Electronic Colloquium on Computational Complexity, Report No. 31 (2019)

Non-deterministic Quasi-Polynomial Time is Average-case Hard for ACC Circuits

Lijie Chen*
Massachusetts Institute of Technology

March 4, 2019

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.

We strengthen the above lower bound to an average case one, by proving that for all constants c, there is a language in NQP, which is not $(1/2 + 1/\log^c n)$-approximable by polynomial-size ACC^0 circuits. In fact, our lower bound holds for a larger circuit class: $2^{\log^a n}$-size ACC^0 circuits with a layer of threshold gates at the bottom, for all constants a. 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:

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

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

*The work was done when the author was visiting the Simons Institute for the Theory of Computing.

ISSN 1433-8092

Contents

1 Introduction

1.1 Background and Motivation

Proving *unconditional* circuit lower bounds for explicit functions (with the ultimate goal of proving $\mathsf{NP} \not\subseteq \mathsf{P}_{/\,\mathrm{poly}}$) is one of the holy grails of theoretical computer science. Back in the 1980s, there was a number of significant progress in proving circuit lower bounds for AC^0 (constant depth circuits consisting of AND/OR gates of unbounded fan-in) [Ajt83, FSS84, Yao85, Has89] and $\mathsf{AC}^0[p]$ [Raz87, Smo87] (AC^0 circuits extended with MOD_p gates) for a prime p. But this quick development was then met with an obstacle—there were little progresses in understanding the power of $\mathsf{AC}^0[m]$ for a composite m, despite it has been conjectured that they cannot even compute the majority function. In fact, it was a long-standing open question in computational complexity that whether NEXP (non-deterministic exponential time) has polynomial-size ACC^0 circuits[1], until a seminal work by Williams [Wil14b] from a few years ago, which proved NEXP does not have polynomial-size ACC^0 circuits, via a new *algorithmic* approach to circuit lower bounds [Wil13].

This circuit lower bound is an exciting new development after a long gap, especially for it surpasses all previous known barriers for proving circuits lower bounds: relativization [BGS75], algebrization [AW09], and natural proofs [RR97][2]. Moreover, the underlying approach, the algorithmic method [Wil13], puts most important classical complexity results together, ranging from non-deterministic time hierarchy theorem [Zák83], $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92], hardness vs randomness [NW94], to PCP Theorem [ALM$^+$98, AS98].

While this new circuit lower bound 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 prove an average-case lower bound.

Motivated by the above limitations, subsequent works extend the worst-case $\mathsf{NEXP} \notin \mathsf{ACC}^0$ lower bound in several ways.[3] In 2012, by refining the connection between circuit analysis algorithms and circuit lower bounds, Williams [Wil16] proved that $(\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$ does not have polynomial-size ACC^0 circuits. Two years later, by designing a fast $\#\mathsf{SAT}$ algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, Williams [Wil14a] proved that NEXP does not have polynomial-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. Then in 2017, building on [Wil16], Chen, Oliveira and Santhanam [COS18] proved that NEXP is not $1/2 + 1/\operatorname{polylog}(n)$-approximable by polynomial-size ACC^0 circuits. Recently, in an exciting new breakthrough, with a new easy-witness lemma for NQP, Murray and Williams [MW18] proved that NQP does not have polynomial-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

[1] In fact, it had been stressed several times as one of the most *embarrassing* open questions in complexity theory, see [AB09].

[2] There is no consensus that whether there is a PRG in ACC^0 (so it is not clear whether the natural proof barrier applies to ACC^0). A recent work has proposed a candidate construction [BIP$^+$18], which still needs to be tested. But we can say that *if* there is a natural proof barrier for ACC^0, then this lower bound has surpassed it. (We also remark here that there is a recent proposal on how to get a natural proof for ACC^0 circuit lower bounds via torus polynomials [BHLR19].)

[3] There are some other works [ACW16, Tam16, Wil18] proved several circuit lower bounds uncomparable to $\mathsf{NEXP} \notin \mathsf{ACC}^0$.

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 constants a, c, there is an integer b, such that $\mathsf{NTIME}[2^{\log^b n}]$ is not $(1/2 + 1/\log^c n)$-approximable 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 $\mathsf{NTIME}[2^{\log^b n}]$*[4].

In other words, the conclusion of the above theorem is equivalent to that there is a language L in $\mathsf{NTIME}[2^{\log^b n}]$ (resp. $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$) which is not $(1/2 + 1/\log^c n)$-approximable by $2^{\log^a n}$ size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits, for *all* constants $d_\star, m_\star$. We also remark that our new circuit lower bound builds crucially on another classical complexity gem: the Barrington's theorem [Bar89] together with a random self-reducible NC^1-complete language [Bab87, Kil88].

From Modest-Improvement on Gap-UNSAT Algorithms to Average-Case Lower Bounds

Like other lower bounds proved via the "algorithmic approach" [Wil13], the only property of $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits exploited by us is the non-trivial satisfiability algorithm for them [Wil14a]. Hence, our results also apply to other natural circuit classes if the corresponding algorithms are discovered.

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] Then formally, we have:

Theorem 1.2. *For a circuit class $\mathscr{C} \in \{\mathsf{TC}^0, \mathsf{Formula}, \mathsf{P}_{/\mathrm{poly}}\}$, if for a constant $\varepsilon > 0$, there is a 2^{n-n^ε} time non-deterministic Gap-UNSAT algorithm for 2^{n^ε}-size $\mathscr{C}$ circuits, then for all constants a and c, NQP is not $(1/2 + 1/n^c)$-approximable by $2^{\log^a n}$-size $\mathscr{C}$ circuits.*

Remark 1.3. *Since the circuits classes listed above can compute* majority*, we can use better hardness amplification to prove a $(1/2 + 1/n^c)$-inapproximability result, instead of the $(1/2 + 1/\log^c n)$ one. See the proof of Theorem 1.2 for the detail. We also remark that if we only want the original $(1/2 + 1/\log^c n)$-inapproximability, the above theorem holds for all circuit classes $\mathscr{C}$ closed under composition of AC^0 at the top (that is, $\mathsf{AC}^0 \circ \mathscr{C} \subseteq \mathscr{C}$).*

Remark 1.4. *One may ask whether the potentially $(1/2 + 1/n^c)$-inapproximability lower bounds from Theorem 1.2 can be used to construct PRG for the corresponding classes (that is, whether it boosts a "non-trivial" derandomization algorithm to a much faster PRG construction). While the answer is yes, such a bootstrapping result for these classes is already implicit in [Wil13, Wil16], see Appendix D for details.*

Therefore, we essentially strengthen the similar algorithmic-to-circuit-lower-bounds connections in [MW18] from worst-case lower bounds against NQP to average-case lower bounds against NQP. We remark that our connection actually *does not* rely on the "easy-witness lemma", as it is not clear how one can get an average-case easy witness lemma (i.e., NQP can be approximated by $\mathsf{P}_{/\mathrm{poly}}$ implies all NQP verifiers have succinct witnesses). Rather, we use a different approach similar

[4] See Definition 8.1 for a formal definition of $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T(n)]_{/1}$.

[5] So this problem is weaker than both the SAT problem, and the CAPP problem which asks you to estimate the accepting probability of C given a random assignment.

to [Wil16] and prove the average case lower bound *directly*, without going through the easy-witness lemma.[6]

1.3 Intuition

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

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 [Wil14b, MW18] to prove an average-case lower bound against ACC^0.

The first step of the $\mathsf{NQP} \not\subseteq \mathsf{ACC}^0$ lower bound by Murray and Williams [MW18], is applying the so called *easy witness lemma*. The easy witness lemma states: assuming $\mathsf{NQP} \subseteq \mathsf{ACC}^0$, for every language L in NQP with a verifier $V(x, y)$, whenever $V(x, \cdot)$ is satisfiable, it has a succinct witness y which is the truth-table of a small ACC^0 circuit. Then they apply a similar argument as in [Wil13, Wil14b] to contradict the *non-deterministic* time hierarchy theorem [Žák83], using the non-trivial SAT algorithm for ACC^0 circuits in [Wil14b].

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. As already explained by [COS18], 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 [IKW02, MW18] completely break when we only have the approximability assumption.

Review of [COS18]'s Approach

In order to get around the above difficulty, [COS18] start from a worst-case lower bound against ACC^0 [Wil16], and then apply a worst-case to average-case *hardness amplification*. Their approach works roughly as follows:

1. By [Wil16], there is a language $L \in (\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$, which doesn't have a $\mathrm{poly}(n)$ size ACC^0 circuit.

2. Using the locally-list-decodable codes of [GGH⁺07, GR08], one can compute a language $\widetilde{L} \in (\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$, which cannot be $(1/2 + 1/\log n)$-approximated by a $\mathrm{poly}(n)$ 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 codes, and set $\widetilde{L}_m$ to compute the codeword of z for an appropriate $m = m(n)$. (Note that here it is important to work with a language L in $(\mathsf{NEXP} \cap \mathsf{coNEXP})_{/1}$, as otherwise we don't know how to compute the truth-table of L in NEXP.)

[6] In Section 3.2, we discuss an alternative perspective on our proof: indeed, our results imply a weaker version of the average-case easy-witness lemma, which only holds for unary languages. This weaker lemma can still be used to contradict the non-deterministic time hierarchy theorem for unary languages [Žák83], see Section 3.2 for more details.

3. In particular, the above $\widetilde{L} \in \mathsf{NEXP}_{/1}$. They then get rid of the advice bit via an enumeration trick, and therefore prove the average case lower bound for NEXP.

Unfortunately, it seems very hard to generalize the above approach to prove an average-case lower bound for NQP: the second step of the above approach 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, which 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 [MW18].

The New Easy-Witness Lemma: "Almost" Almost-Everywhere (a.a.e.) MA Lower Bound and i.o. Non-deterministic PRG (NPRG)

(An instantiation of) the new easy-witness lemma of [MW18] states that if $\mathsf{NQP} \subseteq \mathsf{P}_{/\,\mathrm{poly}}$, then all verifiers for NQP languages have succinct (polynomial-size) witness. For the sake of contradiction, we now suppose $\mathsf{NQP} \subseteq \mathsf{P}_{/\,\mathrm{poly}}$ and some verifier for a language $L \in \mathsf{NQP}$ doesn't have $\mathrm{poly}(n)$-size witness. That is, there is a polynomial-time verifier $V(x, y)$ with $|x| = n$ and $y = 2^{\log^b n}$ for a constant b, such that for an infinite number of 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-randomness connection [Uma03], one can construct a pseudorandom generator G_{y_n} with seed length $O(\ell)$, running time $2^{O(\ell)}$, and 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}. This would be a non-deterministic PRG with seed length $O(\log^b S)$, running time $2^{O(\log^b S)}$, and fooling all S-size circuits.

The key ingredient of [MW18] is an "almost" almost-everywhere (a.a.e.) MA circuit lower bound, which builds on the MA circuit lower bound by Santhanam [San09].[7] For the simplicity of arguments, we now pretend that we have an almost-everywhere MA circuit lower bound. Specifically, for each c, there is an integer $k = k(c)$ such that there is a language L^c in $\mathsf{MATIME}[n^k]$, such that $\mathsf{SIZE}(L^c_n) \ge n^c$ 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 guess-and-verified). To derandomize $\mathsf{MATIME}[n^k]$, it suffices to use the PRG which fools circuits of size $S = n^{2k}$. Therefore, by setting $a = 2k$, we have a language $L^\star \in \mathsf{NTIME}[2^{\log^{b+1} n}]_{/n}$, such that it agrees with L^c on an infinite number of input lengths. Since c can be an arbitrary integer, we conclude that $\mathsf{NTIME}[2^{\log^{b+1} n}]_{/n}$ 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).

[7] [MW18, San09]'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.

Our Approach: "Almost" Almost-Everywhere Average-Case MA Lower Bound and i.o. PRG

A natural attempt to adapt the above approach, is to start with an MA "almost" almost-everywhere average-case circuit lower bound, and try to derandomize it non-deterministically via an i.o. PRG.

More precisely, assume that NQP can be $(1-\delta)$-approximated by ACC^0 circuits for the sake of contradiction. Suppose we have a language $L \in \mathsf{MAQP}$ such that for all sufficient large n, $\mathsf{heur}_{1-\delta}\text{-}\mathsf{SIZE}(L_n) \ge n^{\omega(1)}$.[8] Then with an appropriate i.o. PRG, there is a language $L^\star \in \mathsf{NQP}$ which agrees with L on an infinite number of input lengths, which contradicts our assumption as this $L^\star$ cannot be approximated by polynomial-size ACC^0.

An "Almost" Almost-Everywhere Average-Case MA Lower Bound

In order to implement this idea, the first obvious challenge is to strengthen the worst-case "almost" almost-everywhere MA circuit lower bounds [MW18] to an average-case one. This could be solved by combing ideas from the average-case circuit lower bound for MA [San09], together with a new construction of a PSPACE-complete language.

Roughly speaking, the MA circuit lower bounds in [San09] and [MW18] make crucial use of a PSPACE-complete language by [TV07], which admits several nice properties, including being same-length checkable, downward self-reducible, and paddable (see Definition 2.2 for details). We modify the construction from [TV07] to obtain a PSPACE-complete language L^{PSPACE} which is in addition *robust*: 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.

i.o. Non-deterministic PRG

The next challenge is more serious, how do we construct the required i.o. NPRG? One starting point is the (unconditional) witness-size lower bound for NE. That is, [Wil16] showed 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 $|y| = 2^n$; on an infinite number of n's, $V(1^n, \cdot)$ 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 $\mathsf{P} \subseteq \mathsf{ACC}^0$, [Wil16] showed that the above implies an i.o. NPRG for general circuits. Note that $\mathsf{P} \subseteq \mathsf{ACC}^0$ implies the Circuit-Evaluation problem has an ACC^0 circuit, and consequently $\mathsf{P}_{/\mathrm{poly}}$ collapses to ACC^0. Therefore, for a y with $V(1^n, y) = 1$, y cannot be computed by a 2^{n^ε}-size general circuit as well, which means one can substitutive y into the known hardness-to-pseudorandomness construction [NW94, Uma03], and get a quasi-polynomial time i.o. NPRG.

However, starting with our assumption NQP can be $(1-\delta)$-approximated by ACC^0, it is not clear how to show $\mathsf{P}_{/\mathrm{poly}}$ 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 [SV10, GSV18], which means we don't even know how to get a PRG fooling ACC^0 circuits, from a y which is only worst-case hard for ACC^0.

[8] $\mathsf{heur}_{1-\delta}\text{-}\mathsf{SIZE}(L_n)$ is the minimum size of a circuit computing correctly at least a $(1-\delta)$ fraction of inputs to L_n. See Section 2.1.2 for a formal definition.

i.o. Non-deterministic PRG for Low-Depth Circuits

So we want to work with a stronger circuit class, for which at least hardness amplification is possible, like NC^1. Fortunately, there is an NC^1-complete problem which admits a nice random self-reduction [Bar89, Bab87, Kil88]. By our assumption, this problem can clearly be $(1-\delta)$-approximated by ACC^0 circuits. Utilizing this random self-reduction, and the fact that approximate-majority can be computed in AC^0 [Ajt83, Vio09], we can show that this NC^1-complete problem has a poly(n)-size ACC^0 circuits. This in particular means NC^1 collapses to ACC^0. More specifically, there are two constants $d_\star, m_\star$, such that any depth d general (fan-in two) 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. non-deterministic PRG which fools polylog(n)-depth circuits.

However, in order to non-deterministically derandomize a general MA algorithm, a PRG for polylog(n)-depth NC circuits is not enough. Suppose the MA algorithm A takes an input x, guesses a string y, and flips some random coins r; in order to obtain a non-deterministic simulation, we actually want to fool circuits $C_y(r) := A(x, y, r)$, for all possible y. The circuit C_y could well be a general circuit, which does not necessarily have low depth.

An Average-Case Hard MA Language with a Low-Depth Computable Predicate

The next key observation is that we don't really need the language in MA to be average-case hard for general circuits; to obtain a contradiction, it suffices to require it cannot be approximated by *low-depth* circuits, as our assumption is that NQP can be $(1-\delta)$-approximated by ACC^0 circuits, which is contained in NC^1.

