Electronic Colloquium on Computational Complexity, Report No. 173 (2017)

An Average-Case Lower Bound against ACC^0

Ruiwen Chen[*] Igor C. Oliveira[†] Rahul Santhanam[‡]

Department of Computer Science
University of Oxford

November 6, 2017

Abstract

In a seminal work, Williams [26] showed that NEXP (non-deterministic exponential time) does not have polynomial-size ACC^0 circuits. Williams' technique inherently gives a worst-case lower bound, and until now, no average-case version of his result was known.

We show that there is a language L in NEXP (resp. $\mathsf{EXP}^{\mathsf{NP}}$) and a function $\varepsilon(n) = 1/\log(n)^{\omega(1)}$ (resp. $1/n^{\Omega(1)}$) such that no sequence of polynomial size ACC^0 circuits solves L on more than a $1/2 + \varepsilon(n)$ fraction of inputs of length n for all large enough n.

Complementing this result, we give a nontrivial pseudo-random generator against polynomial-size $\mathsf{AC}^0[6]$ circuits. We also show that learning algorithms for quasi-polynomial size ACC^0 circuits running in time $2^n/n^\omega(1)$ imply lower bounds for the randomised exponential time classes RE (randomized time $2^{O(n)}$ with one-sided error) and $\mathsf{ZPE}/1$ (zero-error randomized time $2^{O(n)}$ with 1 bit of advice) against polynomial size ACC^0 circuits. This strengthens results of Oliveira and Santhanam [17].

1 Introduction

1.1 Motivation and Background

Significant advances in unconditional lower bounds are few and far between, specially in non-monotone boolean circuit models. In the 80s, there was substantial progress in proving circuit lower bounds for AC^0 (constant-depth circuits with unbounded fan-in AND and OR gates) [2, 7, 28, 10] and $\mathsf{AC}^0[p]$ (AC^0 circuits extended with MOD$_p$ gates) for p prime [18, 21]. But even the case of $\mathsf{AC}^0[m]$ with m composite has remained little understood after decades of investigation, despite our expectation that MOD$_m$ gates do not have much computational power.

In a seminal paper from a few years ago, Williams [26] proved a super-polynomial lower bound against ACC^0 (constant-depth circuits with unbounded fan-in AND, OR and MOD$_m$ gates, for a fixed but arbitrary m) using a new lower bound technique: *the algorithmic method.* This result represents exciting progress on circuit lower bounds after a long gap. However, it has a couple of drawbacks when compared to previous lower bounds.

[*]Email: `rwchenmail@gmail.com`. Most of this work was conducted while the author was a postdoctoral researcher at the University of Oxford.

[†]Email: `igor.carboni.oliveira@cs.ox.ac.uk`.

[‡]Email: `rahul.santhanam@cs.ox.ac.uk`.

ISSN 1433-8092

First, while previous lower bounds were for *explicit* functions, i.e., functions in P (deterministic polynomial time), Williams' lower bound is only known to hold for functions in NEXP [26], or in closely related classes [27]. (We note that even proving a lower bound for these much larger classes had been a longstanding open problem.) Unfortunately, the algorithmic method of Williams does not seem to be easily adaptable to give lower bounds for explicit functions.

Second, previous lower bounds and their subsequent extensions worked also in the *average case* setting, i.e., they showed that there were explicit functions which cannot be computed by polynomial-size circuits on significantly more than half the inputs of any input length. In other words, even computing the function correctly on a random input is hard. Williams' lower bound, on the other hand, only seems to give a worst-case lower bound, meaning that any polynomial-size family of ACC^0 circuits is only guaranteed to fail to compute the hard function in NEXP on a negligible fraction of inputs of length n, for infinitely many n. The question of strengthening the existing worst-case ACC^0 lower bound to the average case has been recently posed and discussed in [27] and [1].

1.2 Our Results

Our main result addresses this second drawback of Williams' lower bound, and strengthen the main result from [26] to an average-case lower bound.

Theorem 1 (An average-case lower bound against ACC^0)**.**
There is a function $\varepsilon(n) = 1/\log(n)^{\omega(1)}$ such that the following holds. There is a language L in NEXP such that for any polynomial-size family $\{C_n\}$ of ACC^0 circuits, there are infinitely many n such that C_n computes L correctly on at most a $1/2 + \varepsilon(n)$ fraction of inputs of length n.

Our proof of Theorem 1 in fact gives a much smaller upper bound on $\varepsilon(n)$, but stating this bound is a bit technical, so we defer it to the main body of the paper.

Before sketching the main ideas behind the proof of Theorem 1, we attempt to explain why the original proof based on the algorithmic method fails to give an average-case lower bound. Williams' proof [26] employs an indirect diagonalization technique. The technique exploits Williams' algorithm solving satisfiability of poly-size ACC^0 circuits in time $2^{n-\omega(\log(n))}$. Assume, for the sake of contradiction, that $\mathsf{NTIME}(2^n)$ does have ACC^0 circuits of polynomial-size. It can be shown from this assumption that languages in $\mathsf{NTIME}(2^n)$ have *succinct witnesses*, i.e., YES instances have witnesses which can be represented succinctly by ACC^0 circuits of size $\mathsf{poly}(n)$. Now we can use the following guess-and-check procedure to compute any $L \in \mathsf{NTIME}(2^n)$ in non-deterministic time $2^n/n^{\omega(1)}$, contradicting the non-deterministic time hierarchy theorem. We guess a poly-size ACC^0 circuit encoding a witness for the instance, and then check that this circuit indeed encodes a witness by using the satisfiability algorithm for ACC^0 circuits. From the fact that the satisfiability algorithm runs in time $2^n/n^{\omega(1)}$, it follows that this guess-and-check procedure runs in time $2^n/n^{\omega(1)}$, giving the desired contradiction to the non-deterministic time hierarchy theorem. (This is just a high-level description – we refer to [26] for more details.)

A crucial step in the above proof is to use the assumption that NEXP is in poly-size ACC^0 to get succinct witnesses for NEXP languages. This step simply does not work if our asssumption is that NEXP is in poly-size ACC^0 on average rather than in the worst case, and the proof fails completely.

It seems difficult to adapt the algorithmic method to get an average-case lower bound, so we try a different approach. A popular paradigm in complexity-theoretic pseudorandomness is *hardness amplification*: transforming a function that is worst-case hard for some class of circuits to a function that is average-case hard for the same class of circuits. Williams' result gives us a worst-case lower bound against polynomial-size ACC^0 circuits. Can we use hardness amplification to derive an average-case lower bound from this?

There are a couple of obstacles we need to overcome to make this approach work. First, hardness amplification typically requires that the class of circuits against which we are performing amplification can compute the Majority function [20]. We are trying to show an average-case lower bound against ACC^0 circuits, and we do not believe that poly-size ACC^0 circuits can compute Majority. However, while this does preclude us from amplifying to hardness $1/2 - 1/\mathsf{poly}(n)$ (i.e., showing that any circuits computing the function must fail on a $1/2 - 1/\mathsf{poly}(n)$ fraction of inputs) in a black-box way, we can still hope for weaker hardness amplification results which get us hardness $1/2 - o(1)$. Indeed, using the connection between hardness amplification and list-decodable error-correcting codes due to Sudan, Trevisan and Vadhan [23], hardness amplification procedures are known [8, 9] which are applicable in our setting.

Second, and more problematically, the hard function we begin with is in NEXP, and we would like our new function resulting from hardness amplification also to be in NEXP. If we were to do hardness amplification in a black-box way starting from a NEXP function, the amplification needs to be *monotone*, and it is not hard to see that black-box monotone hardness amplification cannot even amplify worst-case hardness to hardness 0.99.[1]

To overcome this obstacle, we use instead a later result of Williams [27], where he shows that his lower bound [26] also holds for a function in $(\mathsf{NE} \cap \mathsf{coNE})/1$, i.e., both in $\mathsf{NE} = \mathsf{NTIME}[2^{O(n)}]$ and $\mathsf{coNE} = \mathsf{coNTIME}[2^{O(n)}]$, but with 1 bit of advice depending only on the input length. The advantage of starting from this later result of Williams is that when standard hardness amplification is applied, the resulting function *stays* in $(\mathsf{NE} \cap \mathsf{coNE})/1$.

This still leaves the problem of eliminating the 1 bit of advice in the upper bound. Doing this in a naive way would stop us from achieving hardness less than $3/4$, but we show how to eliminate the advice with a negligible loss in our hardness parameter. This concludes our high-level description of the proof of Theorem 1.

A natural question that arises when we have an average-case lower bound against a circuit class is whether we can construct *pseudo-random generators* (PRG) against the circuit class. An influential paradigm of [15], motivated by a similar paradigm in the cryptographic setting, shows how to transform average-case hardness into pseudorandomness, and conversely. However, this is only applicable when we have a hardness parameter $\varepsilon(n) \leq 1/n^{\Omega(1)}$, which Theorem 1 fails to achieve.

More recent work of [6] studies how to derive pseudorandomness from average-case hardness in cases where the hardness is not strong enough to apply [15]. It is shown in [6] that when the hard function has a property known as *resamplability* (a certain form of random self-reducibility), it is possible to get low-stretch pseudorandom generators with error $o(1)$ even under the weaker assumption that $\varepsilon(n) = o(1)$. We cannot directly apply their result in our setting because it is unclear if our hard function in NEXP satisfies the resamplability property.

