doi.org/10.1145/3188745.3188910, https://doi.org/10.1145/3188745.3188910.

[47] N. Nisan and A. Wigderson, *Hardness vs randomness*, J. Comput. Syst. Sci., 49 (1994), pp. 149–167, https://doi.org/10.1016/S0022-0000(05)80043-1, https://doi.org/10.1016/S0022-0000(05)80043-1.

[48] I. C. Oliveira and R. Santhanam, *Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness*, in 32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia, vol. 79 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 18:1–18:49, https://doi.org/10.4230/LIPIcs.CCC.2017.18, https://doi.org/10.4230/LIPIcs.CCC.2017.18.

[49] A. A. Razborov, *Lower bounds on the size of bounded depth circuits over a complete basis with logical addition*, Mathematical Notes of the Academy of Sciences of the USSR, 41 (1987), pp. 333–338.

[50] A. A. Razborov and S. Rudich, *Natural proofs*, J. Comput. Syst. Sci., 55 (1997), pp. 24–35, https://doi.org/10.1006/jcss.1997.1494, https://doi.org/10.1006/jcss.1997.1494.

[51] R. Santhanam, *Circuit lower bounds for merlin–arthur classes*, SIAM J. Comput., 39 (2009), pp. 1038–1061, https://doi.org/10.1137/070702680, https://doi.org/10.1137/070702680.

[52] J. I. Seiferas, M. J. Fischer, and A. R. Meyer, *Separating nondeterministic time complexity classes*, J. ACM, 25 (1978), pp. 146–167, https://doi.org/10.1145/322047.322061, https://doi.org/10.1145/322047.322061.

[53] R. Shaltiel and E. Viola, *Hardness amplification proofs require majority*, SIAM J. Comput., 39 (2010), pp. 3122–3154, https://doi.org/10.1137/080735096, https://doi.org/10.1137/080735096.

[54] A. Shamir, *IP = PSPACE*, J. ACM, 39 (1992), pp. 869–877, https://doi.org/10.1145/146585.146609, https://doi.org/10.1145/146585.146609.

[55] R. Smolensky, *Algebraic methods in the theory of lower bounds for boolean circuit complexity*, in Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA, 1987, pp. 77–82, https://doi.org/10.1145/28395.28404, https://doi.org/10.1145/28395.28404.

[56] L. J. Stockmeyer and A. R. Meyer, *Word problems requiring exponential time*, in Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA, ACM, 1973, pp. 1–9, https://doi.org/10.1145/800125.804029, https://doi.org/10.1145/800125.804029.

[57] M. Sudan, L. Trevisan, and S. P. Vadhan, *Pseudorandom generators without the XOR lemma*, J. Comput. Syst. Sci., 62 (2001), pp. 236–266, https://doi.org/10.1006/jcss.2000.1730, https://doi.org/10.1006/jcss.2000.1730.

[58] L. Trevisan, *Extractors and pseudorandom generators*, J. ACM, 48 (2001), pp. 860–879, https://doi.org/10.1145/502090.502099, https://doi.org/10.1145/502090.502099.

[59] L. Trevisan and S. P. Vadhan, *Pseudorandomness and average-case complexity via uniform reductions*, Computational Complexity, 16 (2007), pp. 331–364, https://doi.org/10.1007/s00037-007-0233-x, https://doi.org/10.1007/s00037-007-0233-x.

[60] C. Umans, *Pseudo-random generators for all hardnesses*, J. Comput. Syst. Sci., 67 (2003), pp. 419–440, https://doi.org/10.1016/S0022-0000(03)00046-1, https://doi.org/10.1016/S0022-0000(03)00046-1.
[61] J. H. Van Lint, *Introduction to coding theory*, vol. 86, Springer-Verlag Berlin Heidelberg, 1999.
[62] E. Viola, *On approximate majority and probabilistic time*, Computational Complexity, 18 (2009), pp. 337–375, https://doi.org/10.1007/s00037-009-0267-3, https://doi.org/10.1007/s00037-009-0267-3.
[63] L. R. Welch and E. R. Berlekamp, *Error correction for algebraic block codes*, U.S. Patent 4 633 470, 1986.
[64] R. Williams, *Improving exhaustive search implies superpolynomial lower bounds*, SIAM Journal on Computing, 42 (2013), pp. 1218–1244.
[65] R. Williams, *New algorithms and lower bounds for circuits with linear threshold gates*, in Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014, 2014, pp. 194–202, https://doi.org/10.1145/2591796.2591858, http://doi.acm.org/10.1145/2591796.2591858.
[66] R. Williams, *Nonuniform ACC circuit lower bounds*, Journal of the ACM (JACM), 61 (2014), p. 2.
[67] R. Williams, *Natural proofs versus derandomization*, SIAM J. Comput., 45 (2016), pp. 497–529, https://doi.org/10.1137/130938219, https://doi.org/10.1137/130938219.
[68] A. C. Yao, *Separating the polynomial-time hierarchy by oracles (preliminary version)*, in 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985, 1985, pp. 1–10, https://doi.org/10.1109/SFCS.1985.49, https://doi.org/10.1109/SFCS.1985.49.
[69] S. Žák, *A turing machine time hierarchy*, Theor. Comput. Sci., 26 (1983), pp. 327–333, https://doi.org/10.1016/0304-3975(83)90015-4, https://doi.org/10.1016/0304-3975(83)90015-4.