This brings us to our final technical component—an MA language L^{hard} with a low-depth computable predicate, and is average-case hard for low-depth circuits. That is, suppose the MA algorithm A takes an input x, guesses a string y, and flips some random coins r; we require that $A(x, y, r)$ ($A(x, y, r)$ is called the predicate of the MA algorithm) is computable by a uniform low-depth circuit. Now, clearly the circuit $C_y(r) := A(x, y, r)$ is 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.

The construction of such an MA language is the technical centerpiece of this paper; the key observation is that for our PSPACE-complete problem L^{PSPACE}, all its nice properties: being same-length checkable, downward self-reducible, and paddable, have corresponding low-depth uniform oracle circuits. For instance, the instance checker in the same-length checkable property (see Definition 2.2), can actually be implemented by a uniform TC^0 *non-adaptive* oracle circuit. Using these low-depth circuits in the previous proof for average-case "almost" almost-everywhere MA circuit lower bounds, together with other additional ideas, we can exhibit the language L^{hard}.

A Technicality: Dealing with Advice Bits

In the above discussion, we (intentionally) omitted a technical detail—the "almost" almost-everywhere MA lower bound proved in [MW18] is actually for $\mathsf{MA}_{/O(\log n)}$. Therefore our i.o. derandomization of the MA algorithm also needs to use these $O(\log n)$ advice bits. But then, we only have $\mathsf{NQP}_{/O(\log n)}$ is average-case hard for polynomial-size ACC^0 circuits. And the enumeration trick from [COS18] requires the advice to be $o(\log n)$.

Luckily, we further relax the definition of an "almost" almost-everywhere circuit lower bound, which is weak enough for us to prove such an MA average-case lower bound with only $O(1)$ bits of advice, but also strong enough to allow us to prove the average-case circuit lower bound. Then we can apply the enumeration trick from [COS18] to get the desired lower bound for NQP, without advice.

2 Preliminaries

We use $\mathsf{GF}(p^r)$ to denote the finite field of size p^r, where p is a prime and r is an integer.

2.1 Complexity Classes and Basic Definitions

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

2.1.1 Basic Circuit Families

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

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

AC^0 is the class of circuit families of constant depth and polynomial size, with $\mathsf{AND}, \mathsf{OR}$ and NOT gates, where AND and OR gates have unbounded fan-in. For an integer m, the function $\mathsf{MOD}_m : \{0,1\}^* \to \{0,1\}$ 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 $\mathsf{ACC}^0 = \cup_{m\geq 2}\mathsf{AC}^0[m]$.

The function majority, denoted as $\mathsf{MAJ} : \{0,1\}^* \to \{0,1\}$, is the function that outputs 1 if the number of ones in the input is no less than the number of zeros, and outputs 0 otherwise. TC^0 is the class of circuit families of constant depth and polynomial size, with unbounded fan-in MAJ gates. 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 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.[9]

We also use NC circuits to denote circuits with fan-in two AND and OR gates and unary NOT gates. For a circuit class $\mathscr{C}$, we say a circuit $C^?$ is a $\mathscr{C}$ oracle circuit, if $C^?$ is also allowed to use a special oracle gate (which can occur multiple times in the circuit, but with the same fan-in), in addition to the usual gates allowed by $\mathscr{C}$ circuits. We say an oracle circuit is *non-adaptive*, if on any path from an input gate to the output gate, there is at most one oracle gate.

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

We say 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

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

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

2.1.2 Notations

We say a circuit $C : \{0,1\}^n \to \{0,1\}$ γ-approximates a function $f : \{0,1\}^n \to \{0,1\}$, if $C(x) = f(x)$ for a γ fraction of inputs from $\{0,1\}^n$. If a circuit C does not γ-approximates a function f, we say f is not γ-approximable by C.

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

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

We say that a language L is not $\gamma(n)$-approximable by a circuit class $\mathscr{C}$ if it cannot be $\gamma(n)$-approximated by $\mathscr{C}$. That is, for each $\{C_n\}_{n\in\mathbb{N}} \in \mathscr{C}$, there is an infinite number of n's, such that L_n is not $\gamma(n)$-approximable by C_n. We say a class of language $\mathscr{L}$ is not $\gamma(n)$-approximable by a circuit class $\mathscr{C}$, if there is a language $L \in \mathscr{L}$ which is not $\gamma(n)$-approximable by $\mathscr{C}$.

2.2 Pseudorandom Generators for Low-Depth Circuits

The following PRG construction follows directly from the local-list-decodable codes with low-depth decoder [IJKW10, GGH+07, GR08], and the hardness-to-pseudorandomness transformation of [NW94].

Theorem 2.1. *Let $\delta > 0$ be a constant. There are universal constants c and g, and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, if $Y : \{0,1\}^\ell \to \{0,1\}$ does not have ℓ^δ-depth NC circuit, then for $S = 2^{\ell^{c\cdot\delta}}$, and for all NC circuit C with depth $\log(S)$,*

$$\left| \Pr_{x\in\{0,1\}^w}[C(G(Y,x)) = 1] - \Pr_{x\in\{0,1\}^S}[C(x) = 1] \right| < 1/S,$$

where $w = \ell^g$. That is, $G(Y,\cdot)$ $1/S$-fools all $\log S$-depth NC circuits. Moreover, G is computable in $2^{O(\ell)}$ time.

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

2.3 A PSPACE-complete Language with Low-complexity Reducibility Properties

A fundamental results often used in complexity theory is the existence of a PSPACE-complete language [TV07] satisfying strong reducibility properties, including the time-hierarchy theorem for

BPP with one bit of advice [FS04], the fixed polynomial circuit lower bound $\mathsf{MA}_{/1} \subseteq \mathsf{SIZE}(n^k)$ for any k [San09], and the recent new witness lemmas for NQP and NP [MW18].

The key technical ingredient of our new average-case lower bound is a modified construction of the PSPACE-complete language in [TV07], which satisfies the additional "robust" and "error correctable" properties, which are useful for proving average-case lower bound[10]. Moreover, we observe that the "reducers" in these reducibility properties of our PSPACE-complete languages are of low-complexity circuit classes (i.e., uniform polylog(n)-depth circuits). We believe this new construction would be of independent interest, and may be useful to further improvement.

We first define these reducibility properties.

Definition 2.2. Let $L : \{0,1\}^* \to \{0,1\}$ be a language, we define the following properties:

- L is $\mathscr{C}$ *downward self-reducible* if there is a constant c such that for all sufficiently large n, there is an n^c size uniform $\mathscr{C}$ circuit $A^?$ such that for all $x \in \{0,1\}^n$, $A^{L_{n-1}}(x) = L_n(x)$.

- L is *robust* if there are constants c and $\delta > 0$ such that for all sufficiently large n and $\varepsilon \geq 2^{-n^\delta}$, $\mathsf{SIZE}(L_n) \leq (\mathsf{heur}_{1/2+\varepsilon}\text{-}\mathsf{SIZE}(L_n) \cdot \varepsilon^{-1})^c$.

- L is *paddable*, if there is a polynomial time computable projection Pad (that is, each output bit is either a constant or only depends on 1 input bit), such that for all integers $1 \leq n < m$ 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.

- L is $\mathscr{C}$ *weakly error correctable* if there is a constant c such that for all sufficiently large n, for every oracle $O : \{0,1\}^n \to \{0,1\}$ which 0.99-approximates L_n, there is an n^c size $\mathscr{C}$ oracle circuit $D^?$, such that D^O exactly computes L_n.

- L is *same-length checkable* if there is a probabilistic polynomial-time oracle Turing machine M with output in $\{0,1,?\}$, such that, for any input x,

  - M asks its oracle queries only of length $|x|$.
  - If M is given L as an oracle, then M outputs $L(x)$ with probability 1.
  - M outputs $1 - L(x)$ with probability at most $1/3$ no matter which oracle is given to it.

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

Remark 2.3. *Note that 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 [San09] (modifying a construction of Trevisan and Vadhan [TV07]).

Theorem 2.4 ([San09, TV07])**.** *There is a PSPACE-complete language L_{TV} which is paddable, TC^0 downward self-reducible, and same-length checkable.*[11]

[10] The error correctable property here is stronger than the piecewise random self-reducible property in [San09].
[11] [TV07] doesn't explicitly state the TC^0 downward self-reducible property, but it is evident from their proof.

Based on the above language L_{TV}, we construct a modified PSPACE-complete language L^{PSPACE} which is also robust and NC^3 weakly error correctable. Moreover, with a careful analysis, we observe that the instance checker for L^{PSPACE} can be implemented in uniform TC^0. That is, L^{PSPACE} is TC^0 same length checkable.

Theorem 2.5. *There is a PSPACE-complete language L^{PSPACE} which is paddable, TC^0 downward self-reducible, TC^0 same-length checkable, robust and NC^3 weakly error correctable. Moreover, all the corresponding oracle circuits for the above properties are in fact* non-adaptive*: that is, on any path from an input gate to the output gate, there is at most one oracle gate.*

2.4 Average-Case Hard Languages with Low Space

We also need the following folklore result, which can be proved by applying standard worst-case to average-case hardness amplification [STV01] to a hard language in $\mathsf{SPACE}[s(n)^{O(1)}]$ obtained via diagonalization.

Theorem 2.6. *Let $n \le s(n) \le 2^{o(n)}$ be space-constructible. There is a universal constant c and a language $L \in \mathsf{SPACE}[s(n)^c]$ that $\mathsf{heur}_{1/2+1/n^3}$-$\mathsf{SIZE}(L_n) > s(n)$ for all sufficiently large n.*

3 The Structure of the Whole Proof and Alternative Perspectives

The presentation of this paper roughly follows the intuition part (so it is recommended to read the intuition part before reading the whole paper). That is, we divide the whole proof into three parts: (1) An i.o. non-deterministic PRG for low-depth circuits assuming that NQP can be approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$; (2) An Average-Case Hard MA Language with a low-depth computable predicate (this is unconditional); (3) Assuming NQP can be approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$, we combine (1) and (2) to get a contradiction.

As the whole proof is quite involved and consists of several technical ingredients, in this section, we present an outline of the whole proof, together with a diagram (Figure 1) on how all the components fit together. Moreover, to maximize helpful intuitions for the reader, we discuss an alternative perspective of our proof at the end of this section, which is closer the the original "easy-witness lemma paradigm" of [Wil13, Wil14b, MW18].

3.1 Outline of the Proof

As illustrated by Figure 1, Section 4 and Section 5 are devoted to construct the required i.o. NPRG for low-depth circuits, assuming that NQP can be approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$. More specifically:

- In Section 4, we first introduce the random self-reducible NC^1-complete language, and specify its random self-reduction. Then, in the rest of this section, we show that this language can be used to establish the collapse theorem we want.

  The proofs in this section mainly deal with some technical details (a certain amount of work is required to make sure the reduction can be implemented as a *projection*), but are conceptually very simple.

- In Section 5, we first recall the ACC^0 witness-size lower bound for NE in [Wil16], and remark that it generalizes to an $\mathsf{ACC}^0 \circ \mathsf{THR}$ witness-size lower bound for NE easily, if one makes use of

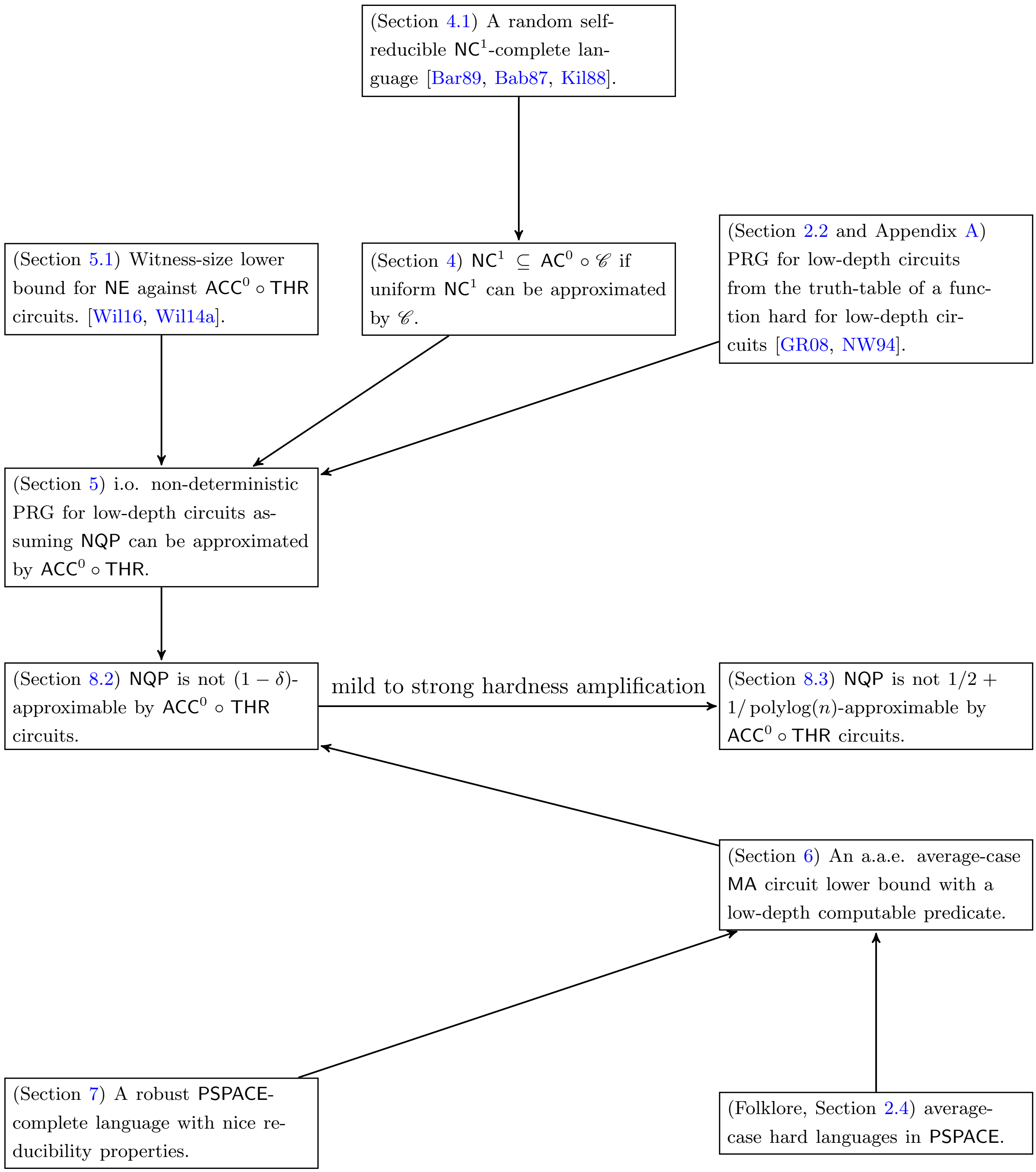

Figure 1: The structure of the whole proof.

the recent PCP construction in [BSV14] and the algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$ in [Wil14a]. Then we combine this $\mathsf{ACC}^0 \circ \mathsf{THR}$ witness-size lower bound together with the collapse theorem in Section 4 and the standard PRG construction [NW94] to construct our conditional i.o. NPRG for low-depth circuits.

The proofs in this section basically combine several previous known results together in a sophisticated way, in order to achieve what we need.

Section 6 and Section 7 are devoted to prove the needed a.a.e. average-case MA lower bound with a low-depth computable predicate. In fact, we actually prove a slightly stronger average-case lower bound for $\mathsf{MA} \cap \mathsf{coMA}$. More specifically:

- In Section 6, we first prove an average-case $\mathsf{MA} \cap \mathsf{coMA}$ a.a.e. lower bound for general circuits (that is, a strict strengthening of the corresponding worst-case lower bound in [MW18]). Next, we generalize it to an average-case $\mathsf{MA} \cap \mathsf{coMA}$ a.a.e. lower bound for low-depth circuits, and with a low-depth computable predicate. Our proofs build crucially on our new PSPACE-complete language with nice reducibility properties (constructed in the next Section), and a win-win analysis similar to that of [MW18] and [San09].