However, by a win-win analysis and a result from [6], we are able to get a low-stretch pseudo-random generator against $\mathsf{AC}^0[6]$.[2] Ideally, we would like this generator to be computable in deterministic exponential time, but because our hard function for ACC^0 is in $(\mathsf{NE} \cap \mathsf{coNE})/1$, we are only able to get computability in *strong non-deterministic linear exponential time with 1 bit of advice*.[3]

[1]This corresponds to monotone error-correcting codes, which cannot have good distance. We refer to [4] for more details.

[2]We stick to modulo 6 gates mostly for simplicity. Our result can be extended to any modulus m for which Theorem 15 holds. We refer to [6] for more details.

[3]In other words, the non-deterministic algorithm, when given the correct advice bit (that only depends on the

Theorem 2 (A non-deterministic pseudo-random generator against $\mathsf{AC}^0[6]$).
For every depth $d \geq 1$ and $\delta > 0$, there is a sequence of functions $\{G_n\}$ computable in $(\mathsf{NE}\cap\mathsf{coNE})/1$, where each $G_n : \{0,1\}^\ell \to \{0,1\}^n$ has seed length $\ell(n) = n - n^{1-\delta}$, for which the following holds. Let $\{C_n\}$ be a sequence of $\mathsf{AC}^0[6]$ circuits, where each C_n has depth $\leq d$ and size $\leq n^d$. Then, for infinitely many values of n,

$$\left|\mathbf{Pr}_{y\in\{0,1\}^\ell}[C_n(G_n(y)) = 1] - \mathbf{Pr}_{x\in\{0,1\}^n}[C_n(x) = 1]\right| \leq o(1).$$

We observe that, using the pseudo-random generator in Theorem 2, we can get an alternative proof of a variant of Theorem 1. Since this is obtained in a somewhat more indirect way, we do not discuss it further.

There are a couple of directions in which we could aspire to strengthen these results. First, in Theorem 1, we might hope to get a hardness parameter $\varepsilon(n) = 1/n^{\Omega(1)}$, or even $\varepsilon(n) = 1/n^{\omega(1)}$. Indeed, we are able to obtain an analogous result with $\varepsilon(n) = 1/n^{\Omega(1)}$, but for a hard function in E^{NP} instead of NEXP (Section 4). Nevertheless, getting even stronger results seems to be a difficult task using existing techniques, for the following reason. Even for the substantially simpler case of $\mathsf{AC}^0[p]$ circuits, when p is prime, we do not know how to get $\varepsilon(n) = o(1/\sqrt{n})$ for an explicit function, and showing a stronger hardness result is a long-standing open problem (cf. [22]).

Second, we could hope to get a PRG computable in *deterministic* linear exponential time in Theorem 2. But this would imply that EXP is hard on average for poly-size $\mathsf{AC}^0[6]$ circuits, and so far we have been unable to show even *worst-case* hardness against poly-size $\mathsf{AC}^0[6]$ for EXP. This brings us back to the first drawback in Williams' algorithm technique, discussed in Section 1.1, and which we further explore now.

While substantially improving the explicitness in Williams' lower bounds [26, 27] and in Theorem 1 remains a major challenge, [16] recently introduced another approach that could lead to further progress in this direction. They considered a *learning-theoretic* analogue of Williams' approach. While Williams derives circuit lower bounds from circuit satisfiability algorithms that are "non-trivial" in that they beat the naive brute force search algorithm, [16] shows implications for circuit lower bounds from learning algorithms that are similarly non-trivial in that they beat a brute force learning approach.

We say that a randomized learning algorithm is a *non-trivial* learner for a circuit class $\mathcal{C}$ if it runs in time bounded by $2^n/n^{\omega(1)}$. For concreteness and simplicity, we consider learning algorithms that make membership queries, and that learn under the uniform distribution with error at most $1/n$ and with failure probability at most $1/n$.

For convenience, we use $\mathsf{ACC}^0_{d,m}(s(n))$ to denote the class of boolean functions computable by depth-d ACC^0 circuits over a fixed modulo m and of size $\leq s(n)$. The following connection between learning algorithms and non-uniform lower bounds was established in [16].

Proposition 1 (REXP lower bounds from learning sub-exponential size ACC^0 circuits [16]).
If for every depth $d \geq 1$ and modulo $m \geq 1$ there is $\varepsilon > 0$ such that $\mathsf{ACC}^0_{d,m}(2^{n^\varepsilon})$ can be learned in non-trivial time, then $\mathsf{REXP} \nsubseteq \mathsf{ACC}^0$.

Recall that $\mathsf{REXP} \subseteq \mathsf{NEXP}$ is the class of languages decided by one-sided randomized exponential time algorithms, and that under standard derandomization hypotheses, $\mathsf{REXP} = \mathsf{EXP}$.[4] Consequently, Proposition 1 offers a potential path to more explicit (worst-case) ACC^0 lower bounds

input length parameter), outputs either "abort" of the correct string, and outputs the correct string in at least one computation path. We refer to Section 2 for more details.

[4]For a concrete example of the benefits of improving an NEXP lower bound to randomized exponential time classes such as REXP, we refer the reader to [17].

via the design of non-trivial learning algorithms, and it can be seen as another instantiation of the algorithmic method.[5]

However, note that the learnability of *sub-exponential* size circuits is a strong assumption. Indeed, by the Speedup Lemma of [16], it implies that polynomial size ACC^0 circuits can be learned in *quasi-polynomial* time, a result that is only known to hold for AC^0 and $\mathsf{AC}^0[p]$ circuits [5]. Ideally, we would like to get stronger and more explicit lower bounds from much weaker assumptions.

The proof of Proposition 1 relies on a variety of techniques from complexity theory. An important element in the argument is the use of Williams' unconditional proof that $\mathsf{NEXP} \nsubseteq \mathsf{ACC}^0$. This lower bound is employed as a *black-box* in the argument from [16], and in Section 8 of the same work, the authors speculate about the possibility of establishing stronger connections between non-trivial algorithms and lower bounds by combining ideas from different frameworks.

We present a new application of the interaction between the learning framework of [16], and the satisfiability framework of Williams [26, 27]. We combine *the proofs* of existing connections between non-trivial algorithms and non-uniform lower bounds, and establish the following result.

Theorem 3 (Stronger connection between ACC^0-learnability and lower bounds)**.**
Assume that for every fixed choice of parameters $d, m, c \geq 1$, the class $\mathsf{ACC}^0_{d,m}(n^{(\log n)^c})$ can be non-trivially learned. Then,

$$\mathsf{RTIME}[2^{O(n)}] \nsubseteq \mathsf{ACC}^0(n^{\log n}) \quad \textit{and} \quad \mathsf{ZPTIME}[2^{O(n)}]/1 \nsubseteq \mathsf{ACC}^0(n^{\log n}).$$

We note that the worst-case lower bound for $\mathsf{ZPTIME}[2^{O(n)}]/1$ in Theorem 3 can be strengthened to an average-case lower bound using the same technique as in the proof of Theorem 1. We also note that more generally, for any circuit class $\mathcal{C}$ closed under composition with AC^0 circuits, such an average-case lower bound against $\mathcal{C}$ follows from the existence of non-trivial learning and satisfiability algorithms for $\mathcal{C}$, using the same proof as for Theorem 3.

Observe that this result strengthens Proposition 1 in a few ways. The assumption is considerably weaker, and the lower bound is quantitatively stronger. In addition, it provides a lower bound for *zero-error* randomized computations with one bit of advice, while in Proposition 1 the randomized algorithm computing the hard function makes mistakes. Interestingly, Theorem 3 is not known to hold for other circuit classes, and its proof explores specific results about ACC^0 circuits in a crucial way.

We note that there is a connection between non-trivial algorithms and non-uniform lower bounds for ZPEXP, but it assumes the existence of P-natural properties useful against *sub-exponential* size circuits (see Theorem 44 from [16], and also [11]). Although in Theorem 3 the uniformity over the hard language is not as strong (i.e., REXP and $\mathsf{ZPEXP}/1$ versus ZPEXP), it almost matches the uniformity condition, while its assumption is considerably weaker.

Organization. We introduce definitions, notation, and some necessary technical results in the following section. In Section 3, we explore Williams' proof that $(\mathsf{NE} \cap \mathsf{coNE})/1 \nsubseteq \mathsf{ACC}^0$ [27], alluded to in the discussion above. We obtain slightly better bounds by using a more careful choice of parameters, which is important in the proof of Theorem 1 and its extensions. Crucially, we extract from his argument a stronger technical consequence (Lemma 1) that plays a fundamental role in the proof of Theorem 3. We then present the proofs of Theorems 1, 2, and 3 and their consequences in Sections 4, 5, and 6, respectively.

[5]The design of concrete non-trivial learning algorithms for some circuit classes and in some alternative but related learning models has been recently investigated in [19].