  A technical remark is that the a.a.e. MA lower bound in [MW18] is actually for $\mathsf{MA}_{O(\log n)}$, we slightly relax the a.a.e. requirement so that our lower bounds actually apply for $(\mathsf{MA} \cap \mathsf{coMA})_{/1}$. This reduction in the number of advice bits is crucially for the average-case lower bound (although the number of advice bits doesn't matter much in [MW18] as long as it is $o(n)$).

  This is the most technically involved part of this paper.

- In Section 7, we construct the needed PSPACE-complete language. Our proof builds carefully on the original PSPACE-complete language in [TV07].

  The proofs in this section apply several well-known previous results (such as the local list decoder of the Reed-Muller codes and Walsh-Hadamard codes) and several crucial observations on the PSPACE-complete language of [TV07] (the most important observation is that the instance checker of that language can be implemented in TC^0).

Then, finally in Section 8 we prove our average-case lower bound. More specifically:

- In Section 8, we first combine the conditional i.o. NPRG and the a.a.e. average-case $\mathsf{MA} \cap \mathsf{coMA}$ lower bound to show a $(1-\delta)$-inapproximability result for $(\mathsf{NQP} \cap \mathsf{coNQP})_{/O(1)}$ against $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. Then we apply mild to strong hardness amplification to strengthen that to a $1/2 + 1/\operatorname{polylog}(n)$ one (note that it is crucial to start with an $(\mathsf{NQP} \cap \mathsf{coNQP})$ lower bound in order to apply the hardness amplification). Finally we show how to get rid of the advice bits and get the desired lower bound for NQP.

  The proofs in this section basically implement the proof strategy outlined in the intuition part: combine conditional i.o. NPRG and a.a.e. average-case MA lower bound to get a contradiction. It also makes use of several previous results: mild to strong hardness amplification [IJKW10], and the enumeration trick to get rid of the advice bits while keeping the average-case hardness [COS18].

3.2 An Alternative Perspective: Average-Case Easy Witness Lemma for Unary Languages

In the intuition part we have discussed why it seems hard to prove an average-case witness lemma, and that is the reason that we took an alternative approach. But in fact, our results actually imply a weaker version of the average-case witness lemma, which is still enough to be utilized to contradict the non-deterministic time hierarchy.

More specifically, the ideal average-case easy-witness lemma would be:

Ideal Lemma. *(Average-Case Easy-Witness Lemma) For a typical circuit class $\mathscr{C}$, if NQP can be approximated by poly-size $\mathscr{C}$, then all NQP verifiers have poly-size $\mathscr{C}$ witnesses.*

Our results in fact imply the following weaker version, which only holds for NQP verifiers for *unary* languages:

Lemma 3.1. *(Average-Case Easy-Witness Lemma for Unary Languages) There is a universal constant δ such that, for a typical circuit class $\mathscr{C}$[12], if NQP can be $(1-\delta)$-approximated by poly-size $\mathscr{C}$, then all NQP verifiers for unary languages have poly-size $\mathscr{C}$ witnesses.*

For the completeness, we provide a proof sketch in Appendix C. Given the above lemma, one can still apply the non-trivial SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$ [Wil14a] to contradict the non-deterministic time hierarchy theorem for unary languages [Zák83].

This new perspective actually brings us closer to the original proof strategy of [Wil14b, Wil13], in which the last step of the proof is to contradict the non-deterministic time-hierarchy theorem. That is, the non-deterministic time-hierarchy plays the central part. While in our presentation of the proof, the last step is to derandomize the a.a.e. average-case MA lower bound to get a contradiction; the non-deterministic time-hierarchy theorem "sits in the middle", and is only used to construct the conditional i.o. NPRG.[13]

Of course, these two perspectives are mathematically equivalent[14]. But we hope clarifying this alternative perspective would provide more intuition for the reader, and hopefully stimulate future works in this direction.

4 A Collapse Theorem for NC^1

In this section we prove our collapse theorem for NC^1. In Section 4.1 we introduce the NC^1-complete language by Barrington, together with its random-self reduction. Next in Section 4.2 we define a special encoding of the input to that language. The purpose here is to make sure the random-self reduction can be implemented as a *projection*, which is crucial for the proof. Finally, in Section 4.3, we prove the needed collapse theorem.

We remark that we can also prove a similar collapse theorem for TC^0: if uniform TC^0 can be approximated by ACC^0, then TC^0 collapses to ACC^0. We include this in Appendix B as it may be of independent interest, and it does not rely on Barrington's theorem.

[12] Here we require $\mathscr{C}$ is closed under adding AC^0 at the top. That is, $\mathsf{AC}^0 \circ \mathscr{C} \subseteq \mathscr{C}$.

[13] We remark that this is similar to the proof strategy in [Wil16].

[14] One caveat here is that it seems not easy to prove lower bounds for $\mathsf{NQP} \cap \mathsf{coNQP}$, if we simply reason along Lemma 3.1. In this case, one may need hardness amplification for non-deterministic time classes [O'D04, HVV06] to get a $1/2 + 1/\operatorname{polylog}(n)$-inapproximability lower bound for NQP.

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 $m_1, m_2, \ldots, m_n \in S_5$, compute $\prod_{i=1}^{n} m_i$.

From the classical Barrington's theorem [Bar89], we know this function is NC^1-complete under projection. Formally, we have:

Lemma 4.1 ([Bar89])**.** *For any depth-d NC circuit C on n input bits, there is a projection $P : \{0,1\}^n \to \{0,1\}^{2^{O(d)}}$, 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 [Bab87, Kil88], 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 $\vec{m} = (m_1, m_2, \ldots, m_n) \in (S_5)^n$ to W_{S_5}. We draw $n+1$ i.i.d. random elements $\vec{u} = (u_1, u_2, \ldots, u_n, u_{n+1})$ from S_5, and consider the following input to W_{S_5}:
>
> $$\mathsf{Rand}(\vec{m}, \vec{u}) := (u_1 m_1 u_2^{-1}, u_2 m_2 u_3^{-1}, \ldots, u_n m_n u_{n+1}^{-1}).$$
>
> For all possible $\vec{m}$, over the randomness in $\vec{u}$, $\mathsf{Rand}(\vec{m}, \vec{u})$ distributes as a uniform random input to W_{S_5}. Moreover, we have:
>
> $$\mathsf{W}_{S_5}(\vec{m}) = u_1^{-1} \cdot \mathsf{W}_{S_5}(\mathsf{Rand}(\vec{m}, \vec{u})) \cdot u_{n+1}.$$

4.2 A Special Encoding

It may seems Lemma 4.1 and the random self-reduction are already sufficient for the collapse theorem we want, but there are still some technical problems remained.[15]

- 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; and therefore this would make it a *promise problem* only defined on a negligible fraction of inputs, which is not suited for our purpose.
- Second, a straightforward implementation of the random self-reduction actually requires NC^0 circuits, as one needs to implement product 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 don't know yet how to do circuit analysis of $\mathsf{ACC}^0 \circ \mathsf{THR} \circ \mathsf{NC}^0$ faster than brute-force.

[15] We remark similar issues arise in [GGH+07] as well.

A Special Encoding for the Second Issue. We first deal with the second issue via a special encoding of the group elements. Let $N = |S_5| = 120$. For each $i \in [N]$, let $e_i \in \{0,1\}^N$ be the vector with i-th bit being 1 while others are all zero. We identify S_5 with $[N]$ (that is, we fix a bijection between S_5 and $[N]$), 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 (W_{S_5})
>
> Given n vectors $e_{m_1}, e_{m_2}, \ldots, e_{m_n} \in \{0,1\}^N$, compute $a = \prod_{i=1}^{n} m_i$ and output e_a.

The advantage of this special encoding is that for all $p, q \in S_5$, there is a projection $P_{p,q} : \{0,1\}^N \to \{0,1\}^N$ (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 as a *projection*, and our collapse theorem doesn't introduce any additional sub-circuits at the bottom (so we can collapse NC^1 to $\mathsf{ACC}^0 \circ \mathsf{THR}$ instead of $\mathsf{ACC}^0 \circ \mathsf{THR} \circ \mathsf{NC}^0$).

Slightly abusing notation, we sometimes use $p \cdot e_a \cdot q$ to denote $e_{p \cdot a \cdot q}$.

A Redundant Encoding for the First Issue. But the first issue remains: W_{S_5} is still a promise problem, as we require all vectors to be one of the e_a's. We use a redundant encoding to make this problem defined on all possible inputs.

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

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

> Iterated group product over S_5 with a redundant encoding ($\mathsf{Redundant}\text{-}\mathsf{W}_{S_5}$)
>
> We are given n^2 $\{0,1\}^N$ vectors $\{m_{i,j}\}_{(i,j)\in[n]\times[n]}$.
> 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 N in this case.
> - Otherwise, we call the input a good input, and the goal is to compute $a = \prod_{i=1}^{n} m_{i,j_i}$ and output e_a.

4.3 NC^1 Collapses to $\mathsf{AC}^0 \circ \mathscr{C}$ if Uniform NC^1 can be Approximated by $\mathscr{C}$

We define $\mathsf{Approx}\text{-}\mathsf{MAJ}_n$ be the function that outputs 1 (resp. 0) if at least a $2/3$ fraction of the inputs are 1 (resp. 0), and is undefined otherwise. To establish our collapse theorem, we need the following standard construction for approximate-majority in AC^0.

Lemma 4.2 ([AB84, Ajt90, Vio09])**.** *$\mathsf{Approx}\text{-}\mathsf{MAJ}_n$ can be computed by* $\operatorname{poly}(n)$*-size uniform AC_3 circuits.*

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, $S : \mathbb{N} \to \mathbb{N}$ be a size parameter. There is a universal constant δ such that suppose all languages in uniform NC^1 can be $(1-\delta)$-approximated by S-size $\mathscr{C}$ circuit families. Then any depth-d NC circuit C on n input 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 NC circuit on n input. By Lemma 4.1, there is a projection $P : \{0,1\}^n \to \{0,1\}^\ell$ where $\ell = 2^{O(d)}$, such that $D(x) = \mathsf{W}_{S_5}(P(x))_{\mathsf{id}}$ (for $a \in S_5$, $(e_a)_{\mathsf{id}} = 1$ if and only if $a = \mathsf{id}$). Without loss of generality, we can assume n is sufficiently large and $d \geq \log n$.

Construction of The Circuit C Approximating Redundant-W_{S_5}. Now, let $t = \ell/120$ (that is, W_{S_5} on ℓ bits computes the iterated group product of t permutations from S_5). Consider the Redundant-W_{S_5} problem on t^2 vectors, clearly it is in uniform NC^1.

Note that Redundant-W_{S_5} has 120 output bits, so we can construct 120 $\mathscr{C}$ circuits $\{C_i\}_{i\in[N]}$, each $(1-\delta)$-approximates an output bit of Redundant-W_{S_5}. We denote $C(x) \in \{0,1\}^N$ as the vector consists of $C_i(x)$'s.

By a simple union bound, we have

$$\Pr_z\left[\text{Redundant-}\mathsf{W}_{S_5}(z) = C(z)\right] \geq 1 - \delta \cdot 120 \geq 0.75,$$

where z is a random input to Redundant-W_{S_5} from $\{0,1\}^{120\cdot t^2}$.

Implementation of the Random Self Reduction. Now, we know that for a random input to Redundant-W_{S_5}, it is a good input to Redundant-W_{S_5} with probability at least

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

when n (and therefore t) is sufficiently large.

Now we define the function $\mathsf{First} : \{0,1\}^{120\cdot t} \to \mathcal{S}_{\mathsf{good}} \cup \{\perp\}$. Given an input $\vec{m} = (m_1, m_2, \ldots, m_t) \in (\{0,1\}^N)^t$, 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 $\mathsf{First}(\vec{m}) = \perp$.

For each $m \in \mathcal{S}_{\mathsf{good}}$, we define $\mathcal{M}_m$ be the uniform distribution over the set $\{\mathsf{First}(z) = m : z \in \{0,1\}^{120\cdot t}\}$. 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 $\vec{w} = (w_1, w_2, \ldots, w_t) \leftarrow \mathcal{M}_m$ 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 $\vec{w} = (w_1, w_2, \ldots, w_t) \leftarrow \mathcal{M}_m$ 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 $\{0,1\}^N$.

One can observe that when the randomness of the above process is fixed, each bit of the sample depends on at most one bit of m (that is, it is a projection).

Next, given a valid input $\vec{m} = (m_1, m_2, \ldots, m_t)$ to W_{S_5}, we draw $t+1$ i.i.d. random elements $\vec{u} = (u_1, u_2, \ldots, u_t, u_{t+1})$ from S_5, and consider the following input to W_{S_5}:

$$\mathsf{Rand}(\vec{m}, \vec{u}) := (u_1 m_1 u_2^{-1}, u_2 m_2 u_3^{-1}, \ldots, u_t m_t u_{t+1}^{-1}).$$

Note that for all $\vec{m} \in \mathcal{S}_{\mathsf{good}}^t$, $\mathsf{Rand}(\vec{m}, \vec{u})$ distributes uniformly random on set $\mathcal{S}_{\mathsf{good}}^t$. Moreover,

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

Next, consider the following input distribution to $\mathsf{Redundant\text{-}W}_{S_5}$:

$$\mathcal{M}_{\vec{m},\vec{u}} := (\mathcal{M}_{\mathsf{Rand}(\vec{m},\vec{u})_1}, \mathcal{M}_{\mathsf{Rand}(\vec{m},\vec{u})_2}, \cdots, \mathcal{M}_{\mathsf{Rand}(\vec{m},\vec{u})_t}).$$

It is easy to see that it distributes identically to a random good input to $\mathsf{Redundant\text{-}W}_{S_5}$.

Let r be the randomness used to generate a sample from $\mathcal{M}_{\vec{m},\vec{u}}$, according to the previously discussed sampler for $\mathcal{M}_m$. Specifically, there is a set $\mathcal{R}$ and a function $\mathsf{Gen}(\vec{m}, \vec{u}, r)$, such that $\mathsf{Gen}(\vec{m}, \vec{u}, r)$ distributes identical to $\mathcal{M}_{\vec{m},\vec{u}}$ when r is drawn from $\mathcal{R}$.

Therefore, for any $\vec{m} \in \mathcal{S}_{\mathsf{good}}^t$, we have

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

Construction of the Final Circuit E. Now, one can see that $\vec{u}$ is *fixed*, $\mathsf{Rand}(\vec{m}, \vec{u})$ is a projection of $\vec{m}$. 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 $\vec{m}$.

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

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

By previous discussion, C_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

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

Therefore, we set our final circuit to be an approximate-majority of these T circuits $C_1, C_2, \ldots, C_T$. By a simple Chernoff bound, there exists 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 4.2, E is an $\mathsf{AC}_3 \circ \mathscr{C}$ circuit of size $\mathrm{poly}(S_1) = \mathrm{poly}(S(2^{O(d)}), n)$, which completes the proof. □

Remark 4.4. *We remark that the above theorem only requires that some special languages in uniform NC^1 can be approximated by $\mathscr{C}$ circuits (the languages corresponding to the output bits of $\mathsf{Redundant\text{-}W}_{S_5}$).*

5 An i.o. Non-deterministic PRG for Low-Depth Circuits

In this section we construct the required i.o. non-deterministic PRG for low-depth circuits, assuming NQP can be approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

In Section 5.1 we recall the witness-size lower bound for ACC^0 [Wil16], and observe that the proof generalizes to $\mathsf{ACC}^0 \circ \mathsf{THR}$. Then in Section 5.2, we construct the required conditional i.o. NPRG.

5.1 Witness-Size Lower Bound for NE

The following lemma is implicit in [Wil16] (with the new PCP construction of [BSV14] and the SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits from [Wil14a]) (see also Section 3 of [COS18]).