2 Preliminaries

2.1 Complexity Classes and Basic Definitions

Let $\mathsf{TIME}[t(n)]$ be the classes of languages decided by deterministic Turing machines (TM) running in time $O(t(n))$, and let $\mathsf{NTIME}[t(n)]$ be the class of languages decided by non-deterministic Turing machines (NTM) running in time $O(t(n))$. We use standard notions of complexity classes, such as P, NP, EXP, NEXP, etc. In particular, $\mathsf{E} = \mathsf{TIME}[2^{O(n)}]$, $\mathsf{NE} = \mathsf{NTIME}[2^{O(n)}]$, and L is the class of languages computable in (uniform) logarithmic space. A function $t : \mathbb{N} \to \mathbb{N}$ is time-constructible if there is a TM M, which on input 1^n outputs $t(n)$ in time $O(t(n))$. We sometimes informally use the term algorithm instead of Turing machines. We refer to a textbook such as [3] for more background in complexity theory.

A *strong non-deterministic Turing machine* (SNTM) is a NTM where each branch of the computation has one of three possible outputs: 0, 1, and '?'. We say that a SNTM M decides a language L if the following *promise* holds: if $x \in L$, each branch ends with 1 or '?', and at least one branch ends with 1; if $x \notin L$, each branch ends with 0 or '?', and at least one branch ends with 0. It is easy to see that a language $L \in \mathsf{NE} \cap \mathsf{coNE}$ if and only if L is decided by a SNTM in time $2^{O(n)}$. When we say that a sequence of functions $G_n \colon \{0,1\}^\ell \to \{0,1\}^n$ is computed by a SNTM M, we formally mean that the language $L_G \subseteq \{0,1\}^\star$ that encodes $\{G_n\}$ is computed by M, where L_G is defined in a natural way. For concreteness, we let L_G be the set of strings encoding tuples $\langle 1^n, y, i, b\rangle$, where $b \in \{0,1\}$, $y \in \{0,1\}^{\ell(n)}$, $i \in [n]$, and $G_n(y)_i = b$. We assume that the tuples obtained from each choice of the parameter n have all the same length as strings in $\{0,1\}^\star$ (this is relevant when defining computation with advice below).

We define *advice classes* as follows. For a deterministic or non-deterministic uniform complexity class $\mathcal{C}$ and a function $\alpha(n)$, the class $\mathcal{C}/\alpha(n)$ is the set of languages L such that there is a language $L' \in \mathcal{C}$ and a sequence of strings $\{a_n\}$ with $|a_n| = \alpha(n)$ which satisfy that $L(x) = L'(x, a_{|x|})$ for all strings $x \in \{0,1\}^\star$.

For *semantic classes* $\mathcal{C}$ (such as BPP, $\mathsf{NE} \cap \mathsf{coNE}$, etc.) with advice, we only require the *promise condition* for the class $\mathcal{C}$ to hold when the correct advice is given. For example, a language L is in $(\mathsf{NE} \cap \mathsf{coNE})/\alpha(n)$ if there is a SNTM M running in time $2^{O(n)}$ and a sequence of advice strings $\{a_n\}$ with $|a_n| = \alpha(n)$ such that, on each input x, the computation paths of $M(x, a_{|x|})$ satisfy the promise condition in the definition of SNTMs. Note that M running with incorrect advice may not satisfy the promise on its branches.

We also define *infinitely often classes*. For a (syntactic) deterministic or non-deterministic class $\mathcal{C}$, the class $\mathsf{i.o.}\mathcal{C}$ is the set of languages L for which there is a language $L' \in \mathcal{C}$ such that, for infinitely many values of n, $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$. For a semantic class $\mathcal{C}$, we relax the definition, and let $\mathsf{i.o.}\mathcal{C}$ be the class of languages L decided by a Turing machine M such that, for infinitely many input lengths n, M is of type $\mathcal{C}$ on inputs of length n (i.e., it satisfies the corresponding promise). Note that M might not be of type $\mathcal{C}$ on other input lengths.

We use standard notation for circuit classes. In particular, AC^0 is the class of circuit families of constant depth and polynomial size, with AND, OR, and NOT gates, where AND and OR gates have unbounded fan-in. $\mathsf{AC}^0[m]$ extends AC^0 by allowing unbounded fan-in MOD_m gates, where m is fixed, and $\mathsf{ACC}^0 \overset{\text{def}}{=} \bigcup_m \mathsf{AC}^0[m]$ (we often write $\mathsf{AC}^0[m]$ and $\mathsf{ACC}^0[m]$ interchangeably). For convenience, we use $\mathcal{C}_d(s)$ to restrict a circuit class to circuits of depth $\leq d$ and size $\leq s$. Finally, recall that NC^1 is the class of circuit families of logarithmic depth and polynomial size with AND, OR, and NOT gates, where the gates have bounded fan-in. We often deliberately conflate a class of circuit families with the class of languages computed by the circuit families. These circuit families are all *non-uniform*, unless otherwise stated.

We say that a language L is $\gamma(n)$-*hard* for a circuit class $\mathcal{C}$ if for each $L' \in \mathcal{C}$ and for infinitely many values of n, $\mathbf{Pr}_{x\in\{0,1\}^n}[L(x) = L'(x)] \leq 1 - \gamma(n)$. Finally, a class Γ is $\gamma(n)$-hard for $\mathcal{C}$ if there is a language in Γ that is $\gamma(n)$-hard for $\mathcal{C}$.

In Section 6, we show some consequences of learnability. Our model of learnability here is randomized learning with membership queries – the learner is given access to an oracle for the target function on the given input length n, and is allowed to make arbitrary queries to the oracle. The task of the learner is to output with probability $1 - o(1)$ a hypothesis which agrees with the oracle on a $1 - o(1)$ fraction of inputs of length n.

Given a circuit class $\mathcal{C}$, a size bound s and a time bound T, we say that $\mathcal{C}[s(n)]$ is learnable in time T if there is a learner running in time $T(n)$ which succeeds given any oracle computed by circuits from $\mathcal{C}$ of size $s(n)$. We say that $\mathcal{C}[s(n)]$ is *non-trivially learnable* if it is learnable in time $2^n/n^{\omega(1)}$. Note that the class of all Boolean functions is trivially learnable in time $O(2^n)$ with this definition of learnability, just by querying every input of length n and outputting the truth table. Non-trivial learnability corresponds to learners that are just marginally more efficient than this.

For more detailed definitions of concepts in this learning model, see [16].

2.2 ACC^0 Lower Bounds and Pseudorandomness

We recall the following ACC^0 circuit lower bounds.

Theorem 4 ([26]). *For every $d \geq 1$ and $m \geq 1$, there is a $\delta > 0$ and a language in E^{NP} that is not computable by a sequence of $\mathsf{ACC}^0[m]$ circuits of depth d and size $O(2^{n^\delta})$.*

Theorem 5 ([27]). *There is a language in $(\mathsf{NE} \cap \mathsf{coNE})/1$ that does not admit ACC^0 circuits of size $O(n^{\log n})$.*

The results in Section 3 make use of the following connection between hard functions and pseudo-random generators.

Theorem 6 ([25]). *There is a fixed constant $\lambda \geq 1$ and a function $G\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for every $s \in \mathbb{N}$ and $Y \in \{0,1\}^*$, where Y viewed as the truth-table of a Boolean function requires circuits of size $\geq s^\lambda$, and for all circuits C of size $\leq s$,*

$$\left|\mathbf{Pr}_{x\in\{0,1\}^{\lambda \log|Y|}}[C(G(Y,x)) = 1] - \mathbf{Pr}_{w\in\{0,1\}^s}[C(w) = 1]\right| \leq 1/s.$$

In addition, G can be computed in deterministic time $\mathsf{poly}(|Y|)$.

2.3 Error Correcting Codes and Hardness Amplification

We follow part of the terminology from [9]. For our applications in Section 4, we need error correcting codes with certain special properties that admit a uniform encoding procedure, but whose decoding can be non-uniform.

Definition 1 (Local-list-decoding in error correcting codes). *We say that a family $\{C_M\}_M$ of functions $C_M\colon \{0,1\}^M \to \{0,1\}^N$ is a* (d, L)-locally-list-decodable code *if there is an oracle Turing machine D that takes an index $i \in [M]$, advice $a \in [L]$, and a random string r, and for which the following holds. For every input $x \in \{0,1\}^M$ and $y \in \{0,1\}^N$ for which $\Delta(C_M(x), y) \leq d$, there exists $a \in [L]$ such that, for all $i \in [M]$,*

$$\mathbf{Pr}_r[D^y(i, a, r) = x_i] > 9/10.$$

Here $\Delta(w_1, w_2) \in [0, 1]$ is the relative hamming distance *between strings w_1 and w_2, and one should think of $N = N(M)$, $d = d(M)$, etc. as a sequence of parameters indexed by M. We say that a code of this form is* explicit *if it can be computed in time $\mathsf{poly}(N(M))$.*

We will need results on hardness amplification and constructions of efficient error correcting codes.

Definition 2 (Black-box hardness amplification)**.** *A $(1/2 - \epsilon, \delta)$-*black-box hardness amplification *from input length k to input length n 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, and for which the following holds. For every pair of functions $f\colon \{0,1\}^k \to \{0,1\}$ and $h\colon \{0,1\}^n \to \{0,1\}$ such that*

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

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

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

(We will also view Dec^h as a non-uniform oracle boolean circuit. Observe that if $\delta = 2^{-k}$ then there is a way to fix the randomness and the advice string of Dec^h so that it correctly computes f on every input $x \in \{0,1\}^k$.[6])

The following is a well-known connection [23] between fully black-box hardness amplification and binary locally-list-decodable codes. (We will not formally specify all details of this reduction since this is a standard construction.)