Lemma 5.1 (Essentially Theorem 9 of [COS18], combing with the algorithm in [Wil14a])**.** *For all constants $a, d_\star, m_\star$, there is an integer b and a polynomial-time verifier $V(x,y)$ with $|x| = \log^b n$, $|y| = 2^{\log^b n}$, such that for an infinite number of n's, $V(1^{\log^b n}, \cdot)$ is satisfiable, and $V(1^{\log^b n}, y) = 1$ implies y cannot be computed by a $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuit.*

Remark 5.2. *We remark that this is the only part of our argument where special properties (the existence of non-trivial circuit-analysis algorithms) of $\mathsf{ACC}^0 \circ \mathsf{THR}$ is exploited: for a typical circuit class $\mathscr{C}$, the proof of the above lemma only requires a non-trivial algorithm for Gap-UNSAT of $\mathsf{AC}^0 \circ \mathscr{C}$.*

5.2 The PRG Construction

Now we show that under the assumption that uniform NC^1 can be $(1-\delta)$-approximated by $\mathsf{ACC}^0 \circ \mathsf{THR}$, we have an i.o. NPRG for low-depth circuits.

Theorem 5.3. *(Conditional i.o. NPRG for Low-Depth Circuits) There is a universal constant δ such that for all constants $a, d_\star, m_\star$, there is an integer b such that if uniform NC^1 can be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuit families, then for an infinite number of n's, there is a non-deterministic PRG which works as follows:*

- *Let $\ell = \log^b n$, there is a polynomial time algorithm $V(x,y)$ with $|x| = \ell$ and $|y| = 2^\ell$, computable in $2^{O(\ell)}$ time.*
- *$V(1^\ell, \cdot)$ is satisfiable, and the PRG guesses a y such that $V(1^\ell, y) = 1$.*
- *The PRG then computes a function $G_y : \{0,1\}^{O(\ell)} \to \{0,1\}^{2^{\log^a n}}$, which $1/2^{\log^a n}$ fools all $\log^a n$ depth NC circuits. Moreover, G_y is computable in $2^{O(\ell)}$ time.*

Proof. Let δ be the universal constant in Theorem 4.3. We can without of loss generality assume that n is a sufficiently large integer.

Construction of the "Hardness Certifier" V' for Low-Depth Circuits. We first combine the collapse theorem with the witness-size lower bound to construct a hardness certifier V'.

By Theorem 4.3 and our assumption, we know that for a depth-d NC circuit on n bits, there is an equivalent $2^{c_e \cdot d^a}$-size $\mathsf{AC}_{d_\star + c_d}[m_\star] \circ \mathsf{THR}$ circuit for universal constants c_e and c_d.

Let a_1 be an integer to be specified later, and $d_1 = d_\star + c_d$. Now we apply Lemma 5.1 with parameters $a_1, d_1, m_\star$. Then there is another constant $b_1 = b_1(a_1, d_1, m_\star)$ such that there is a polynomial-time algorithm $V'(x,y)$ with $|x| = \log^{b_1} n$, $|y| = 2^{\log^{b_1} n}$, such that for infinite n's, we have $V'(1^{\log^{b_1} n}, \cdot)$ is satisfiable, and $V'(1^{\log^{b_1} n}, y) = 1$ implies y cannot be computed by a $2^{\log^{a_1} n}$-size $\mathsf{AC}_{d_1}[m_\star] \circ \mathsf{THR}$ circuit.

Let $d = \log^k n$ for a constant k to be specified later. A depth-d NC circuit has an equivalent $2^{c_e \log^{ak} n}$-size $\mathsf{AC}_{d_1}[m_\star] \circ \mathsf{THR}$ circuit. Now, we set $a_1 = ak + 1$ (hence $\log^{a_1} n > c_e \log^{ak} n$) so that on these infinite n's, for a y of length $2^{\log^{b_1} n}$ with $V'(1^{\log^{b_1} n}, y) = 1$, we know that y cannot be computed by a $\log^k n$-depth NC circuits.

Construction of the NPRG. Now we can plug this y into a standard construction of a PRG. Let c_2, g and $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be the constants and the function in Theorem 2.1. Now, on these infinite n's, we guess a y such that $V'(1^{\log^{b_1} n}, y) = 1$, and computes the corresponding PRG G_y.

By Theorem 2.1, the PRG $G_y : \{0,1\}^{\log^{g \cdot b_1} n} \to \{0,1\}^{S'}$, where $S' = 2^{\log^{c_2 k} n}$, $1/S'$-fools $\log S'$-depth NC circuit, and is computable in $\mathrm{poly}(|y|) \le 2^{O(\log^{b_1} n)}$ time. Now we can set k so that $\log S' = \log^{c_2 k} n \ge \log^a n$ (that is, $k = 2a/c_2$) and $b = g \cdot b_1$, which completes the proof (the final verifier V takes x, y with $|x| = \ell = \log^b n$ and $|y| = 2^\ell$, and simulates V' with $x = 1^{\log^{b_1} n}$ and the first $2^{\log^{b_1}}$ bits of y). $\square$

Remark 5.4. *The guarantee on the above algorithm is that on an infinite number of n's. The algorithm computes a PRG G_y with all y such that $V(1^{\log^b n}, y) = 1$. That is, on different such valid y's, it could compute different PRG G_y's.*

6 Average-Case "Almost" Almost Everywhere Lower Bounds for MA

In this section we prove the average-case circuit lower bounds for MA (in fact, $\mathsf{MA} \cap \mathsf{coMA}$), which is the most important technical component of our proof. In Section 6.1 we introduce some definitions and lemmas which will be helpful for our proof. In Section 6.2, we prove an average-case $\mathsf{MA} \cap \mathsf{coMA}$ a.a.e. lower bound for general circuits. In Section 6.3, we generalize it to an average-case $\mathsf{MA} \cap \mathsf{coMA}$ a.a.e. lower bound for low-depth circuits, and with a low-depth computable predicate.

6.1 Preliminaries

We first prove some folklore lemmas and introduce some notations. The following lemma is a direct corollary of Theorem 2.6.

Lemma 6.1. *For all constants a, there is an integer $h = h(a)$ and a language L^{diag} in $\mathsf{SPACE}(2^{\log^h n})$ such that for all sufficiently large n, $\mathsf{heur}_{1/2+1/n}\text{-}\mathsf{SIZE}(L_n^{\mathsf{diag}}) > 2^{\log^a n}$.*

The following is a simple corollary of the above lemma.

Corollary 6.2. *For all constants a, there is an integer $h = h(a)$ and a language L^{diag} in $\mathsf{SPACE}(2^{\log^h n})$ such that for all sufficiently large n, $\mathsf{heur}_{1/2+1/n}\text{-}\mathsf{DEPTH}(L_n^{\mathsf{diag}}) > \log^a n$.*

Before proving our lower bound, we first introduce a convenient definition of an $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T(n)]$ algorithm, which simplifies the presentation.

Definition 6.3. Let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A language L is in $(\mathsf{MA} \cap \mathsf{coMA})\mathsf{TIME}[T(n)]$, if there is a deterministic algorithm $A(x, y, z)$ (which is called the predicate) such that:

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

$$\Pr_z[A(x, y, z) = L(x)] \ge 2/3.$$

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

Remark 6.4. *($\mathsf{MA} \cap \mathsf{coMA}$) 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*. While 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.

6.2 An Average-Case $\mathsf{MA} \cap \mathsf{coMA}$ a.a.e. Lower Bound for General Circuits

Now we are ready to prove our average-case lower bound for $\mathsf{MA} \cap \mathsf{coMA}$, which is "almost" almost-everywhere. We first state a simpler version of our result with $O(\log n)$ advice bits. This is an average-case strengthening of the worst-case MA "almost" almost everywhere lower bound in [MW18].

Theorem 6.5. *For all constants a, there are integers b and c, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\,\mathsf{TIME}(2^{O(\log^b n)})_{/O(\log n)}$, such that for all sufficiently large $n \in \mathbb{N}$ and $m = \lceil 2^{\log^c n} \rceil$, either*

- $\mathsf{heur}_{1/2+1/n}\text{-}\mathsf{SIZE}(L_n) > 2^{\log^a n}$*, or*
- $\mathsf{heur}_{1/2+1/m}\text{-}\mathsf{SIZE}(L_m) > 2^{\log^a m}$*.*

Remark 6.6. *This "almost almost-everywhere" condition states that, in a precise sense, L is hard on at least "half" of the input lengths.*

Proof. Let L^{PSPACE} be the language specified by Theorem 2.5. By Lemma 6.1 with parameter a, there is a constant h and a language $L^{\mathsf{diag}} \in \mathsf{SPACE}(2^{\log^h n})$ such that $\mathsf{heur}_{1/2+1/n}\text{-}\mathsf{SIZE}(L^{\mathsf{diag}}_n) > 2^{\log^a n}$ for all sufficiently large n. Since L^{PSPACE} is PSPACE-complete, there is a constant c_1 such that L^{diag}_n can be reduced to L^{PSPACE} on input length $2^{\log^{c_1} n}$ in $2^{O(\log^{c_1} n)}$ time. We set $c \ge c_1$.

The Algorithm. Given an input x of length n and let $m = \lceil 2^{\log^c n} \rceil$, we first provide an informal description of the $\mathsf{MA} \cap \mathsf{coMA}$ algorithm which computes the language L. There are two cases:

1. When $\mathsf{SIZE}(L^{\mathsf{PSPACE}}_m) \le 2^{\log^b n}$. That is, when L^{PSPACE}_m is *easy*. In this case, we guess-and-verify a circuit for L^{PSPACE}_m of size $2^{\log^b n}$, and use that to compute L^{diag}_n.
2. Otherwise, we know L^{PSPACE}_m is *hard*. On input of length m, we are given an advice y which is the largest integer such that $L^{\mathsf{PSPACE}}_y \le 2^{\log^b n}$. We guess-and-verify a circuit for L^{PSPACE}_y, and compute it (that is, compute L^{PSPACE}_y on the first y input bits while ignoring the rest).

Intuitively, the above algorithm computes an average-case hard function because either it computes the average-case hard language L^{diag}_n on inputs of length n, or it computes the average-case hard language L^{PSPACE}_y on inputs of length m (L^{PSPACE} is robust). A formal description of the algorithm is given in Algorithm 1, while the algorithm for setting the advice bits is given in Algorithm 2 (note that a y_n may be set twice).

Algorithm 1: The $\mathsf{MA} \cap \mathsf{coMA}$ algorithm for the average-case hard language L

```
Given an input x with length n = |x|;
Given an advice integer y = y_n ∈ [−1, n] ∩ ℤ;
Let m = ⌈2^{log^c n}⌉;
Let n_0 = n_0(n) be the integer such that ⌈2^{log^c n_0}⌉ = n; if no such integer exists, n_0 = −1;
if y = −1 then
    Output 0 and terminate
if y = 0 then
    (y = 0 indicates we are in the case that SIZE(L_m^PSPACE) ≤ 2^{log^b n}.);
    Compute a z of length m in 2^{O(log^c n)} time such that L_n^diag(x) = L_m^PSPACE(z);
    Guess a circuit C of 2^{log^b n} size;
    Let M be the instance checker for L_m^PSPACE;
    Flip an appropriate number of random coins, let them be r;
    Output M^C(z, r);
else
    (y > 0 indicates we are in the case that SIZE(L_n^PSPACE) > 2^{log^b n_0}.);
    Let z be the first y bits of x;
    Guess a circuit C of 2^{log^b n_0} size;
    Let M be the instance checker for L_y^PSPACE;
    Flip an appropriate number of random coins, let them be r;
    Output M^C(z, r);
```

Algorithm 2: The algorithm for setting advice bits of Algorithm 1

```
All y_n's are set to −1 by default;
for n = 1 → ∞ do
    Let m = ⌈2^{log^c n}⌉;
    if SIZE(L_m^PSPACE) ≤ 2^{log^b n} then
        Set y_n = 0;
    else
        Set y_m = max{y : SIZE(L_y^PSPACE) ≤ 2^{log^b n}};
```

The Algorithm Satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ Promise. We first show the algorithm satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise (Definition 6.3). 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 2.2) ensure that in this case the algorithm satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise. Let $y = y_n$, there are three cases:

1. $y = -1$. In this case, the algorithm computes the all zero function, and clearly satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise.

2. $y = 0$. In this case, from Algorithm 2, we know that $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \le 2^{\log^b n}$ for $m = \lceil 2^{\log^c n} \rceil$. Therefore, at least one guess of the circuit is a correct circuit for L_m^{PSPACE}, and on that guess, the algorithm outputs $L_n^{\mathsf{diag}}(x) = L_m^{\mathsf{PSPACE}}(z)$ with probability at least $2/3$, by the property of the instance checker (Definition 2.2).

   Still by the property of the instance checker, on all possible guesses, the algorithm outputs $1 - L_n^{\mathsf{diag}}(x) = 1 - L_m^{\mathsf{PSPACE}}(z)$ with probability at most $1/3$. Hence, the algorithm correctly computes L_n^{diag} on inputs of length n, with respect to Definition 6.3.

3. $y > 0$. In this case, from Algorithm 2, we know that $n_0 \neq -1$, $n = \left\lceil 2^{\log^b n_0} \right\rceil$, $\mathsf{SIZE}(L_n^{\mathsf{PSPACE}}) > 2^{\log^b n_0}$, and $\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \le 2^{\log^b n_0}$. Therefore, at least one guess of the circuit is a correct circuit for L_y^{PSPACE}, and on that guess, the algorithm outputs $L_y^{\mathsf{PSPACE}}(z)$ ($z = z(x)$ is the first y bits of x) with probability at least $2/3$, by the property of the instance checker (Definition 2.2).

   Still by the property of the instance checker, on all possible guesses, the algorithm outputs $1 - L_y^{\mathsf{PSPACE}}(z)$ with probability at most $1/3$. Hence, the algorithm correctly computes $L_y^{\mathsf{PSPACE}}(z(x))$ on inputs of length n, with respect to Definition 6.3.

The Algorithm Computes an "Almost" Almost Everywhere Average-Case Hard Language. Next we show that the algorithm indeed computes an average-case hard language. Let n be a sufficiently large integer and $m = \lceil 2^{\log^c n} \rceil$. According to Algorithm 2, there are two cases.

- $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) \le 2^{\log^b n}$. In this case, Algorithm 2 sets $y_n = 0$. And by previous analysis, we know that L_n computes the average-case hard language L_n^{diag}, and therefore $\mathsf{heur}_{1/2+1/n}\text{-}\mathsf{SIZE}(L_n) > 2^{\log^a n}$ as n is sufficiently large.

- $\mathsf{SIZE}(L_m^{\mathsf{PSPACE}}) > 2^{\log^b n}$. We set b so that $2^{\log^b n} \ge 2^{\log^{2a}(m)}$ (we can set $b \ge 3ac$). Let y be the largest integer such that $\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \le 2^{\log^b n}$. By Remark 2.3, we have $y < m$.

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

  $$\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \ge \mathsf{SIZE}(L_{y+1}^{\mathsf{PSPACE}}) / \left\lceil 2^{\log^c n} \right\rceil^d \ge 2^{\Omega(\log^b n)}.$$

  Now, on an input of length m, clearly we have $n_0(m) = n \neq -1$ and $y_m \neq -1$ by Algorithm 2. Therefore, L_m either computes L_m^{diag} or $L_{y_m}^{\mathsf{PSPACE}}$ (since $y_m \neq -1$). The first case is already discussed. In the second case, we know $y_m = y$ and $\mathsf{heur}_{1/2+1/m}\text{-}\mathsf{SIZE}(L_m) = \mathsf{heur}_{1/2+1/m}\text{-}\mathsf{SIZE}(L_y^{\mathsf{PSPACE}})$.

Now, since $\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \le 2^y$, we have $y \ge \Omega(\log^b n)$. Let c_1 and δ_1 be the corresponding constants of the robust property of L^{PSPACE}. For $\varepsilon \ge 2^{-y^{\delta_1}}$, we have

$$\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \le (\mathsf{heur}_{1/2+\varepsilon}\text{-}\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \cdot \varepsilon^{-1})^{c_1},$$

and hence

$$\mathsf{heur}_{1/2+\varepsilon}\text{-}\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \ge \varepsilon \cdot \mathsf{SIZE}(L_y^{\mathsf{PSPACE}})^{1/c_1} \ge \varepsilon \cdot 2^{\Omega(\log^b n)}.$$

We set b so that $y^{\delta_1} \ge \Omega\left(\log^{\delta_1 \cdot b} n\right) \ge \log(m)$ (that is, we can set $b \ge 2c/\delta_1$), and then set $\varepsilon = 1/m$. It follows that $\mathsf{heur}_{1/2+1/m}\text{-}\mathsf{SIZE}(L_y^{\mathsf{PSPACE}}) \ge 2^{\Omega(\log^b n)}/2^{\log^c n} \ge 2^{\Omega(\log^b n)} \ge 2^{\log^a(m)}$, which completes the whole proof.

□

6.3 An Average-Case $\mathsf{MA} \cap \mathsf{coMA}$ a.a.e. Lower Bound for Low Depth Circuits

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

By significantly relaxing the "almost" almost everywhere requirement, we are able to construct an average-case hard language with only one bit of advice, yet still enough for our final average-case circuit lower bound proof.

Theorem 6.7. *For all constants a, there are integers b and c, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\,\mathsf{TIME}(2^{O(\log^b n)})_{/1}$ (specified by Algorithm 3 and Algorithm 4), such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*

- $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_n) > \log^a n$*, or*
- $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_m) > \log^a m$*, for an* $m \in (2^{\log^c n}, 2^{\log^c n+1}) \cap \mathbb{N}$.

Remark 6.8. *We remark that in the real proof, we slightly deviate from the intuition section of the introduction: we actually don't need the precise condition that the corresponding predicate is low-depth computable as it is not required by the proof (the proof only requires that the instance checker part (the composed circuit $D^C_{\mathsf{checker}}(z,\cdot)$) is computable by low-depth circuits). Still, it is not hard to make the entire predicate corresponding to Algorithm 3 low-depth computable.*

Proof of Theorem 6.7. Let L^{PSPACE} be the language specified by Theorem 2.5. By Corollary 6.2 with parameter a, there is a language $L^{\mathsf{diag}} \in \mathsf{SPACE}(2^{\log^h n})$ for a constant h such that $\mathsf{heur}_{1/2+1/n}$ $\text{-}\mathsf{DEPTH}(L_n^{\mathsf{diag}}) > \log^a n$ for all sufficiently large n. Since L^{PSPACE} is PSPACE-complete, there is a constant c_1 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 $c \ge c_1$, and recall that $\mathsf{heur}_{1/2+1/n}\text{-}\mathsf{DEPTH}(L_n^{\mathsf{diag}}) > \log^a n$, and therefore $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_n^{\mathsf{diag}}) > \log^a n$.

Algorithm 3: The $\mathsf{MA} \cap \mathsf{coMA}$ algorithm for the language L which is average-case hard for low-depth circuits

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

Algorithm 4: The algorithm for setting advice bits for Algorithm 3

All y_n's are set to 0 by default;
for $\tau = 1 \to \infty$ **do**
    Let $n = 2^\tau$;
    Let $m = 2^{\log^c n}$;
    if *$DEPTH$*$(L_m^{\mathsf{PSPACE}}) \leq \log^b n$ **then**
        Set $y_n = 1$;
    else
        Let $\ell = \max\{\ell : \mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \leq \log^b n\}$;
        Set $y_{m+\ell} = 1$;

The Algorithm. Let $\tau \in \mathbb{N}$ be sufficiently large. Given an input x of length $n = 2^\tau$ and let $m = 2^{\log^c n}$, we first provide an informal description of the $\mathsf{MA} \cap \mathsf{coMA}$ algorithm which computes the language L. There are two cases:

1. When $\mathsf{DEPTH}(L_m^{\mathsf{PSPACE}}) \le \log^b n$. That is, when L_m^{PSPACE} is *easy*. In this case, 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 L_m^{PSPACE} is *hard*. Let ℓ be the largest integer such that $\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \le \log^b n$. On input of length $m_1 = m+\ell$, we guess-and-verify a circuit for L_ℓ^{PSPACE}, and compute it (that is, compute L_ℓ^{PSPACE} on the first ℓ input bits while ignoring the rest). Note that by Remark 2.3, we have $0 < \ell < m$ and therefore $m + \ell$ is not a power of 2.

Intuitively, the above algorithm 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 the algorithm is given in Algorithm 3, while the algorithm for setting the advice bits is given in Algorithm 4. It is not hard to see that a y_n can only be set once in Algorithm 4.

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

The Algorithm Satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ Promise. Again, by Algorithm 4, the algorithm tries to guess a circuit for L^{PSPACE} only if that circuit exists. Therefore, by a similar argument as in the proof of Theorem 6.5, the algorithm satisfies the $\mathsf{MA} \cap \mathsf{coMA}$ promise. Moreover, L_n computes L_n^{diag} if $y_n = 1$ and n is a power of 2, and L_ℓ^{PSPACE} if $y_n = 1$ and n is not a power of 2.

The Algorithm Computes an "Almost" Almost Everywhere Average-Case Hard Language for Low Depth Circuits. Next we show that the algorithm 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 4, there are two cases:

- $\mathsf{DEPTH}(L_m^{\mathsf{PSPACE}}) \le \log^b n$. In this case, Algorithm 4 sets $y_n = 1$. And by previous analysis, we know that L_n computes the average-case hard language L_n^{diag}, and therefore $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_n) > \log^a n$ as n is sufficiently large.
- $\mathsf{DEPTH}(L_m^{\mathsf{PSPACE}}) > \log^b n$. We set b so that $\log^b n \ge \log^{2a}(2m)$ (we can set $b \ge 3ac$). Let ℓ be the largest integer such that $\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \le \log^b n$. By Remark 2.3, 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 (because L^{PSPACE} is TC^0 downward self-reducible, and the corresponding TC^0 oracle circuit is *non-adaptive*). Therefore,

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

  Now, on inputs of length $m_1 = m + \ell$, we have $y_{m_1} = 1$ by Algorithm 4. Therefore, L_{m_1} computes L_ℓ^{PSPACE}, and therefore $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_{m_1}) = \mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}})$.

  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

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

Therefore, by our choice of b, it follows

$$\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \geq \mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) - d\log^3 \ell \geq \Omega(\log^b n) - O(\log^{3c} n) \geq \Omega(\log^b n).$$

Finally, note that $\Omega(\log^b n) \geq \Omega(\log^{2a}(2m)) \geq \log^a(m_1)$. We have $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_{m_1}) = \mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_\ell^{\mathsf{PSPACE}}) \geq \log^a(m_1)$, which completes the proof.

□

Finally, using a similar trick as in the proof of Theorem 6.7, we can also reduce the number of advice in Theorem 6.5 to 2 bits.

Corollary 6.9. *For all constants a, there are integers b and c, and a language $L \in (\mathsf{MA} \cap \mathsf{coMA})\,\mathsf{TIME}(2^{O(\log^b n)})_{/1}$, such that for all sufficiently large $\tau \in \mathbb{N}$ and $n = 2^\tau$, either*

- *$\mathsf{heur}_{0.99}\text{-}\mathsf{SIZE}(L_n) > 2^{\log^a n}$, or*
- *$\mathsf{heur}_{0.99}\text{-}\mathsf{SIZE}(L_m) > 2^{\log^a m}$, for an $m \in (2^{\log^c n}, 2^{\log^c n+1}) \cap \mathbb{N}$.*

7 A PSPACE-complete Language with Nice Reducibility Properties

In this section we construct a PSPACE-complete language with the needed nice properties.

In Section 7.1 we introduce the necessary definitions for the construction of this section. In Section 7.2 we review the original construction in [TV07]; and in Section 7.3 we briefly discuss what adaption is required to make it suitable for our purpose. In Section 7.4 we construct the needed PSPACE-complete language.

7.1 Notations and Boolean Encodings of Field Elements

We first need to introduce some notations. Let $\mathsf{pow}(n)$ be the smallest power of 2 which is no less than n.

Let $\mathbb{F}_n$ be $\mathsf{GF}(2^{\mathsf{pow}(n)})$. Note that for all $n < m$, either $\mathbb{F}_n = \mathbb{F}_m$, or $\mathbb{F}_n$ is a sub-field of $\mathbb{F}_m$. An element from $\mathbb{F}_n$ can be encoded in $\mathsf{pow}(n)$ bits via a natural bijection ϕ_n between $\mathbb{F}_n$ and $\mathsf{GF}(2)^{\mathsf{pow}(n)}$. We encode them in a consistent way that for any $2^\ell < \mathsf{pow}(n)$, the first 2^ℓ bits of the encoding correspond to an element from $\mathsf{GF}(2^\ell)$.

That is, for all $n < m$ and an element a from $\mathbb{F}_n$, the first $\mathsf{pow}(n)$ bits of $\phi_m(a)$ equals $\phi_n(a)$. Note that all these fields $\mathbb{F}_n$ (i.e., a degree $\mathsf{pow}(n)$ irreducible $\mathsf{GF}(2)$-polynomial) and bijections ϕ_n can be constructed deterministically in $\mathrm{poly}(n)$ time [Sho88].

7.2 Review of the Construction in [TV07]

We need the following lemma from [TV07], which builds on the proof of $\mathsf{IP} = \mathsf{PSPACE}$ theorem [LFKN92, Sha92].

Lemma 7.1. *For some polynomials t and m, there is a collection of functions $\{f_{n,i} : (\mathbb{F}_n)^{t(n,i)} \to \mathbb{F}_n\}_{n\in\mathbb{N}, i\leq m(n)}$ with the following properties:*

1. *(Self-Reducibility) For $i < m(n)$, $f_{n,i}$ can be evaluated with oracle access to $f_{n,i+1}$ in $\mathrm{poly}(n)$ time. $f_{n,m(n)}$ can be evaluated in $\mathrm{poly}(n)$ time, and in fact it is computable by a $\mathrm{poly}(n)$-size uniform TC^0 circuit.*

2. *(PSPACE-hardness) For every language L in PSPACE, there is a polynomial-time computable function ℓ and g, such that for all $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, $L(x) = f_{\ell(1^n),0}(g(x))$, and $\ell(1^n)$ is bounded by a polynomial in n (which depends on L).*

3. *(Low Degree) $f_{n,i}$ is a polynomial of total degree at most $\mathrm{poly}(n)$.*

Remark 7.2. *In [TV07], the field $\mathbb{F}_n$ is just $\mathsf{GF}(2^n)$, we make it slightly large in order to establish the padability.[16] We formulate the second property in a slightly different way than [TV07] for convenience. Also, it is easy to see that in the construction of [TV07], $t(n,i)$ and $m(n)$ are both increasing functions in n.*

The polynomial $f_{n,m(n)}$ in [TV07] is very simple, and it is easy to see that it can be computed by a $\mathrm{poly}(n)$-size uniform TC^0 circuit.

More specifically, for all $i < m(n)$, $f_{n,i}(x)$ has $\ell = t(n,i)$ variables, and it is defined in terms of $f_{n,i+1}$ using one of the following rules:

Three definitions rules of $f_{n,i}(x)$

$$f_{n,i}(x_1,\ldots,x_\ell) = f_{n,i+1}(x_1,\ldots,x_\ell,0)\cdot f_{n,i+1}(x_1,\ldots,x_\ell,1). \tag{1}$$

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

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

7.3 Technical Challenges to Adapt [TV07] for Our Purpose

The original language in [TV07] just computes $f_{n,i}$ in the order of first increasing in n and then decreasing in i. By Lemma 7.1, this can be easily seen to be downward self-reducible and error correctable (as polynomials are error correctable). To make it further paddable, [FS04, San09] simply use a padding construction so that now on a single input length, the language computes $f_{n,i}$ and all polynomials come before it.

In order to construct a PSPACE-complete language which is both error correctable and paddable, there are some technical challenges:

- First, after the padding construction, the language now is not a single polynomial, but a bunch of different polynomials. We need to do some interpolation to "wrap" them 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 *sub-field* of $\mathbb{F}_m$), and adding dummy variables.

[16] The problem with the original encoding is, $\mathsf{GF}(2^n)$ is not a sub-field of $\mathsf{GF}(2^{n+1})$ for $n \geq 2$.

- Another problem is that a simple interpolation would actually destroy the padability. Suppose we have k polynomials $g_1, g_2, \ldots, g_k : \mathbb{F}^n \to \mathbb{F}$ of degree D. We can construct a single polynomial $G_k : \mathbb{F}^{n+1} \to \mathbb{F}$ with degree $D + k$, such that $G_k(i, x) = g_i(x)$, via a simple interpolation. But the issue here is that then G_{k-1} cannot be reduced to G_k easily (so it is not paddable). We resolve this via a different choice of interpolation, specifically, we define $G_k : \mathbb{F}^n \times \mathbb{F}^k \to \mathbb{F}$ as $G_k(x, y_1, y_2, \ldots, y_k) := \sum_{i=1}^{k} g_i(x) \cdot y_i$.

- Finally, the polynomials are over a large alphabet $\mathbb{F}_n$, and we have to turn them into Boolean functions. This step is standard as one can just make use of Walsh-Hadamard codes.

The next step is to argue that the reducibility properties of the constructed new language actually have low complexity oracle circuits implementations. For padability it is trivial; for downward self-reducibility it is evident from the way that these $f_{n,i}$'s are defined; and for weakly error correctability, it is still straightforward from the local decoders of Reed-Muller codes and Walsh-Hadamard codes. The main difficulty here is to argue this for same-length checkability.

- This actually looks counter-intuitive at first—the instance-checker in [TV07, FS04, San09] actually simulates the interactive proof protocol for PSPACE [LFKN92, Sha92]. Since it is an interactive proof protocol, it appears that this checking process should proceed one step after another step (that is, highly sequentially), and it should not have a highly parallelizable implementation such as TC^0 oracle circuits.

- The key observation is that, despite the fact that we are simulating an interactive proof protocol, the prover's strategy *is already committed to the given oracle*. 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.

7.4 The Construction of the PSPACE-complete Language

Now we are ready to construct the needed PSPACE-complete language, we first restate the theorem for convenience.

Reminder of Theorem 2.5 *There is a PSPACE-complete language L^{PSPACE} which is paddable, TC^0 downward self-reducible, TC^0 same-length checkable, robust and NC^3 weakly error correctable. Moreover, all the corresponding oracle circuits for the above properties are in fact* non-adaptive*: that is, on any path from an input gate to the output gate, there is at most one oracle gate.*

Proof of Theorem 2.5. In the following, we roughly follows the ideas outlined in Section 7.3. Our construction is a careful modification of the construction from [TV07], together with an application of Walsh-Hadamard codes to turn the polynomials into Boolean functions.

Construction of Interpolated Polynomial G_k. First, we order all polynomials in the following order

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

Let g_k be the k-th polynomial in the above list. Suppose g_k is $f_{n,i}$. Let $d = d(k)$ be the maximum number of variables of a polynomial in $g_1, g_2, \ldots, g_k$. By introducing some dummy variables at the end, we can make all polynomials $g_1, g_2, \ldots, g_k$ have d variables. Moreover, since all fields $\mathbb{F}_1, \mathbb{F}_2, \ldots, \mathbb{F}_{n-1}$ are sub-fields of $\mathbb{F}_n$ (or equal to $\mathbb{F}_n$), we can treat all $g_1, g_2, \ldots, g_k$ as polynomials from $\mathbb{F}_n^d \to \mathbb{F}_n$.

Now, we introduce k more variables $y_1, y_2, \ldots, y_k$, and define the following polynomial $G_k : \mathbb{F}_n^{d+k} \to \mathbb{F}_n$,

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

Since all g_i's are of total degree at most $\mathrm{poly}(n)$, G_k is also of total degree $\mathrm{poly}(n)$.

Converting G_k into a Boolean Function via Walsh-Hadamard Codes. Next, we need to turn the polynomial G_k into a Boolean function. We do this by applying Walsh-Hadamard codes.