Theorem 7 (Connection between hardness amplification and local-list-decodable codes)**.**
If there is a $(1/2 - \epsilon, L)$-locally list decodable code $C\colon \{0,1\}^K \to \{0,1\}^N$ with a corresponding decoder D then there is a $(1/2 - \epsilon, 2^{-k})$-black-box hardness amplification procedure from length $k = \log K$ to length $n = \log N$, where Amp is defined by the encoder of C, and Dec is defined by the decoder D with advice length $a = \log L$.

We need the following construction of list-decodable codes (and corresponding hardness amplification procedure).

Theorem 8 (Efficient construction of locally-list-decodable codes [8, 9])**.** *For every choice of $\exp(-\Theta(\sqrt{\log M})) \leq \epsilon < 1/2$, there is an explicit $(1/2 - \epsilon, \mathsf{poly}(1/\epsilon))$-locally-list-decodable code $C_M\colon \{0,1\}^M \to \{0,1\}^{\mathsf{poly}(M)}$ with a local decoder that can be implemented by a family of constant-depth circuits of size $\mathsf{poly}(\log M, 1/\epsilon)$ using majority gates of fan-in $\Theta(1/\epsilon)$ and AND/OR gates of unbounded fan-in.*

We can get the following AC^0 decoder by a standard simulation of majority gates using large AC^0 circuits.

Corollary 1 (Limited hardness amplification via constant-depth circuits of bounded size)**.**
For every parameter $\exp(-\Theta(\sqrt{\log M})) \leq \epsilon < 1/2$ and each large enough constant d, there is an explicit $(1/2 - \epsilon, \mathsf{poly}(1/\epsilon))$-locally-list-decodable code $C_M\colon \{0,1\}^M \to \{0,1\}^{\mathsf{poly}(M)}$ with a local decoder that can be implemented by AC^0 circuits of size $\mathsf{poly}(\log M, \exp((1/\epsilon)^{O(1/d)}))$ and depth at most d.

[6]Note that the process of amplifying the success probability of randomized algorithms and fixing the randomness can be done with only an AC^0 overhead on the overall complexity, since approximate majority functions can be computed in this circuit class.

Corollary 1 and the connection to hardness amplification are the crucial results needed in Section 4. We will implicitly use these locally-list-decodable codes in order to amplify from worst-case hardness to average-case hardness.

3 Stronger Consequences from Williams' Lower Bound Argument

In this section, we aim to improve Williams' size lower bound [27] from $n^{\log n}$ to sub-third-exponential size. The argument closely follows his presentation, but our choice of parameters will allow us later on to prove Theorem 1 and its extensions. We also extract a certain stronger formulation of his lower bound (Lemma 1 below) that plays an important role in Section 6.

We need the following technical definitions. A function $f\colon \mathbb{N} \to \mathbb{N}$ is *sub-half-exponential* if for every fixed $k \geq 1$, $f(f(n^k)^k) \leq 2^{n^{o(1)}}$. Similarly, a function $g\colon \mathbb{N} \to \mathbb{N}$ is *sub-third-exponential* if for every fixed $k \geq 1$, $g(g(g(n^k)^k)^k) \leq 2^{n^{o(1)}}$. For instance, for a fixed integer $a \geq 1$, $g(n) \overset{\text{def}}{=} 2^{(\log n)^a}$ is sub-third-exponential. For technical reasons, throughout this section we will restrict our attention to time-constructible functions only, even when this is not explicitly stated.

We introduce some definitions to capture languages with witnesses encoded by circuits. We say that an algorithm $V(x, y)$ is a *good predicate* for a language $L \in \mathsf{NTIME}[t(n)]$ if

- V runs in time at most $\mathsf{poly}(|y| + t(|x|))$;
- For every input $x \in \{0,1\}^\star$, $x \in L$ if and only if there is (a witness string) y such that $V(x, y) = 1$, where for convenience we assume that the length of y is a power of two, and that $|y| \geq \Omega(t(|x|))$ for every witness.

Clearly, for each $L \in \mathsf{NTIME}[t(n)]$, there is at least one good predicate for L. Let V be such a predicate, and $\mathcal{C}$ be a circuit class, such as ACC^0. We say V has *$\mathcal{C}$-witnesses* of size $s(n)$ if for every x, if $x \in L$ then there is a $\mathcal{C}$-circuit C_x of size $\leq s(n)$ such that $V(x, \mathsf{tt}(C_x)) = 1$, where $\mathsf{tt}(C_x)$ is the string that encode the truth-table of the function computed by C_x. Similarly, L has $\mathcal{C}$-witnesses of size $s(n)$ if for every good predicate V for L, V admits $\mathcal{C}$-witnesses of size $\leq s(n)$.

Recall that a language $L \subseteq \{0,1\}^\star$ is *unary* if it only contains strings of the form 1^ℓ, for different values of ℓ.

Theorem 9 (A variant of Corollary 4.2 from [27])**.** *For every fixed depth $d \geq 1$ and modulus $m \geq 1$, there is $\epsilon > 0$ and an (infinite) unary language L^u in $\mathsf{NTIME}[2^n]$ that does not have witness strings encoded by $\mathsf{AC}^0[m]$ circuits of depth $\leq d$ and size $O(2^{n^\epsilon})$.*

In Corollary 4.2 of [27], the same result is stated under the additional assumption that P admits ACC^0 circuits of size $n^{\log n}$. This extra hypothesis is used to show that general circuits can be simulated by ACC^0 circuits of appropriate size, an important ingredient in the proof. The assumption is no longer necessary, thanks to the more efficient reduction presented in [12].

The Circuit Approximation Probability Problem (CAPP) is defined as follows: Given a boolean circuit C, compute a rational number $p \in [0,1]$ (represented as a binary string) such that $\left|\mathbf{Pr}_x[C(x) = 1] - p\right| \leq 1/6$.

In order to simplify our next statement, we will abuse notation, and for an increasing function $S\colon \mathbb{N} \to \mathbb{N}$, we use $S^{-1}(b)$ to denote the least element $a \in \mathbb{N}$ such that $S(a) \geq b$.

Theorem 10 (Analogue of Theorem 4.1 in [27])**.** *Suppose P has ACC^0 circuits of size $t(n)$. Let $S(n) \geq s(n)^\lambda \geq n$ be an increasing function such that $t(10 \cdot S(n) \log S(n))) \leq 2^{n^\epsilon}$, for each fixed $\epsilon > 0$ and every large enough n, where $\lambda \geq 1$ is the constant in Theorem 6. Then, for infinitely many values of n, CAPP on circuits of size $\leq s(n)$ and over n input variables is computable in nondeterministic time $\mathsf{poly}(n, 2^{S^{-1}(s(n)^\lambda)})$.*

Williams [27] established a similar result using $t(n) = n^{\log n}$, and here we verify this more general version.

Sketch of the proof of Theorem 10. Recall the definition of the Circuit Evaluation Problem (CEP): Given a boolean circuit C and a string x, each encoded by a string of length $\leq n$, decide if $C(x) = 1$. Clearly, CEP is in P. If $\mathsf{P} \subseteq \mathsf{E}$ admits ACC^0 circuits of size $t(n)$, then CEP has $\mathsf{AC}^0[m^\star]$ circuits of depth $d^\star$ and size $t(O(n))$, for fixed $d^\star, m^\star \geq 1$. This implies that any circuit C of size $\leq S$ on $n \leq S$ input bits can be converted to an $\mathsf{AC}^0[m^\star]$ circuit of depth $\leq d^\star$ and size $\leq t(10 \cdot S \log S)$, by plugging in the description of C into the corresponding inputs of the $\mathsf{AC}^0[m^\star]$ circuit solving the Circuit Evaluation Problem.

By Theorem 9, we know that there is $\epsilon > 0$ and an infinite unary language $L^u \in \mathsf{NTIME}[2^n]$ that does not admit witnesses encoded by $\mathsf{AC}^0[m^\star]$ circuits of depth $\leq d^\star$ and size $\leq 2^{n^\epsilon}$. By the preceding paragraph, L^u does not have witnesses encoded by *unrestricted* circuits of size $\leq S$, using the assumption that $t(10 \cdot S \log S)) \leq 2^{n^\epsilon}$. Accordingly, let V be a good predicate for L^u that does not admit such witnesses for infinitely many values of n.

In order to complete the proof, we rely on the pseudorandom generator G from Theorem 6. It follows from this result using standard techniques that to solve CAPP over circuits of size $s = s(n)$, it is sufficient to obtain a string Y that encodes a function of circuit complexity at least s^λ. The overall running time of the resulting CAPP algorithm is then $\mathsf{poly}(n, 2^{\lambda \cdot \log |Y|}) = \mathsf{poly}(n, |Y|)$.

We obtain a hard truth-table nondeterministically using the good predicate V, in the following way. Since L^u is infinite, unary, and V does not admit witnesses of circuit size $\leq S$ infinitely often, there is an infinite set $A \subseteq \mathbb{N}$ such that, for every $\ell \in A$, $V(1^\ell, Y_\ell) = 1$ for some Y_ℓ of length $2^{\Theta(\ell)}$, and moreover, Y_ℓ has circuit complexity larger than $S(\ell)$. Thus we can solve CAPP infinitely often on circuits of size $s(n)$ if we succeed in guessing a string Y_ℓ such that $V(1^\ell, Y_\ell) = 1$, $\ell \in A$, and $S(\ell) \geq s(n)^\lambda$. For this to happen, since $S(\cdot)$ is increasing it is enough to run the CAPP routine just described on the first input 1^ℓ such that $S(\ell) \geq s(n)^\lambda$, which gives $\ell(n) \leq S^{-1}(s(n)^\lambda)$. The value $\ell(n)$ can be found by binary search. This nondeterministic algorithm succeeds infinitely often, using that $S^{-1}(s(n)^\lambda)$ is surjective over $\mathbb{N}$ as a function in n. Finally, it has complexity at most $\mathsf{poly}(n, |Y|) = \mathsf{poly}(n, 2^{\Theta(\ell)}) = \mathsf{poly}(n, 2^{S^{-1}(s(n)^\lambda)})$. □

Before describing the main result of this section and its proof, we need one last ingredient.

Theorem 11 (A parameterized version of "$\mathsf{EXP} \subseteq \mathsf{P}/\mathsf{poly}$ implies $\mathsf{EXP} = \mathsf{MA}$" [14])**.**
Let $g(n) \geq 2^n$ and $s(n) \geq n$ be time-constructible functions. There is a fixed $c \geq 1$ for which the following holds. If $\mathsf{TIME}[2^{O(n)}] \subseteq \mathsf{SIZE}[s(n)]$, then $\mathsf{TIME}[g(n)] \subseteq \mathsf{MATIME}[s(c \cdot \log g(n))^c]$.

The following extension of Theorem 5 holds.

Theorem 12 (Sub-third-exponential lower bounds against ACC^0)**.**
$(\mathsf{NE} \cap \mathsf{coNE})/1$ *does not have ACC^0 circuits of sub-third-exponential size.*

Proof. The proof is by contradiction. Suppose that $(\mathsf{NE} \cap \mathsf{coNE})/1 \subseteq \mathsf{ACC}^0[t(n)]$, where $t(n)$ is sub-third-exponential. This assumption implies in particular that:

$$\mathsf{E} = \mathsf{DTIME}[2^{O(n)}] \subseteq \mathsf{ACC}^0[t(n)] \subseteq \mathsf{SIZE}[t(n)^{O(1)}].$$

By Theorem 11, taking $g(n) = 2^{t(n)^2}$ and $s(n) = t(n)^{O(1)}$,

$$\mathsf{DTIME}[2^{t(n)^2}] \subseteq \mathsf{MATIME}[t(t(n)^{O(1)})^{O(1)}].$$

Since a deterministic time complexity class is closed under complementation,

$$\mathsf{DTIME}[2^{t(n)^2}] \subseteq \mathsf{coMATIME}[t(t(n)^{O(1)})^{O(1)}].$$

Consequently, for an arbitrary language $L \in \mathsf{DTIME}[2^{t(n)^2}]$, both L and its complement $\overline{L}$ can be decided by Merlin-Arthur protocols of complexity at most $t(t(n)^{O(1)})^{O(1)}$.

We next design a nondeterministic algorithm for L, by derandomizing the Merlin-Arthur protocol via Theorem 10. We take a circuit $C(x, y, z)$ of size $s(n) = t(t(n)^{O(1)})^{O(1)}$ encoding the predicate of the Merlin-Arthur game for L. That is, C takes input x, Merlin's string y, and Arthur's string z, where x has length n, and y, z both have length $\leq s(n)$. Our nondeterministic algorithm for L first guesses Merlin's string y, then simulates Arthur's string z using the pseudorandom generator for $C(x, y, \cdot)$. In order to apply Theorem 10, we choose $S(n) = s(n)^{\lambda}$. Using our assumptions, E has ACC^0 circuits of size at most $t(n)$, $2^{O(S^{-1}(s(n)^{\lambda}))} = 2^{O(n)}$, and $t(10 \cdot S(n) \log S(n)) \leq 2^{n^{o(1)}}$, since $t(n)$ is sub-third-exponential. Therefore, CAPP on circuits of size $s(n)$ and over n input variables is in $\mathsf{NE} = \mathsf{NTIME}[2^{O(n)}]$ for infinitely many values of n. Let $A \subseteq \mathbb{N}$ be the set of such input lengths. In particular, L (and also its complement $\overline{L}$) can be simulated infinitely often in $\mathsf{NE} \cap \mathsf{coNE}$. We can define two nondeterministic Turing machines M and M' which take (the same) one bit of advice (indicating whether an input length is in A) such that M accepts L and M' accepts $\overline{L}$ on every input length $n \in A$. Thus, given a language $L \in \mathsf{TIME}[2^{t(n)^2}]$, there is a language $L^{\star} \in (\mathsf{NE} \cap \mathsf{coNE})/1$ which agrees with L on infinitely many input lengths. That is,

$$\mathsf{DTIME}[2^{t(n)^2}] \subseteq \mathsf{i.o.}((\mathsf{NE} \cap \mathsf{coNE})/1).$$

Assuming $(\mathsf{NE} \cap \mathsf{coNE})/1$ has circuits of size $t(n)$ on every large enough input length, we immediately obtain from the preceding class inclusion that

$$\mathsf{DTIME}[2^{t(n)^2}] \subseteq \mathsf{i.o.SIZE}[t(n)].$$

However, this is a contradiction by a standard diagonalization argument. Indeed, we can define a language $L_{\mathsf{hard}} \in \mathsf{TIME}[2^{t(n)^2}]$ which does not have circuits of size $t(n)$ on every large enough input length n. That is, for every n, we determine the lexicographically first boolean function f_n which does not have circuits of size $t(n)$ by enumerating all circuits of size at most $t(n)$ and listing their truth tables in time $O(2^{t(n)^2})$, and let $L_{\mathsf{hard}}(x) \overset{\text{def}}{=} f_{|x|}(x)$. This can be done since by our assumptions $t(n)$ is constructive and $t(n) \ll 2^n/n$, thus a hard truth table exists for every large enough n. This completes the proof of Theorem 12. $\square$

In Section 6, we will need a slightly more technical formulation of the ACC^0 lower bound.

Lemma 1 (A consequence of the proof of Theorem 12)**.**
There exists a constant $k \geq 1$ for which at least one of the following separations hold:

$$\mathsf{E} \not\subseteq \mathsf{ACC}^0[n^{\log n}] \quad \textit{or} \quad \mathsf{NTIME}[n^{(\log n)^k}]/1 \cap \mathsf{coNTIME}[n^{(\log n)^k}]/1 \not\subseteq \mathsf{ACC}^0(n^{\log n}).$$

Also, in case the second separation holds, it is witnessed by a language $L \in \mathsf{NTIME}[n^{(\log n)^k}]/1 \cap \mathsf{coNTIME}[n^{(\log n)^k}]/1$ that can be decided by non-deterministic Turing machines using the same advice sequence $\alpha(n)\colon \mathbb{N} \to \{0,1\}$.

Proof Sketch. This is implicit in the proof of Theorem 12, under the particular choice of $t(n) = n^{\log n}$, and consequent optimization of the definition of $S(n)$. It is enough to observe that the negation of the statement of Lemma 1 is sufficient to obtain a contradiction. The existence of the advice sequence $\alpha(n)$ is immediate from the proof. (It is also possible to check this by closely following the argument and parameters in [27].) $\square$

4 Average-Case Lower Bounds against ACC^0

We start off by showing a $(1/2 - 1/n^{\Omega(1)})$-hardness result for E^{NP}.

Theorem 13 (An average-case lower bound for E^{NP})**.** *For every $d \geq 1$ and $m \geq 1$, there is a $\gamma > 0$ and a language in E^{NP} that is $(1/2 - 1/n^{\gamma})$-hard for nonuniform $\mathsf{AC}^0[m]$ circuits of depth d and size 2^{n^γ}.*

Proof. Given a sufficiently large $d \geq 1$ and a fixed modulo m, let $L^d \in \mathsf{E}^{\mathsf{NP}}$ be the language guaranteed to exist by Theorem 4, and $\delta = \delta(d, m) > 0$ be the corresponding constant. In other words, L^d is not computed by $\mathsf{AC}^0[m]$ circuits of depth d and size 2^{n^δ} for infinitely many values of n. For a function $\epsilon' = \epsilon'(M') \geq \exp(-\Theta(\sqrt{\log M'}))$ to be fixed later in the proof, and d' sufficiently large (but smaller than d), let $\{C_{M'}\}$ be the sequence of explicit error-correcting codes provided by Corollary 1, where each $C_{M'} \colon \{0,1\}^{M'} \to \{0,1\}^N$, and $N(M') = M'^c$ for a fixed positive integer $c \geq 1$. Consider a new language $L^\star$ that depends on L^d and on $\{C_{M'}\}$, defined as follows. Given $x \in \{0,1\}^n$, if n is not of the form cm' for some $m' \in \mathbb{N}$, then x is not in $L^\star$. Otherwise, let $T \in \{0,1\}^{2^{m'}}$ be the truth-table of L^d on m'-bit inputs, and consider the codeword $C_{M'}(T) \in \{0,1\}^N$, where $M' = 2^{m'}$ and $N = M'^c = 2^{cm'} = 2^n$. Then $x \in L^\star$ if and only if the entry of $C_{M'}(T)$ indexed by x is 1. This completes the description of $L^\star$.