Let $\ell = \mathsf{pow}(n)$. We use the bijection $\phi = \phi_n$ between $\mathbb{F}_n$ and $\mathsf{GF}(2)^\ell$ described in Section 7.1.

We define $F_k : \mathbb{F}_n^{d+k} \times \mathsf{GF}(2)^\ell \to \mathsf{GF}(2)$ as,

$$F_k(z, r) := \langle \phi(G_k(z)), r \rangle,$$

where $\langle \phi(G_k(z)), r \rangle$ is the inner product between $\phi(G_k(z))$ and r over $\mathsf{GF}(2)$.

F_k can be easily interpreted as a Boolean function from $\{0,1\}^{e(k)} \to \{0,1\}$, where $e(k) = (d+k+1) \cdot \ell$.

Now, for each input length m, let k be the largest integer such that $e(k) \le m$, and we set L_m^{PSPACE} to compute F_k on its first $e(k)$ bits (if there is no such k, L_m^{PSPACE} just computes the all-zero function).

In the following we verify that L^{PSPACE} have all the desired properties.

L^{PSPACE} is Paddable. Let $n < m$ be two input lengths. Suppose L_n^{PSPACE} computes function F_{k_1} and L_m^{PSPACE} computes function F_{k_2}. Clearly, $k_1 \le k_2$, and the case $k_1 = k_2$ is trivial. When $k_1 < k_2$, note that

$$G_{k_1}(x, y_1, y_2, \ldots, y_{k_1}) = \sum_{i=1}^{k_1} g_i(x) \cdot y_i = G_{k_2}(x, y_1, y_2, \ldots, y_{k_1}, 0, 0, \ldots, 0).$$

The padability is then evident with our encoding of the fields $\mathbb{F}_n$'s (see Section 7.1).

L^{PSPACE} is Robust. Supposing L_m^{PSPACE} computes the function F_k, we only need to show this property for the Boolean function F_k. By the well-known local-list-decoders of the Walsh-Hadamard codes [GL89] and of the Reed-Muller codes [STV01], this property follows directly.

L^{PSPACE} is NC^3 Weakly Error Correctable. This follows from the well-known local-decoders of the Reed-Muller codes and the Walsh-Hadamard codes [STV01]. Walsh-Hadamard codes have NC^1 local decoders [GL89], while the computational bottleneck of the local decoder of Reed-Muller is solving a system of linear equations over $\mathbb{F}_n$. Solving a system of linear equation can be done by an $O(\log^2 n)$ depth arithmetic circuit with field operations over $\mathbb{F}_n$, and a field operation over $\mathbb{F}_n$ can be implemented by a uniform TC^0 circuit [HV06] (and therefore a uniform NC^1 circuit). Hence, the whole local decoder can be implemented by a uniform NC^3 circuit, and this property follows.

L^{PSPACE} **is TC^0 Same-length Checkable.** Suppose we want to check whether $F_k(x,y,r) = 1$ given an oracle O which is supposed to compute F_k (the case for checking whether $F_k(x,y,r) = 0$ is analogous). Suppose $g_k = f_{n,i}$, and let $\ell = \mathsf{pow}(n)$. Note that given an oracle for F_k, one can ask it ℓ times to get $G_k(x,y)$ for any valid x, y.

We first query the oracle O to get $G_k(x,y)$, and reject immediately if it is not consistent with $F_k(x,y,r)$. Since $G_k(x,y) = \sum_{i=1}^{k} g_k(x) \cdot y_i$, we next ask the oracle O to get $g_1(x) = G_k(x,1,0,\dots,0), g_2(x) = G_k(x,0,1,0,\dots),\dots,g_k(x) = G_k(x,0,0,\dots,0,1)$. We reject immediately if these queried values are not consistent with $G_k(x,y)$. Now we can use the original instance checker in [TV07, FS04] to check whether these obtained $g_i(x)$'s are correct.

Therefore, now it suffices to show that the instance checker of [TV07, FS04] can be implemented by a uniform polynomial size TC^0 circuit. Suppose we want to check the value of $f_{n,i}(x)$ for some n and i, given oracle access to alleged polynomials $\widetilde{f}_{n,i}, \widetilde{f}_{n,i+1}, \dots, \widetilde{f}_{n,m(n)}$, which are supposed to compute the polynomials $f_{n,i}, f_{n,i+1}, \dots, f_{n,m(n)}$ (by the way we order polynomials, these alleged polynomials are accessible given the oracle O).

For all $i < m(n)$, $f_{n,i}(x)$ has $\ell = t(n,i)$ variables, recall that it is defined in terms of $f_{n,i+1}$ using one of the following rules:

$$f_{n,i}(x_1,\dots,x_\ell) = f_{n,i+1}(x_1,\dots,x_\ell,0) \cdot f_{n,i+1}(x_1,\dots,x_\ell,1). \tag{4}$$

$$f_{n,i}(x_1,\dots,x_\ell) = 1 - (1 - f_{n,i+1}(x_1,\dots,x_\ell,0)) \cdot (1 - f_{n,i+1}(x_1,\dots,x_\ell,1)). \tag{5}$$

$$f_{n,i}(x_1,\dots,x_k,\dots,x_\ell) = x_k \cdot f_{n,i+1}(x_1,\dots,1,\dots,x_\ell) + (1 - x_k) \cdot f_{n,i+1}(x_1,\dots,0,\dots,x_\ell). \tag{6}$$

Let $D = \text{poly}(n)$ be a degree bound on all the polynomials $f_{n,i}, f_{n,i+1}, \dots, f_{n,m(n)}$. Suppose we want to check whether $f_{n,i}(x_1,\dots,x_\ell) = T_i$, the instance checker works as follows:

- For case (1) and case (2), we first query the oracle polynomials $\widetilde{f}_{n,i+1}$ on points $(x_1,\dots,x_\ell,z)$ for $z \in \{0,1,2,\dots,D\}$, and interpolate a polynomial $P_i(z)$ of degree D, which is supposed to be the polynomial $f_{n,i+1}(x_1,\dots,x_\ell,z)$.
  - We first check whether $P_i(0) \cdot P_i(1) = T_i$ in case (1), or $1 - (1 - P_i(0)) \cdot (1 - P_i(1)) = T_i$ in case (2), and reject immediately if they are not satisfied.
  - We pick a random value $z_i \in \mathbb{F}_n$, and proceed to check whether $f_{n,i+1}(x_1,\dots,x_\ell,z_i) = P_i(z_i)$.
- For case (3), we first query the oracle polynomials $\widetilde{f}_{n,i+1}$ on points $(x_1,\dots,x_{k-1},z,x_{k+1},\dots,x_\ell)$ for $z \in \{0,1,2,\dots,D\}$, and interpolate a polynomial $P_i(z)$ of degree D, which is supposed to be the polynomial $f_{n,i+1}(x_1,\dots,x_{k-1},z,x_{k+1},\dots,x_\ell)$.
  - We first check whether $x_k \cdot P_i(1) + (1 - x_k) \cdot P_i(0) = T_i$.
  - We pick a random value $z_i \in \mathbb{F}_n$, and proceed to check whether $f_{n,i+1}(x_1,\dots,x_{k-1},z_i,x_{k+1},\dots,x_\ell) = P_i(z_i)$.
- Finally, when we reach the stage of checking whether $f_{n,m(n)}(x_1,x_2,\dots,x_{t(n,m(n))}) = T_{m(n)}$. We simply evaluate the polynomial $f_{n,m(n)}$ on the given point and reject it is not equal to $T_{m(n)}$.

The correctness of the instance checker follows directly from the proof of $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha92]. Now we show it can be implemented in uniform TC^0.

First notice that we can draw all the random values $z_i, z_{i+1}, \ldots, z_{m(n)}$ in the beginning, and each interpolated polynomials P_i are completed determined by the input $x_1, x_2, \ldots, x_\ell$, the random values z_i's, and the oracle polynomial $\widetilde{f}_{n,i}$'s. By Lagrange's formula and [HV06], all P_i's can be computed by uniform TC^0 non-adaptive oracle circuits with the oracle O.

After constructing the polynomials, one can see the instance checker only needs to perform some additional consistency checks. Note that we have $T_{i+1} = P_i(z_i)$, so all consistency checks only involve at most two polynomials P_i and P_{i+1}, and they can be easily implemented by uniform TC^0 circuits, again by [HV06].

L^{PSPACE} is TC^0 Downward Self-reducible. Finally we show how to compute L_m^{PSPACE} given an oracle to $L_{m-1}^{\mathsf{PSPACE}}$. Suppose L_m^{PSPACE} computes the function F_k. The case when $L_{m-1}^{\mathsf{PSPACE}}$ also computes F_k is trivial, so we can assume $L_{m-1}^{\mathsf{PSPACE}}$ computes F_{k-1}.

To compute $F_k(x, y, r)$, it suffices to compute $G_k(x, y)$. Computing $G_k(x, y)$ can be in turn reduced to computing $g_1(x), g_2(x), \ldots, g_k(x)$. Recall that these $g_i(x)$'s are defined by one of the rules (4), (5) and (6), we can see either $g_i(x)$ is itself computable by a uniform TC^0 circuit (it is $f_{n,m(n)}$ for some n), or it can be computed by a uniform TC^0 non-adaptive oracle circuit with g_{i-1} as the oracle [HV06].

Given oracle access to F_{k-1}, we also get the access to polynomials $g_1(x), g_2(x), \ldots, g_{k-1}(x)$, and therefore we can compute each $g_1(x), g_2(x), \ldots, g_k(x)$ with a uniform TC^0 non-adaptive oracle circuit with the oracle F_{k-1}. Combing them with another TC^0 circuit on the top, we can compute $F_k(x, y, r)$ with a uniform TC^0 non-adaptive oracle circuit with the oracle F_{k-1}, which completes the proof. $\square$

8 NQP is not $1/2 + o(1)$-approximable by Polynomial Size $\mathsf{ACC}^0 \circ$ THR Circuits

In this section we prove that NQP is not $(1/2 + 1/\operatorname{polylog}(n))$-approximable by polynomial-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

In Section 8.1 we introduce some definitions and lemmas which will be helpful for our proof. In Section 8.2, we prove a $(1-\delta)$-inapproximability result for $(\mathsf{NQP} \cap \mathsf{coNQP})_{/O(1)}$ against $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. And in Section 8.3, we apply mild to strong hardness amplification to obtain a $(1/2 + 1/\operatorname{polylog}(n))$-inapproximability result for $(\mathsf{NQP} \cap \mathsf{coNQP})_{/O(1)}$ against $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, and then apply an enumeration trick to get rid of that advice, and prove the same lower bound for NQP.

8.1 Preliminaries

We first introduce some definitions. 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 bits 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 $\mathsf{pair}(m,n)$, one can easily decode the pair of number m and n.

The following is a convenient definition for an $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$ algorithm, which simplifies the presentation.

Definition 8.1. Let $T:\mathbb{N}\to\mathbb{N}$ be a time-constructible function. A language L is in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$, if there is an algorithm $A(x,y)$ (which is called the predicate) such that:

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

Remark 8.2. *$(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[T(n)]$ languages with advice are defined similarly, with A being an algorithm with the corresponding advice.*

Similar to the case of $(\mathsf{MA}\cap\mathsf{coMA})_{/1}$ and $\mathsf{MA}_{/1}\cap\mathsf{coMA}_{/1}$, the semantic of $(\mathsf{NP}\cap\mathsf{coNP})_{/1}$ is also different from $\mathsf{NP}_{/1}\cap\mathsf{coNP}_{/1}$, as it requires the $\mathsf{NP}_{/1}$ and the $\mathsf{coNP}_{/1}$ algorithms use the same advice sequence.

8.2 $(1-\delta)$ Average-Case Lower Bounds

We first show that there is a function in $(\mathsf{NQP}\cap\mathsf{coNQP})_{/2}$ which is not $(1-\delta)$-approximable by $\mathsf{ACC}^0\circ\mathsf{THR}$ circuits, for a universal constant δ.

Theorem 8.3. *For all constants a, there is an integer b, a universal constant $\delta>0$, such that $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ is not $(1-\delta)$-approximable by $2^{\log^a n}$ size $\mathsf{ACC}^0\circ\mathsf{THR}$ circuits.*

Remark 8.4. *In other words, the conclusion of the above theorem is equivalent to that there is a language L in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ which is not $(1-\delta)$-approximable by $2^{\log^a n}$ size $\mathsf{AC}_{d_\star}[m_\star]\circ\mathsf{THR}$ circuits, for all constants $d_\star, m_\star$.*

We will prove a weaker theorem first, and then show it implies Theorem 8.3.

Theorem 8.5. *For all constants $a, d_\star, m_\star$, there is an integer b, 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 is not $(1-\delta)$-approximable by $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star]\circ\mathsf{THR}$ circuits.*

Proof. Let b be an integer to be specified later and δ be the universal constant in Theorem 5.3. Now for the sake of contradiction, suppose all languages in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ have a $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star]\circ\mathsf{THR}$ circuit family which computes it correctly on a $1-\delta$ fraction of inputs for all sufficiently large input length n.

We first apply Theorem 6.7. Let b_1 and c_1 be such that there is a language $L^{\mathsf{hard}}\in(\mathsf{MA}\cap\mathsf{coMA})\mathsf{TIME}(2^{\log^{b_1} n})_{/2}$ specified by Algorithm 3 and Algorithm 4, such that for all sufficiently large $\tau\in\mathbb{N}$ and $n=2^\tau$, either

- $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_n) > \log^{2a} n$, or
- $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_m) > \log^{2a} m$, for an $m \in (2^{\log^{c_1} n}, 2^{\log^{c_1} n+1}) \cap \mathbb{N}$.

Now we try to derandomize L^{hard} non-deterministically, and get a contradiction. In the following we always assume n is sufficiently large.

By Theorem 5.3, there is a constant b_2, such that the following holds for an infinite number of n's (we call them good n's):

- Let $S_{\mathsf{derand}}(n) = 2^{\log^{2b_1 c_1^2} n}$.
- There is a polynomial time algorithm $V(x, y)$ with $|x| = \log^{b_2} n$ and $|y| = 2^{\log^{b_2} n}$ computable in $2^{O(\log^{b_2} n)}$ time.
- $V(1^{\log^{b_2} n}, \cdot)$ is satisfiable, and for all y such that $V(1^{\log^{b_2} n}, y) = 1$, $G_y : \{0,1\}^{O(\log^{b_2} n)} \to \{0,1\}^{S_{\mathsf{derand}}(n)}$ is a PRG which $1/S_{\mathsf{derand}}(n)$ fools all $\log S_{\mathsf{derand}}(n)$ depth NC circuits, and computable in $2^{O(\log^{b_2} n)}$ time.

Now, for all these good n's. Let n_1 be the largest power of 2 which is no greater than n.

We first provide an informal description of our non-deterministic algorithm. There are two cases according to Theorem 6.7.

- When $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_{n_1}) > \log^{2a} n_1$. On inputs of length n, we apply the PRG with parameter n, and try to compute $L^{\mathsf{hard}}_{n_1}$ on the first n_1 bits in $2^{O(\log^{b_2} n)}$ time.
- When $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_m) > \log^{2a} m$, for an $m \in (2^{\log^{c_1} n_1}, 2^{\log^{c_1} n_1+1}) \cap \mathbb{N}$. Now, on an input of length $n_2 = \mathsf{pair}(m, n) = O(mn^2)$, we apply the PRG with parameter n, and try to compute L^{hard}_m on the first m bits in $2^{O(\log^{b_2} n)} \le 2^{O(\log^{b_2} n_2)}$ time.

Formally, the algorithm is specified in Algorithm 5, with a key sub-routine given in Algorithm 6. The advice bits y_n and z_n are set by Algorithm 7. It is not hard to see that a y_n or a z_n can only be set once.

Analysis of the algorithm. It is easy to see that $L \in \mathsf{NTIME}[2^{\log^{b_2+1} n}]_{/2}$; we set $b \ge b_2 + 1$. Then by our assumption, L can be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star]$ circuits on all sufficiently large input length n. In particular, it also implies that L can be $(1-\delta)$-approximated by $O(\log^a n)$-depth NC circuits on all sufficiently large input length n.

Analysis of $\mathsf{Derand}(x, z, n_0)$. Next, we say an execution of $\mathsf{Derand}(x, z, n_0)$ is correct, if z is the correct advice of $L^{\mathsf{hard}}_{|x|}$, n_0 is good, and $2^{\log^{c_1} n_0+1} > |x| = n$. We first show that on a correct execution of $\mathsf{Derand}(x, z, n_0)$, it non-deterministically computes $L^{\mathsf{hard}}(x)$ (with respect to Definition 8.1). We can assume the corresponding $z = 1$ because otherwise it is trivial. Note that in both cases (whether n is a power of 2 in Algorithm 3), we have $\ell \le 2^{\log^{c_1} n}$ and $D \le \log^{b_1} n$. Therefore, D^C_{checker} is equivalent to a depth $O(\log^{c_1} n + \log^{b_1} n) \le \log S(n_0) = \log^{2b_1 c_1^2} n_0$ circuit ($\log^{c_1} n_0 + 1 > \log n$). Hence, since n_0 is good, for any y_{hard} such that $V(1^{\log^{b_2} n_0}, y_{\mathsf{hard}}) = 1$, $G_{y_{\mathsf{hard}}}$ $1/S(n_0)$-fools D^C_{checker}, and it follows that $\mathsf{Derand}(x, z, n_0)$ non-deterministically computes $L^{\mathsf{hard}}(x)$.

Algorithm 5: Non-deterministic Derandomization of L^{hard}

Given an input x with length $n = |x|$;
Given advice bits $y = y_n \in \{0,1\}$ and $z = z_n \in \{0,1\}$;
if $y = 0$ **then**
  Let n_1 be the largest power of 2 which is no greater than n;
  ($y = 0$ indicates we are in the case that $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_{n_1}) > \log^{2a} n_1$ and n is good.);
  Let w be the first n_1 bits of x;
  $\mathsf{Derand}(w, z_n, n)$;
else
  Parse n as two integers (m_0, n_0) (that is, $n = \mathsf{pair}(m_0, n_0)$);
  ($y = 1$ indicates we are in the case that $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_{m_0}) > 2^{\log^b m_0}$ and n_0 is good.);
  Let w be the first m_0 bits of x;
  $\mathsf{Derand}(w, z_n, n_0)$;

Algorithm 6: $\mathsf{Derand}(x, z, n_0)$

Given an input x with length $n = |x|$, $z \in \{0,1\}$ and n_0;
(z is supposed to be the advice for L^{hard} on input length n and n_0 is suppose to be good.);
(In the following the algorithm tries to derandomize Algorithm 3 with the corresponding advice z.);
if $z = 0$ **then**
  Output 0 and terminate
According to whether n is a power of 2 and Algorithm 3, compute z and ℓ such that $L^{\mathsf{hard}}_n(x) = L^{\mathsf{PSPACE}}_\ell(z)$, and guess an NC circuit C of depth $D = D(n)$;
Compute in $\mathrm{poly}(\ell)$ time a TC^0 instance checker $D^?_{\mathsf{checker}}$ for L^{PSPACE}_ℓ;
Guess a y_{hard} such that $V(1^{\log^{b_2} n_0}, y_{\mathsf{hard}}) = 1$;
for $w \leftarrow \{0,1,?\}$ **do**
  $p_w = \Pr_{r \leftarrow \{0,1\}^{O\left(\log^{b_2} n_0\right)}}[D^C_{\mathsf{checker}}(x, G_{y_{\mathsf{hard}}}(r)) = w]$;
if $p_1 > 0.66$ **then**
  Output 1 and terminate
if $p_0 > 0.66$ **then**
  Output 0 and terminate
Output ?;

Algorithm 7: The algorithm for setting advice bits of Algorithm 5

1 All y_n's and z_n's are set to 0 by default;
2 Let $\mathsf{adv} = \{\mathsf{adv}_n\}_{n\in\mathbb{N}}$ be the advice sequence for L^{hard};
3 **for** $n = 1 \to \infty$ **do**
4   **if** n *is good* **then**
5   Let n_1 be the largest power of 2 which is no greater than n;
6   **if** $\mathit{heur}_{0.99}\text{-}\mathit{DEPTH}(L^{\mathsf{hard}}_{n_1}) > \log^{2a} n_1$ **then**
7             $y_n = 0$;
8             $z_n = \mathsf{adv}_{n_1}$;
9   **else**
10   Let m be an integer from $(2^{\log^{c_1} n_1}, 2^{\log^{c_1} n_1+1}) \cap \mathbb{N}$ such that $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_m) > \log^{2a} m$;
11             $n_2 = \mathsf{pair}(m, n)$;
12             $y_{n_2} = 1$;
13             $z_{n_2} = \mathsf{adv}_m$;

Contradiction. Finally, we show the above is a contradiction. Since there are infinite good n's, either Line 7 or Line 12 of Algorithm 7 is executed for an infinite number of times. We consider the following two cases.

- For an infinite number of good n's, $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_{n_1}) > \log^{2a} n_1$. In this case, L_n computes $L^{\mathsf{hard}}_{n_1}$ for all these n's, and therefore $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_n) = \mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_{n_1}) \geq \log^{2a} n_1 = \omega(\log^a n)$, contradiction.

- For an infinite number of good n's, $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_{n_1}) \leq \log^{2a} n_1$. In this case, L_{n_2} computes L^{hard}_m for all these $n_2 = n_2(n)$'s, and therefore $\mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L_{n_2}) = \mathsf{heur}_{0.99}\text{-}\mathsf{DEPTH}(L^{\mathsf{hard}}_m) \geq \log^{2a} m \geq \omega(\log^a n_2)$ ($m \leq n_2 \leq O(mn^2)$, $m \geq 2^{\Omega(\log^{c_1} n)}$), contradiction.

□

Now, we show Theorem 8.5 implies Theorem 8.3.

Proof of Theorem 8.3. Let $b \geq 1$ be an integer to be specified later, and δ be the universal constant in Theorem 8.5.

For the sake of contradiction, suppose all languages in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ have a $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuit family which computes it correctly on a $1-\delta$ fraction of inputs for all sufficiently large input length n.

In particular, the uniform NC^1 languages considered in the proof of Theorem 4.3 (see Remark 4.4) can be $(1-\delta)$-approximated by $2^{\log^a n}$-size $\mathsf{AC}_{d_\circ}[m_\circ] \circ \mathsf{THR}$ circuit families, for two constants $d_\circ, m_\circ$. Therefore, by Theorem 4.3, there exist constants c_e, c_d such that any depth d-NC circuit has an equivalent $2^{c_e \cdot d^a}$-size $\mathsf{AC}_{d_\circ+c_d}[m_\circ] \circ \mathsf{THR}$ circuit.

Note there is a universal constant c_w such that, for all constants $d_\star$ and $m_\star$, a $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuit has an equivalent $c_w \cdot \log^a n$-depth NC circuit, which in turn has an equivalent $2^{c_e \cdot c_w^a \cdot \log^{a^2} n}$-size $\mathsf{AC}_{d_\circ+c_d}[m_\circ] \circ \mathsf{THR}$ circuits.

Finally, by Theorem 8.5, 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 is not $(1-\delta)$-approximable by $2^{\log^{a^2+1} n}$-size $\mathsf{AC}_{d_\circ+c_d}[m_\circ] \circ \mathsf{THR}$ circuits. By the previous discussion, it follows that L is also not $(1-\delta)$-approximable by $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits for all constants $d_\star, m_\star$, contradiction. $\square$

Remark 8.6. *We remark here that the above proof is in fact* non-constructive*: it doesn't give an explicit bound on the integer b.*

8.3 $1/2 + 1/\operatorname{polylog}(n)$ Average-Case Lower Bounds

Finally, we prove Theorem 1.1 from Theorem 8.3 and hardness amplification.

We first define black-box hardness amplification.

Definition 8.7. A $(1/2-\varepsilon, \delta)$-black-box hardness amplification from input length k to input length $n = n(k)$ is a pair $(\mathsf{Amp}, \mathsf{Dec})$ where Amp is an oracle Turing machine that computes a (sequence of) boolean function on n bits, Dec is a randomized oracle Turing machine on k bits which also takes an advice string of length $a = a(k)$, and for which the following holds. For every pair of functions $f : \{0,1\}^k \to \{0,1\}$ and $h : \{0,1\}^n \to \{0,1\}$ such that

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

there is an advice string $\alpha \in \{0,1\}^a$ such that

$$\Pr_{x \sim \{0,1\}^k}[\mathsf{Dec}^h(x, \alpha) = f(x)] > 1 - \delta.$$

Next we state the hardness amplification result we need.[17]

Theorem 8.8 ([IJKW10])**.** *For all constants $\delta > 0$, and a real $\varepsilon = k^{-o(1)}$, there is a $(1/2 - \varepsilon, \delta)$-black-box hardness amplification from input length k to input length $n = O(k^2)$ with oracle Turing machine pair $(\mathsf{Amp}, \mathsf{Dec})$. Moreover, $\mathsf{Amp}^f(x)$ can be computed in $\operatorname{poly}(n, 1/\varepsilon)$ time for all $x \in \{0,1\}^n$, and $\mathsf{Dec}^?$ can be implemented by a constant-depth circuit of size $\operatorname{poly}(n, 1/\varepsilon)$, with unbounded fan-in $\mathsf{AND}, \mathsf{OR}$ gates and majority gates of fan-in $\Theta(1/\varepsilon)$.*

Remark 8.9. *Since a majority gate of $\Theta(1/\varepsilon)$ fan-in can be computed by an $\exp(1/\varepsilon)$-size AC^0 circuit, the decoder can also be implemented by an AC^0 circuit of size $\operatorname{poly}(n, \exp(1/\varepsilon))$.*

We first prove the following lemma with 2 bits of advice.

Lemma 8.10. *For all constants a, c, there is an integer b and a language L in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ such that L is not $(1/2 + 1/\log^c n)$-approximable by $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.*

Proof. By Theorem 8.3, there is an integer b_1 and a language L' in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^{b_1} n}]_{/2}$ such that L' is not $(1-\delta)$-approximable by $2^{\log^{a_1} n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits, for a universal constant δ, and a constant a_1 to be specified later.

Let $b = b_1 + 1$. Applying Theorem 8.8, we construct another language L, such that on input length of $n = n(k) = O(k^2)$ (we can assume without of loss of generality that the function

[17] Theorem 8.8 can be proved by combing the local-decoder of the direct-product codes [IJKW10], and the local-decoder of Walsh-Hadamard Codes [GL89].

$n : \mathbb{N} \to \mathbb{N}$ is injective), L_n computes the function $\mathsf{Amp}^{L'_k}$ with $\varepsilon = 1/\log^c n$. Clearly, L is in $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$.

By theorem 8.8. For all constants $d_\star, m_\star$, if $L_n = \mathsf{Amp}^{L'_k}$ can be $(1/2+\varepsilon)$-approximated by a $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ of size $2^{\log^a n}$. Then L'_k can be $(1-\delta)$-approximated by an

$$(k \cdot \exp(1/\varepsilon))^{O(1)} \cdot 2^{\log^a n} \leq 2^{\log^a n + O(\log^c n)}$$

size $\mathsf{AC}_{d_\star+c_d}[m_\star] \circ \mathsf{THR}$ circuit, for a universal constant c_d.

Finally, we set $a_1 = 2ac$. Then clearly $2^{\log^{a_1} k} \geq 2^{\log^a n + O(\log^c n)}$. Now, for all constants $d_\star, m_\star$, we know that L' is not $(1-\delta)$-approximable by $2^{\log^{a_1} k}$-size $\mathsf{AC}_{d_\star+c_d}[m_\star] \circ \mathsf{THR}$ circuits, and hence L is not $(1/2+1/\log^c n)$-approximable by $2^{\log^a n}$-size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits. This implies that L is not $(1/2+1/\log^c n)$-approximable by $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. □

Now, Theorem 1.1 follows from the same argument as in [COS18].

Proof of Theorem 1.1. By Lemma 8.10, there is an integer b and a language $L' \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/2}$ such that L' is not $(1/2 + 1/\log^{2c} n)$-approximable by $2^{\log^{2a} n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits. Let $w_0, w_1, w_2, w_3 \in \{0,1\}^2$ be an enumeration of the set $\{0,1\}^2$.

NQP Lower Bounds. We first prove the case for $\mathsf{NTIME}[2^{\log^b n}]$. We define another language $L \in \mathsf{NTIME}[2^{\log^b n}]$ as follows: on an input of length n, let $n' = \lfloor n/4 \rfloor$ and $k = n - 4 \cdot n'$, L_n simulates the non-deterministic algorithm for $L'_{n'}$ with advice w_k, on the first n' bits of input.

By the construction of L', for all constants $d_\star, m_\star$, there is an infinite number of 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 which is not $(1/2+1/\log^{2c} n_i)$-approximable by $2^{\log^{2a} n_i}$ size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits. By the construction of L, $L_{(4 \cdot n_i + a_i)}$ computes a function which is not $(1/2+1/\log^{2c} n_i) \leq (1/2+1/\log^c n)$-approximable by $2^{\log^{2a} n_i} \geq 2^{\log^a n}$ size $\mathsf{AC}_{d_\star}[m_\star] \circ \mathsf{THR}$ circuits. Therefore, L is not $(1/2-1/\log^c)$-approximable by $2^{\log^a n}$-size $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuits.

$(\mathsf{NQP} \cap \mathsf{coNQP})_{/1}$ Lower Bounds. Now we prove the case for $(\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$. We first define another language $L \in (\mathsf{N}\cap\mathsf{coN})\mathsf{TIME}[2^{\log^b n}]_{/1}$ as follows: for an input length n, let $n' = \lfloor n/4 \rfloor$ and $k = n - 4 \cdot n'$. We set the advice bit $a_n = 1$ if and only if w_k is the correct advice for input length n' of language L'. When $a_n = 1$, L_n simulates $L'_{n'}$ with advice w_k, on the first n' bits of input; Otherwise, L_n computes the all-zero function. A similar argument as the previous case completes the proof. □

9 Generalization to Other Natural Circuit Classes

Most of our arguments are pretty generic, the only part that makes use of special properties of $\mathsf{ACC}^0 \circ \mathsf{THR}$ circuit is Lemma 5.1, which builds on the non-trivial SAT algorithm for this circuit class from [Wil14a]. (A non-trivial Gap-UNSAT algorithm also suffices in the argument.)

Therefore, as long as we have a non-trivial SAT or CAPP algorithm for a circuit class $\mathscr{C}$, then our argument can also be used to imply an average-case circuit lower bound against $\mathscr{C}$. In this section we sketch the proof for Theorem 1.2.

Reminder of Theorem 1.2. *For a circuit class $\mathscr{C} \in \{\mathsf{TC}^0, \mathsf{Formula}, \mathsf{P}_{/\,\mathrm{poly}}\}$, if for a constant $\varepsilon > 0$, there is a $2^{n-n^{\varepsilon}}$ time non-deterministic Gap-UNSAT algorithm for $2^{n^{\varepsilon}}$-size $\mathscr{C}$ circuits, then for all constants a, c, NQP is not $(1/2 + 1/n^c)$-approximable by $2^{\log^a n}$-size $\mathscr{C}$ circuits.*

Proof Sketch of Theorem 1.2. We first discuss how to prove a $(1-\delta)$-inapproximability result, for a universal constant δ. When $\mathscr{C} = \mathsf{TC}^0$ or Formulas, the proofs are exactly the same as the case for $\mathsf{ACC}^0 \circ \mathsf{THR}$. (when $\mathscr{C} =$ Formulas, we don't even need Theorem 4.3 to get a collapse from NC^1).

When $\mathscr{C} = \mathsf{P}_{/\,\mathrm{poly}}$, we can no longer use Theorem 6.7. But a similar argument can proceed with Corollary 6.9.