Given that $L^d \in \mathsf{E}^{\mathsf{NP}}$ and $C_{M'}$ can be computed in deterministic time $\mathsf{poly}(M')$, we can compute $L^\star$ in E^{NP} as follows. Let x be an input of length N, on which we wish to solve $L^\star$. First, check if $N = M'^c$ for some integer M'. If not, output 0. Otherwise, compute the truth table T of L^d on input length M' by running the E^{NP} machine for L^d on every possible input of length M'. Then compute $C_{M'}(T)$ and output the x'th bit of that string. The computation of T can be done in E^{NP} as it involves at most 2^N runs of an E^{NP} machine on inputs of length $\leq N$, and the computation of $C_{M'}(T)$ can be done in time $2^{O(N)}$ just using the efficiency guarantee for $C_{M'}$. Hence the procedure described above can be implemented in E^{NP}.

Now we show that $L^\star$ has the claimed average-case hardness. For $n = cm'$ and $M' = 2^{m'}$ as above, we set $\epsilon'(M') \overset{\text{def}}{=} 1/n^{2\gamma} \gg \exp(-\Theta(\sqrt{\log M'}))$, where $0 < \gamma < \delta/2$ is a sufficiently small constant. We claim that $L^\star$ cannot be computed with advantage larger than $1/n^\gamma$ on infinitely many input lengths by $AC^0[m]$ circuits of depth $\leq d$ and size $\leq 2^{n^\gamma}$. This follows by the properties of the code $C_{M'}$ and the connection to hardness amplification. Indeed, if for all large n of the form cm' the boolean function computed by $L^\star_n$ could be approximated by such circuits, by hardcoding their descriptions into the AC^0 local decoders provided by Corollary 1 it would follow that for all large n the language L^d is (worst-case) computable by $\mathsf{AC}^0[m]$ circuits of depth $\leq d$ and size $\leq 2^{n^\delta}$, a contradiction. (This last step crucially uses that γ is sufficiently small compared to the other parameters, and the size bound in Corollary 1.) □

Next, we address the more difficult problem of showing an average-case lower bound for NEXP. We first establish a lower bound for $(\mathsf{NE} \cap \mathsf{coNE})/1$, and then show how to remove the advice.

Lemma 2. *$(\mathsf{NE} \cap \mathsf{coNE})/1$ is $(1/2 - 1/\log(t(n)))$-hard for ACC^0 circuits of size $t(n)$, for any (time-constructible) sub-third-exponential function $t(n)$.*

Proof. The argument follows the same high-level approach of Theorem 13, so we use the same notation and only describe the relevant differences. By Theorem 12, there is a language $L \in (\mathsf{NE} \cap \mathsf{coNE})/1$ that is not computable by ACC^0 circuits of sub-third-exponential size $t(n)$. Similarly, we define a language $L^\star$ obtained from L and the locally-list-decodable codes provided by Corollary 1. We need to make sure the new language is still computable in $(\mathsf{NE} \cap \mathsf{coNE})/1$, and explain the choice of parameters in the construction.

Since $L \in (\mathsf{NE} \cap \mathsf{coNE})/1$, there is a strong non-deterministic Turing machine (SNTM) S with one bit of advice computing L. Let the advice sequence for M be $\alpha(\cdot)$, where $|\alpha(n)| = 1$ for all $n \in \mathbb{N}$. We define an SNTM S' with one bit of advice computing $L^\star$. S' acts as follows on input x of length N. It checks if $N = M'^c$ for some integer M'. If not, it rejects. If yes, it simulates S with advice $\alpha(M')$ on each input of length M'. The advice $\alpha(M')$ is the advice bit for S' - note that N completely determines M' and hence $\alpha(M')$. If any of these simulations outputs '?', it outputs '?' and halts. If all of these simulations output non-'?' values, S' uses the results of its simulations of S to compute the truth table T of L on input length M', and applies the mapping $C_{M'}$ to this string. It then outputs the bit with index x of the resulting string.

We need to show that S' is an SNTM with one bit of advice deciding $L^\star$ correctly in time $2^{O(N)}$. By definition of S', and using the fact that S is an SNTM with one bit of advice, we have that whenever S' computes a string T, this is the correct truth table of L on inputs of length M', if S' uses advice $\beta(N) = \alpha(M')$. Moreover, this happens on at least one computation path of S, using the fact that S' is an SNTM with one bit of advice. On any such computation path, the correct value $L^\star(x)$ is output, as S' is completely deterministic after computing T, and using the definition of $L^\star$. The time taken by S' is $2^{O(n)}$, as it simulates S on inputs of length $\leq N$ at most 2^N times, and using the efficiency guarantee on $C_{M'}$.

Finally, we sketch the choice of parameters in the hardness amplification, which correspond to the parameters in the construction of $L^\star$ via the error-correcting code provided by Corollary 1. Following the notation in the proof of Theorem 13, we let $\epsilon(M')$ be of order $1/\log(t(\beta n)^\beta)$, where $\beta > 0$ is sufficiently small. Under this definition, observe that the circuit complexity overhead coming from the decoder in the analysis of the average-case hardness of $L^\star$ is at most $\mathsf{poly}(n, \exp(1/\epsilon')) \leq \mathsf{poly}(n, \exp(\log t(\beta n)^\beta)) \leq t(n)^\gamma$, for a fixed but arbitrarily small $\gamma > 0$ that depends on β. This implies that $L^\star$ is $1/\log t(\Omega(n))^{\Omega(1)}$-hard against circuits of size $t(n)^{\Omega(1)}$. Since our original sub-third-exponential function $t(n)$ was arbitrary and after composition with polynomials a function remains in this class, the proof is complete. □

We give a generic way to eliminate advice from the upper bound for average-case hardness results. The specific case considered below involves 1 bit of advice, but a similar proof works for up to $o(\log(n))$ bits of advice.

Lemma 3. *If* $\mathsf{NE}/1$ *is* $(1/2 - \epsilon(n))$*-hard for* $\mathcal{C}$ *circuits of size* $s(n)$*, then* NE *is* $(1/2 - \epsilon(\lfloor n/2 \rfloor))$*-hard for* $\mathcal{C}$ *circuits of size* $s(\lfloor n/2 \rfloor)$*.*

Proof. Let L be a language in $\mathsf{NE}/1$ which is $(1/2 - \epsilon)$-hard for $\mathcal{C}$ circuits of size $s(n)$. Suppose L is decided by a NTM M running in nondeterministic time $2^{O(n)}$ and taking advice bits $\{b_n\}$, where $|b_n| = 1$. In other words, for every string x, we have $L(x) = M(x, b_{|x|})$.

Define a new language L' as follows. We divide the input string z in the middle, and denote it by xy, where either $|y| = |x|$ (when $|z|$ is even) or $|y| = |x| + 1$ (when $|z|$ is odd). Then we decide by running M on the first half x, using an advice bit which depends only on the length of y. More precisely, we let

$$L'(xy) \overset{\text{def}}{=} \begin{cases} M(x, 0), & \text{if } |y| = |x|; \\ M(x, 1), & \text{if } |y| = |x| + 1. \end{cases}$$

Obviously, L' is in NE by simulating M.

We show that if L_n is hard to approximate, then either L'_{2n} or L'_{2n+1} is also hard to approximate. For contradiction, suppose that both L'_{2n} and L'_{2n+1} can be computed correctly on more than a $1/2 + \epsilon$ fraction of inputs by circuits of size s. If the advice bit $b_n = 0$, let C_0 be a circuit of size s such that $\mathbf{Pr}_{xy}[L'_{2n}(xy) = C_0(xy)] > 1/2 + \epsilon$, where x and y are both chosen independently and uniformly at random from $\{0,1\}^n$. By an averaging argument, there is a specific $y^\star$ such that by fixing $y = y^\star$, $\mathbf{Pr}_x[L'_{2n}(xy^\star) = C_0(xy^\star)] > 1/2 + \epsilon$. Note also that, since $b_n = 0$, we have that for all x of length n, $L'_{2n}(xy^\star) = M(x, 0) = L_n(x)$. Thus $\mathbf{Pr}_x[L_n(x) = C_0(xy^\star)] > 1/2 + \epsilon$. That is, we can use C_0 to approximate L_n by fixing the second half of the inputs to $y^\star$. In the other case where the advice bit $b_n = 1$, we can use the approximate circuit for L'_{2n+1} to approximate L_n in the same way. As a consequence, if L_n is $(1/2 - \epsilon(n))$-hard for $\mathcal{C}$ circuits of size s, then either L'_{2n} or L'_{2n+1} is also $(1/2 - \epsilon(n))$-hard for $\mathcal{C}$ circuits of size s.

Finally, since there are infinitely many input lengths n such that L_n is $(1/2 - \epsilon(n))$-hard for $\mathcal{C}$ circuits of size $s(n)$, there are also infinitely many input lengths n such that L'_n is $(1/2 - \epsilon(\lfloor n/2 \rfloor))$-hard for $\mathcal{C}$ circuits of size $s(\lfloor n/2 \rfloor)$. This completes the proof. □

Remark 1 (Preserving the hardness in Lemma 3)**.** *Observe that to maintain the infinitely often average-case hardness of the language L in the proof of Lemma 3, we have used in a crucial way the input* length *instead of an input bit in the definition of L'. This is because we cannot claim hardness when the advice bit is not set to b_n.*

By combining the previous two lemmas, we get the following strengthened version of Theorem 1.

Theorem 14 (An average-case lower bound for NE against sub-third-exponential size ACC^0)**.** NE *is $(1/2 - 1/\log t(n))$-hard for ACC^0 circuits of size $t(n)$ when $t(n)$ is time-constructible and sub-third-exponential.*

Remark 2 (On stronger average-case lower bounds.)**.** *In Lemma 2 and Theorem 14, we were only able to prove $(1/2 - \epsilon(n))$-hardness for a parameter $\epsilon(n)$* logarithmic *in the size bound. The bottleneck for improving the bounds on $\epsilon(n)$ comes from computing majority in the local list decoder, as in Corollary 1. If one can show that the majority function on $\mathsf{poly}(1/\delta(n))$ bits is computable by ACC^0 circuits of size $s(n)$, where $s(n)$ is at most sub-third-exponential, then Lemma 2 and Theorem 14 would establish $(1/2 - \delta(n))$-hardness of $(\mathsf{NE} \cap \mathsf{coNE})/1$ and of NE for ACC^0 circuits of sub-third-exponential size, respectively.*

5 A Pseudorandom Generator against $\mathsf{AC}^0[6]$

The following result is shown in [6] by using the *resamplability* of a certain complete problem in L, and we refer to their paper for more details.

Theorem 15 ([6])**.** *Suppose $\mathsf{L} \not\subseteq \mathsf{AC}^0[6]$. Then, for every depth $d \geq 1$ and any $\beta > 0$, there is a generator $G_n \colon \{0,1\}^{n-n^{1-\beta}} \to \{0,1\}^n$ computable in polynomial time such that, for infinitely many values of n, for every $\mathsf{AC}^0[6]$-circuit C of depth $\leq d$ and size $\leq n^d$,*

$$\left| \mathbf{Pr}_{s \in \{0,1\}^{n-n^\beta}}[C(G_n(s)) = 1] - \mathbf{Pr}_{x \in \{0,1\}^n}[C(x) = 1] \right| \leq o(1).$$

We now use Theorem 15 to get a pseudo-random generator against $\mathsf{AC}^0_[6]$ circuits of depth d computable in $(\mathsf{NE} \cap \mathsf{coNE})/1$.

Proof of Theorem 2. We use a win-win argument. Since $(\mathsf{NE} \cap \mathsf{coNE})/1 \not\subseteq \mathsf{ACC}^0$ by Theorem 5 and $\mathsf{AC}^0[6] \subseteq \mathsf{ACC}^0$, we have either $(\mathsf{NE} \cap \mathsf{coNE})/1 \not\subseteq \mathsf{L}/\mathsf{poly}$, or $\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{AC}^0[6]$.

If $(\mathsf{NE} \cap \mathsf{coNE})/1 \not\subseteq \mathsf{L}/\mathsf{poly}$, since $\mathsf{NC}^1 \subseteq \mathsf{L}/\mathsf{poly}$, we get $(\mathsf{NE} \cap \mathsf{coNE})/1 \not\subseteq \mathsf{NC}^1$. By using a worst-case to average-case hardness amplification against NC^1 [23] and the hardness-to-pseudorandomness transformation of [15], this implies a PRG with (a much shorter) seed length n^{β} against NC^1 circuits computable in $(\mathsf{NE} \cap \mathsf{coNE})/1$, and hence also a PRG with the same seed length against $\mathsf{AC}^0[6]$ circuits (since $\mathsf{AC}^0[6] \subseteq \mathsf{NC}^1$). It is crucial in this argument that the hardness amplification step and the construction in [15], when applied to a Boolean function f, work on an input length by input length basis. This allows us to maintain a single bit of advice.

If $\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{AC}^0[6]$, we first get $\mathsf{L} \not\subseteq \mathsf{AC}^0[6]$. This can be proved by contraposition. Suppose $\mathsf{L} \subseteq \mathsf{AC}^0[6]$. Let Q be a language in L/poly. Then there is a log-space TM M and a sequence $\{a_n\}$ of advice strings a_n of polynomial length such that $x \in Q$ if and only if $M(x, a_{|x|}) = 1$. Since the language decided by M is in $\mathsf{L} \subseteq \mathsf{AC}^0[6]$, there is an $\mathsf{AC}^0[6]$ circuit C_n of polynomial size such that $C_n(x, y) = M(x, y)$ for each input $x \in \{0,1\}^n$ and string y of length $|a_n|$. Thus, for each n, we get an $\mathsf{AC}^0[6]$ circuit $C'_n(x) \overset{\text{def}}{=} C_n(x, a_{|x|})$ of polynomial size that decides Q on inputs of length n, after fixing the second part of the input in $C_n(\cdot, \cdot)$ with the correct advice string. This shows that $\mathsf{L}/\mathsf{poly} \subseteq \mathsf{AC}^0[6]$, and completes the proof of our claim.

Now, since $\mathsf{L} \not\subseteq \mathsf{AC}^0[6]$, we can use Theorem 15 to get a pseudo-random generator with the desired properties. □

6 Lower Bounds from Learning Algorithms for ACC^0

This section is dedicated to the proof of Theorem 3 and its implications. We refer the reader to [16] for more background in learning theory, since that work shares a similar notation.

Recall that $\mathsf{ACC}^0_{d,m}(s(n))$ is the class of boolean functions computable by depth-d ACC^0 circuits over a fixed modulus m and of size $\leq s(n)$. We will need the following results.

Lemma 4 (A variant of the Speedup Lemma [16])**.** *Fix a modulus $m \geq 2$, and time-constructible size bounds $\alpha(n) \geq \beta(n) \geq n$. Suppose that, for every depth $d \geq 1$, the class $\mathsf{ACC}^0_{d,m}(\alpha(n))$ is non-trivially learnable. Then, for every $d \geq 1$, the class $\mathsf{ACC}^0_{d,m}(\beta(n))$ can be learned in time at most $\mathsf{poly}(n, 2^{\alpha^{-1}(\mathsf{poly}(\beta(\mathsf{poly}(n))))})$.*

Proof Sketch. This is a straightforward generalization of the argument presented in the proof of Lemma 24 from [16], instantiated in the particular case of ACC^0 circuits. □

Lemma 5 (Conditional connection between learning and lower bounds)**.** *There is a PSPACE-hard language $L^{\star} \in \mathsf{DSPACE}[O(n)]$ for which the following holds. Let $s(n) \geq n$ and $T(n) \geq n$ be constructible functions. If $\mathsf{ACC}^0_{d,m}(s(n))$ is learnable in time $T(n)$, then at least one of the following relations is true:*

- $L^{\star} \notin \mathsf{ACC}^0_{d,m}(s(n))$*; or*
- $L^{\star} \in \mathsf{BPTIME}[\mathsf{poly}(T(n))]$.

Proof Sketch. This is a particular case of a result from [24, 13], and we refer to their work for more details (see also [16]). □

We are now ready to give the proof of Theorem 3. It employs the main ideas appearing in the argument from [16], but by relying on the results from Section 3 instead of [26], we are able to achieve a significant strengthening of Proposition 1.

Proof of Theorem 3. Let $L^{\star}$ be the language from Lemma 5. First, if $L^{\star} \notin \mathsf{ACC}^0_{d,m}(n^{\log n})$ for each choice of $d, m \geq 2$, we are done, since $L^{\star} \in \mathsf{DSPACE}[O(n)] \subseteq \mathsf{DTIME}[2^{O(n)}] \subseteq \mathsf{RTIME}[2^{O(n)}] \cap \mathsf{coRTIME}[2^{O(n)}]$, and this class is obviously contained in both $\mathsf{RTIME}[2^{O(n)}]$ and in $\mathsf{ZPTIME}[2^{O(n)}]/1$.

Otherwise, there are integers $d^\star, m^\star \geq 2$ such that $L^\star \in \mathsf{ACC}^0_{d^\star,m^\star}(n^{\log n})$. By assumption, for every $d, c \geq 1$, the class $\mathsf{ACC}^0_{d,m^\star}(n^{(\log n)^c})$ can be learned in non-trivial time. It follows from Lemma 4 that for every $\ell \geq 1$, $\mathsf{ACC}^0_{d^\star,m^\star}(n^{\log n})$ can be learned in time at most $T(n)$, where

$$T(n) \;\leq\; \mathsf{poly}(n, \exp(\alpha^{-1}(n^{O(\log n)}))) \;\text{ and }\; \alpha(n) \;=\; n^{(\log n)^\ell},$$

for sufficiently large n. Since $L^\star \in \mathsf{ACC}^0_{d^\star,m^\star}(n^{\log n})$, we get from Lemma 5 and the PSPACE-hardness of $\mathsf{L}^\star$ that $\mathsf{PSPACE} \subseteq \mathsf{BPTIME}[\mathsf{poly}(n, T(\mathsf{poly}(n)))]$. In particular, we have that $\mathsf{NP} \subseteq \mathsf{BPTIME}[\mathsf{poly}(n, T(\mathsf{poly}(n)))]$.