After that, we can use the same hardness-amplification in Theorem 8.8, but since now $\mathscr{C}$ can compute majority, we can prove a $(1/2+1/n^c)$-inapproximability result, instead of a $(1/2+1/\log^c n)$ one. $\square$

10 Open Questions

There are several interesting questions stemming from this work:

- Can we prove more average-case lower bounds for NQP (or even NP) with the techniques in this paper? Recall that the well-known open question of constructing an explicit rigid matrix is just *construct an average-case hard function for low-rank matrices*. Can we construct an NP explicit rigid matrix for any non-trivial regimes of parameters by refining our approach? This would require us to both tighten our algorithm-to-circuit-lower-bounds connection and to find sufficient algorithms for certain tasks on low-rank matrices.

  Or less ambitiously, can we construct an NP explicit function which cannot be approximated by $\omega(\sqrt{n})$ degree $\mathbb{F}_2$ polynomials?

- We can only prove a $1/2 + 1/\operatorname{polylog}(n)$ inapproximability lower bound for NQP against $\mathsf{ACC}^0 \circ \mathsf{THR}$. Can this be improved to a $1/2 + 1/\operatorname{poly}(n)$ one? This could potentially lead us to an unconditional non-deterministic PRG for ACC^0, with poly-logarithmic seed length (the best non-deterministic PRG for ACC^0 has seed length $n - n^{1-\delta}$ [COS18]).

Acknowledgment

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, in particular, for the discussion which leads to the alternative perspective in Section 3.2. 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 want to thank Josh Alman, Chi-Ning Chou, Shuichi Hirahara, Xuangui Huang, Nutan Limaye, Igor Carboni Oliveira, Zhao Song, Emanuele Viola for helpful discussions during this work.

References

[AB84] Miklós Ajtai and Michael 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*, pages 471–474, 1984.

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

[ACW16] Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 467–476, 2016.

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

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

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

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

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1):2:1–2:54, 2009.

[Bab87] László Babai. Random oracles separate PSPACE from the polynomial-time hierarchy. *Inf. Process. Lett.*, 26(1):51–53, 1987.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BHLR19] Abhishek Bhrushundi, Kaave Hosseini, Shachar Lovett, and Sankeerth Rao. Torus polynomials: An algebraic approach to ACC lower bounds. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 13:1–13:16, 2019.

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: - new simple PRF candidates and their applications. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, pages 699–729, 2018.

[BSV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *International Colloquium on Automata, Languages, and Programming*, pages 163–173. Springer, 2014.

[COS18] Ruiwen Chen, Igor Carboni Oliveira, and Rahul Santhanam. An average-case lower bound against ACC^0. In *LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings*, pages 317–330, 2018.

[CT18] Lijie Chen and Roei Tell. Bootstrapping results for threshold circuits "just beyond" known lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:199, 2018.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 316–324, 2004.

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

[GGH+07] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 440–449, 2007.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32, 1989.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GR08] Dan Gutfreund and Guy N. Rothblum. The complexity of local list decoding. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, pages 455–468, 2008.

[GSV18] Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 956–966, 2018.

[Has89] John Hastad. Almost optimal lower bounds for small depth circuits. *Advances in Computing Research*, 5:143–170, 1989.

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 672–683, 2006.

[HVV06] Alexander Healy, Salil P. Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM J. Comput.*, 35(4):903–931, 2006.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

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

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31, 1988.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[MW18] Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. pages 890–901, 2018.

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

[O'D04] Ryan O'Donnell. Hardness amplification within NP. *J. Comput. Syst. Sci.*, 69(1):68–94, 2004.

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

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[San09] Rahul Santhanam. Circuit lower bounds for merlin–arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.

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

[Sho88] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 283–290, 1988.

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

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

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

[Tam16] Suguru Tamaki. A satisfiability algorithm for depth two circuits with a sub-quadratic number of symmetric and threshold gates. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:100, 2016.

[Tel18] Roei Tell. Quantified derandomization of linear threshold circuits. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 855–865, 2018.

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

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

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

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

[Wil14a] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 194–202, 2014.

[Wil14b] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

[Wil16] Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016.

[Wil18] Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 6:1–6:24, 2018.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 1–10, 1985.

[Zák83] Stanislav Zák. A turing machine time hierarchy. *Theor. Comput. Sci.*, 26:327–333, 1983.

A PRG Construction for Low-Depth Circuits

In this section we sketch the proof of Theorem 2.1, which is a simple combination of the local-list deocdable codes in [GR08] and the Nisan-Wigderson PRG construction [NW94].

Reminder of Theorem 2.1. *Let $\delta > 0$ be a constant. There are universal constants c and g, and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, if $Y : \{0,1\}^\ell \to \{0,1\}$ does not have ℓ^δ-depth NC circuit, then for $S = 2^{\ell^{c\cdot\delta}}$, and for all NC circuit C with depth $\log(S)$,*

$$\left| \Pr_{x \in \{0,1\}^w}[C(G(Y,x)) = 1] - \Pr_{x \in \{0,1\}^S}[C(x) = 1] \right| < 1/S,$$

where $w = \ell^g$. That is, $G(Y,\cdot)$ $1/S$-fools all $\log S$-depth NC circuits. Moreover, G is computable in $2^{O(\ell)}$ time.

Proof Sketch. Given such a function Y, we first apply the local-list decodable codes construction in [GR08] to turn it into a sufficiently average-case hard function against low-depth circuit (the decoder in [GR08] has a low-depth implementation). Now we can simply plug the resulting function into the Nisan Wigderson PRG construction [NW94], which completes the proof. □

Remark A.1. *It is also possible to prove the above theorem using the pseudoentropy generator from [STV01] and a low-depth computable extractor (see, e.g. [Tel18] and [CT18] for a construction in sparse TC^0).*

B TC^0 Collapses to ACC^0 if Uniform TC^0 can be Approximated by ACC^0

In this section we provide a proof that TC^0 collapses to ACC^0 if uniform TC^0 can be approximated by ACC^0. Note that the conclusion here is weaker than Theorem 4.3; we include this because the proof is very elementary and does not rely on Barrington's theorem, and also starts with a weaker assumption.[18]

To show TC^0 collapses to ACC^0, it suffices to show that MAJ is in ACC^0.

Lemma B.1. *Let $S : \mathbb{N} \to \mathbb{N}$ be a size parameter and d, m be two constants. Suppose all languages in uniform TC^0 can be 0.99-approximated by S-size $\mathsf{AC}_{d_\star}[m_\star]$ circuit families. Then MAJ can be computed by a $\mathrm{poly}(S,n)$ size $\mathsf{AC}_{d_\star+O(1)}[m_\star]$ circuit family.*

Proof. The following proof is similar to the self-error correction of the parity function[19].

Construction of the Function g in Uniform TC^0. We first construct a function in uniform TC^0, which encodes MAJ in a nice way. Suppose n is a power of 2 for simplicity. Letting $n = 2^\ell$, we fix a natural bijection between $\{0,1\}^\ell$ and $\mathbb{Z}_{2^\ell}$. We also define $\mathsf{Sum}_n : \{0,1\}^{n\cdot\ell} \to \{0,1\}^\ell$, as the summation of n numbers from the group $\mathbb{Z}_{2^\ell}$.

Now we define a function $g : \{0,1\}^{n\cdot\ell} \times \{0,1\}^\ell \to \{0,1\}$, as

$$g(x,y) := \mathsf{Sum}_n(x) \cdot y,$$

where the inner product is over $\mathsf{GF}(2)$. ($g(x,\cdot)$ is just the Walsh-Hadamard encoding of $\mathsf{Sum}_n(x)$.)

[18] In fact, an earlier version of this paper builds on this collapse theorem, with a more complicated argument than the current version.

[19] Suppose F 0.99-approximates the function Parity_n. Let z be a random vector from $\{0,1\}^n$, we have $F(z) \oplus F(z \oplus x) = \mathsf{Parity}_n(x)$ with probability 0.98 for all $x \in \{0,1\}^n$.

From now on we assume n is large enough. Clearly, since both Sum_n and inner product over $\mathbb{F}_2$ have uniform TC^0 circuits, g has uniform TC^0 circuits. Therefore by our assumption, g can be 0.99-approximated by an $S(2n\log n)$-size $\mathsf{AC}_{d_\star}[m_\star]$ circuit C_g. That is,

$$\Pr_{x\in\{0,1\}^{n\cdot\ell}}\ \Pr_{y\in\{0,1\}^{\ell}}[C_g(x,y)=g(x,y)]\geq 0.99.$$

Construction of an ACC Circuit D Approximating Sum_n from C_g. By a simple Markov's inequality, for at least a 0.9 fraction of x from $\{0,1\}^{n\cdot\ell}$, we have

$$\Pr_{y\in\{0,1\}^{\ell}}[C_g(x,y)=g(x,y)]\geq 0.9.$$

We call an x good if it satisfies the above condition. We can use the following simple decoding algorithm to find an ACC circuit $D:\{0,1\}^{n\cdot\ell}\to\{0,1\}^{\ell}$ approximating Sum_n.

Letting $t=10\ell$, we pick t random strings $z_1,z_2,\ldots,z_t$ from $\{0,1\}^{\ell}$. Given $x\in\{0,1\}^{n\cdot\ell}$ and $i\in[\ell]$, the i-th output bit of D is the approximate-majority of $\{C_g(x,z_j)\oplus C_g(x,z_j\oplus e_i)\}_{j\in[t]}$, where e_i is Boolean string that only the i-th bit is 1, and $z_j\oplus e_i$ means the coordinate-wise addition over $\mathsf{GF}(2)$.

Note that for a fixed good $x\in\{0,1\}^{n\cdot\ell}$, $i\in[\ell]$, $\{C_g(x,z_j)\oplus C_g(x,z_j\oplus e_i)\}_{j\in[t]}$'s are independent, and for each $j\in[t]$, we have

$$\Pr_{z_j}[C_g(x,z_j)\oplus C_g(x,z_j\oplus e_i)=\mathsf{Sum}_n(x)_i]\geq 0.8.$$

Therefore, by a simple Chernoff bound, for a good x, we have $D(x)=\mathsf{Sum}_n(x)$ with probability at least $1-1/n$. That is, by an averaging argument, there exists a set of fixed z_j's, such that the constructed circuit D satisfying $D(x)=\mathsf{Sum}_n(x)$ for at least a $0.9\cdot 0.8\geq 0.7$ fraction of inputs.

By Lemma 4.2, D is an $\mathsf{AC}_{d_\star+O(1)}[m_\star]$ circuit of size $\mathrm{poly}(S,n)$.

The Self-correction of Sum_n. Next, we use the property that Sum_n is self-correctable. For all $x\in\{0,1\}^{n\cdot\ell}$, let z be a uniform element from $\mathbb{Z}_{2^\ell}^n$, we have

$$\Pr_z[D(z+x)-\mathsf{Sum}_n(z)=\mathsf{Sum}_n(x)]\geq 0.7.$$

In above, $z+x$ is the element-wise additions over $\mathbb{Z}_{2^\ell}$. The above holds since $z+x$ is uniformly distributed, and D agrees with Sum_n for a 0.7 fraction of inputs. Let $D_z(x):=D(x+z)$, note that D_z has a $\mathrm{poly}(n,S)$-size $\mathsf{AC}_{d_\star+O(1)}[m_\star]$ circuit, as $x+z$ is element-wise addition over $\mathbb{Z}_{2^\ell}$ with each entries on $\ell=O(\log n)$ bits, one can use $2^{O(\ell)}=\mathrm{poly}(n)$ CNFs at the bottom; we can do the same thing at the top to subtract $\mathsf{Sum}_n(z)$ over $\mathbb{Z}_{2^\ell}$, which is a constant; the total depth increase is $O(1)$.

The Final Circuit E. Finally, we pick $10n\ell$ i.i.d. samples $z_1,z_2,\ldots,z_{10\cdot n\cdot\ell}$'s from $\mathbb{Z}_{2^\ell}^n$.

And our final circuit E computes an approximate majority on the $D_{z_j}(x)$'s with the given input x. By a simple Chernoff bound, with a non-zero probability that E computes Sum_n correctly on all inputs. We fix such a collection of z_j's in our construction of E.

Now we have an exact $\mathsf{AC}_{d_\star+O(1)}[m_\star]$ circuit E for Sum_n. One can easily construct an exact $\mathsf{AC}_{d_\star+O(1)}[m_\star]$ circuit for MAJ_n from E, which completes the proof. $\square$

C Average-Case Easy-Witness Lemma for Unary Languages

In this section we sketch the proof for Lemma 3.1.

Reminder of Lemma 3.1. *(Average-Case Easy-Witness Lemma for Unary Languages) There is a universal constant δ such that, for a typical circuit class $\mathscr{C}$*[20]*, if NQP can be $(1-\delta)$-approximated by poly-size $\mathscr{C}$, then all NQP verifiers for unary languages have poly-size $\mathscr{C}$ witness.*

Proof Sketch. We prove the contrapositive. Let $\delta = 1/1000$.

Suppose there is unary language L in NQP such that is doesn't have poly-size $\mathscr{C}$ circuits. And for the sake of contradiction, we further assume NQP can be $(1-\delta)$-approximated by poly-size $\mathscr{C}$. By Theorem 4.3 and the assumption that $\mathsf{AC}^0 \circ \mathscr{C} \subseteq \mathscr{C}$, it follows that NC^1 collapses to poly-size $\mathscr{C}$.

Now we can proceed similarly as the proof of Theorem 5.3 to construct an i.o. quasi-polynomial time NPRG for polylog(n)-depth circuits (we can use the verifier V for L as the "hardness certifier"). Then we can combine it with the a.a.e. average-case MA lower bound for low-depth circuits with a low-depth computable predicate, and proceed identically as Theorem 1.1, to show that NQP cannot be approximated by poly-size $\mathscr{C}$, which is a contradiction. □

D Bootstrapping from Non-trivial Derandomization Algorithms to Quasi-Polynomial Time NPRGs

In this section we sketch the proof for the following bootstrapping theorem, which is implicit in [Wil13, Wil16].

The proof of the following theorem follows roughly as the proof of Theorem 4.1 of [Wil16].

Theorem D.1. *(Informal) For a circuit class $\mathscr{C} \in \{\mathsf{TC}^0, \mathsf{Formula}, \mathsf{P}_{/\,\mathrm{poly}}\}$, if for a constant $\varepsilon > 0$, there is a $2^{n-n^{\varepsilon}}$ time non-deterministic $\mathsf{Gap\text{-}UNSAT}$ algorithm for $2^{n^{\varepsilon}}$-size $\mathscr{C}$ circuits, then there is a quasi-polynomial time non-deterministic infinite often PRG for polynomial-size $\mathscr{C}$ circuits.*

Proof Sketch. From the assumption, and a proof similar to that of Lemma 5.1. We have that for some constant $\delta > 0$, there is an unary NE verifier which doesn't have $2^{n^{\delta}}$-size $\mathscr{C}$ witness.

Then this NE verifier V can be used as the "worst-case hardness certifier" for $\mathscr{C}$ circuits. That is, we have a polynomial-time algorithm $V(x, y)$, where $|x| = n$ and $|y| = 2^n$, such that for an infinite number of n's, $V(x, \cdot)$ is satisfiable, and $V(x, y) = 1$ implies y is not the truth-table of a $2^{n^{\delta}}$-size $\mathscr{C}$ circuit.

Now we guess an y_{hard} such that $V(1^n, y_{\mathsf{hard}}) = 1$. We interpret y_{hard} as a function $f_{\mathsf{ws}} : \{0,1\}^n \to \{0,1\}$. By [GR08], there are local-list decodable codes with TC^0 decoders and polynomial-size blowup. Therefore, one can construct in $2^{O(n)}$ time another function $f_{\mathsf{avg}} : \{0,1\}^{O(n)} \to \{0,1\}$, which is average-case hard for $\mathscr{C}$ circuits (as $\mathscr{C}$ contains TC^0). Plugging f_{avg} into the Nisan-Widgersion PRG construction [NW94] completes the proof. □

[20]Here we require $\mathscr{C}$ is closed under adding AC^0 at the top. That is, $\mathsf{AC}^0 \circ \mathscr{C} \subseteq \mathscr{C}$.

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