By a simple search-to-decision reduction, it follows that $\mathsf{NP} \subseteq \mathsf{RTIME}[\mathsf{poly}(n, T(\mathsf{poly}(n)))]$, i.e., we can assume the randomized algorithms deciding languages in NP to have one-sided error. Let $k \in \mathbb{N}$ be the constant from Lemma 1. Using a standard translation argument and the upper bound on $T(\cdot)$, it follows that

$$\mathsf{NTIME}[n^{(\log n)^k}] \;\subseteq\; \mathsf{RTIME}[\mathsf{poly}(n, T(n^{(\log n)^{k'}}))] \;\subseteq\; \mathsf{RTIME}[2^n], \tag{1}$$

where k' is a sufficiently large constant. If $\mathsf{E} \nsubseteq \mathsf{ACC}^0(n^{\log n})$ in the statement of Lemma 1, we are done, as there is nothing left to prove in Theorem 3. Otherwise, we can assume (in particular) that

$$\mathsf{NTIME}[n^{(\log n)^k}] \;\nsubseteq\; \mathsf{ACC}^0(n^{\log n}). \tag{2}$$

Let L be the language witnessing this separation. Since $L \notin \mathsf{ACC}^0(n^{\log n})$, in order to complete the proof of the first lower bound in the statement of Theorem 3 it is sufficient to establish that $L \in \mathsf{RTIME}[2^{O(n)}]$. But this follows immediately from Equation 1.

We proceed now with the proof of the $\mathsf{ZPTIME}[2^{O(n)}]/1$ lower bound, which is a bit more technical, and requires the stronger separation

$$\mathsf{NTIME}[n^{(\log n)^k}]/1 \cap \mathsf{coNTIME}[n^{(\log n)^k}]/1 \;\nsubseteq\; \mathsf{ACC}^0(n^{\log n}). \tag{3}$$

in Lemma 1. Arguing as before, we obtain Equations 1 and 3. This time, let L be a language witnessing this stronger separation. Furthermore, assume it is decided by non-deterministic Turing machines using the same advice sequence $\alpha(n) \colon \mathbb{N} \to \{0,1\}$ (Lemma 1). We need the following result.

Lemma 6. $L \in \mathsf{RTIME}[2^n]/1 \cap \mathsf{coRTIME}[2^n]/1$. *In both cases, the advice sequence can be taken as* $\alpha(\cdot)$.

Proof of Lemma 6. Let M be a nondeterministic machine that decides L using advice sequence $\alpha(\cdot)$ in time at most $O(n^{(\log n)^k})$. In other words, $L(x) = M(x, \alpha(|x|))$ on every input x. Consider the language $L' \overset{\text{def}}{=} \{yb \mid y \in \{0,1\}^\star, b \in \{0,1\}, \text{ and } M(y,b) = 1\}$. Clearly, $L' \in \mathsf{NTIME}[n^{(\log n)^k}]$, and by Equation 1, we also have $L' \in \mathsf{RTIME}[2^n]$. By fixing the appropriate input bit of a corresponding randomized machine that decides L', it follows that $L \in \mathsf{RTIME}[2^n]/1$ using advice sequence $\alpha(\cdot)$. This provides the first inclusion in Lemma 6.

Using Equation 1 and complementation, we get that $\mathsf{coNTIME}[n^{(\log n)^k}] \subseteq \mathsf{coRTIME}[2^n]$. Now by assumption we also have $L \in \mathsf{coNTIME}[n^{(\log n)^k}]/1$, and via a completely analogous argument, we conclude that $L \in \mathsf{coRTIME}[2^n]/1$. Note that the advice sequence is also given by $\alpha(\cdot)$. This finishes the proof of Lemma 6. □

Continuing with the proof of Theorem 3, it follows from Lemma 6 by a standard composition of $\mathsf{RTIME}[\cdot]$ and $\mathsf{coRTIME}[\cdot]$ computations that L can be computed in $\mathsf{ZPTIME}[2^n]/1$. Note that the single bit of advice is given by $\alpha(n)$, and that the corresponding randomized algorithm respects the zero-error promise under the correct advice bit. Since we started with a language $L \notin \mathsf{ACC}^0[n^{\log n}]$, the proof is complete. $\square$

Acknowledgements. We would like to thank Marco Carmosino for posing the question of proving average-case hardness against ACC^0, and for useful discussions. Thanks to Ryan Williams for comments on an earlier version of this paper. This work was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agrement no. 615075.

References

[1] TCS Stack Exchange: "How powerful is ACC^0 circuit class in average case?". (Online version accessed on 27/09/2017). URL: https://cstheory.stackexchange.com/q/37232. 2

[2] M. Ajtai. Σ^1_1-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983. URL: http://dx.doi.org/10.1016/0168-0072(83)90038-6, doi:10.1016/0168-0072(83)90038-6. 1

[3] Sanjeev Arora and Boaz Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, Cambridge, 2009. 6

[4] Joshua Buresh-Oppenheim, Valentine Kabanets, and Rahul Santhanam. Uniform hardness amplification in np via monotone codes. *Electronic Colloquium on Computational Complexity* (ECCC), page 154, 2006. URL: https://eccc.weizmann.ac.il/eccc-reports/2006/TR06-154/. 3

[5] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016. URL: https://doi.org/10.4230/LIPIcs.CCC.2016.10, doi:10.4230/LIPIcs.CCC.2016.10. 5

[6] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013. URL: http://dx.doi.org/10.4086/toc.2013.v009a026, doi:10.4086/toc.2013.v009a026. 3, 14

[7] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Systems Theory*, 17(1):13–27, 1984. URL: http://dx.doi.org/10.1007/BF01744431, doi:10.1007/BF01744431. 1

[8] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Symposium on Theory of Computing* (STOC), pages 440–449, 2007. URL: http://doi.acm.org/10.1145/1250790.1250855, doi:10.1145/1250790.1250855. 3, 8

[9] Dan Gutfreund and Guy N. Rothblum. The complexity of local list decoding. In *International Workshop on Approximation, Randomization and Combinatorial Optimization* (RANDOM-APPROX), pages 455–468, 2008. URL: http://dx.doi.org/10.1007/978-3-540-85363-3_36, doi:10.1007/978-3-540-85363-3_36. 3, 7, 8

[10] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Symposium on Theory of Computing* (STOC), pages 6–20, 1986. URL: http://doi.acm.org/10.1145/12130.12132, doi:10.1145/12130.12132. 1

[11] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. *Electronic Colloquium on Computational Complexity* (ECCC), page 023, 2017. URL: https://eccc.weizmann.ac.il/report/2017/023/. 5

[12] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 749–760, 2015. URL: https://doi.org/10.1007/978-3-662-47672-7_61, doi:10.1007/978-3-662-47672-7_61. 9

[13] Adam R. Klivans, Pravesh Kothari, and Igor C. Oliveira. Constructing hard functions using learning algorithms. In *Conference on Computational Complexity* (CCC), pages 86–97, 2013. URL: https://doi.org/10.1109/CCC.2013.18, doi:10.1109/CCC.2013.18. 15

[14] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *International Computing and Combinatorics Conference* (COCOON), pages 210–220, 1999. URL: http://dx.doi.org/10.1007/3-540-48686-0_21, doi:10.1007/3-540-48686-0_21. 10

[15] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. URL: https://doi.org/10.1016/S0022-0000(05)80043-1, doi:10.1016/S0022-0000(05)80043-1. 3, 15

[16] Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017. URL: https://doi.org/10.4230/LIPIcs.CCC.2017.18, doi:10.4230/LIPIcs.CCC.2017.18. 4, 5, 7, 15

[17] Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017. URL: http://doi.acm.org/10.1145/3055399.3055500, doi:10.1145/3055399.3055500. 1, 4

[18] Alexander A. Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987. 1

[19] Rocco Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 1–23, 2017. 5

[20] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. URL: https://doi.org/10.1137/080735096, doi:10.1137/080735096. 3

[21] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Symposium on Theory of Computing* (STOC), pages 77–82, 1987. URL: http://doi.acm.org/10.1145/28395.28404, doi:10.1145/28395.28404. 1

[22] Srikanth Srinivasan. On improved degree lower bounds for polynomial approximation. In *Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS),

pages 201–212, 2013. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2013.201, doi:10.4230/LIPIcs.FSTTCS.2013.201. 4

[23] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. URL: https://doi.org/10.1006/jcss.2000.1730, doi:10.1006/jcss.2000.1730. 3, 8, 14

[24] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. URL: https://doi.org/10.1007/s00037-007-0233-x, doi:10.1007/s00037-007-0233-x. 15

[25] Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003. URL: http://dx.doi.org/10.1016/S0022-0000(03)00046-1, doi:10.1016/S0022-0000(03)00046-1. 7

[26] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. URL: http://doi.acm.org/10.1145/2559903, doi:10.1145/2559903. 1, 2, 3, 4, 5, 7, 15

[27] Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016. URL: http://dx.doi.org/10.1137/130938219, doi:10.1137/130938219. 2, 3, 4, 5, 7, 9, 10, 11

[28] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *Symposium on Foundations of Computer Science* (FOCS), pages 1–10, 1985. URL: https://doi.org/10.1109/SFCS.1985.49, doi:10.1109/SFCS.1985.49. 1

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