An Average-Case Lower Bound Against ACC^0

Ruiwen Chen, Igor C. Oliveira(✉), and Rahul Santhanam

Department of Computer Science, University of Oxford, Oxford, UK
{ruiwen.chen,igor.carboni.oliveira,rahul.santhanam}@cs.ox.ac.uk

Abstract. In a seminal work, Williams [22] showed that NEXP (nondeterministic 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 and a function $\varepsilon(n) = 1/\log(n)^{\omega(1)}$ 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 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 [15].

Keywords: Circuit lower bounds · Average-case complexity
Pseudorandomness · Learning and natural properties

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) [2,8,11,24] and $\mathsf{AC}^0[p]$ (AC^0 circuits extended with MOD_p gates) for p prime [16,19]. 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 [22] proved a superpolynomial 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.

M. A. Bender et al. (Eds.): LATIN 2018, LNCS 10807, pp. 317–330, 2018.
https://doi.org/10.1007/978-3-319-77404-6_24

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 [22], or in closely related classes [23]. (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 [1].

2 Results and Techniques

Our main result addresses this second drawback of Williams' lower bound, and strengthen the main result from [22] 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 Sect. 3.

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 [22] 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 [22] 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 assumption 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 [18]. 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 et al. [21], hardness amplification procedures are known [9,10] 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 [23], where he shows that his lower bound [22] 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.

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

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 [13], 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 $\varepsilon(n) \leqslant 1/n^{\Omega(1)}$, which Theorem 1 fails to achieve.

More recent work of [7] studies how to derive pseudorandomness from average-case hardness in cases where the hardness is not strong enough to apply [13]. It is shown in [7] 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 [7], 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]

Theorem 2 (A pseudo-random generator against $\mathsf{AC}^0[6]$). *For every depth $d \geqslant 1$ 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 $\leqslant d$ and size $\leqslant n^d$. Then, for infinitely many values of n,*

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

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 (see Theorem 9 in Sect. 3.4). 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

[2] We stick to modulo 6 gates mostly for simplicity. Theorem 2 can be extended to any modulus m for which the results from [7] hold.

[3] In other words, the non-deterministic algorithm, when given the correct advice bit (that only depends on the input length parameter), outputs either "abort" of the correct string, and outputs the correct string in at least one computation path. We refer to Sect. 3.1 for more details.

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. [20]).

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 Sect. 1, and which we further explore now.

While substantially improving the explicitness in Williams' lower bounds [22,23] and in Theorem 1 remains a major challenge, [14] 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, [14] show implications for circuit lower bounds from learning algorithms that are similarly non-trivial in that they beat a brute force search 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 $\leqslant s(n)$. The following connection between learning algorithms and non-uniform lower bounds was established in [14].

Proposition 1 (REXP lower bounds from learning sub-exponential size ACC^0 circuits [14]). *If for every depth $d \geqslant 1$ and modulo $m \geqslant 1$ there is $\varepsilon > 0$ such that $\mathsf{ACC}^0_{d,m}(2^{n^\varepsilon})$ can be learned in non-trivial time, then $\mathsf{REXP} \not\subseteq \mathsf{ACC}^0$.*

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 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 [14], 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.

[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 [15].

[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 [17].

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 $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$. This lower bound is employed as a *black-box* in the argument from [14], and in Sect. 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 [14], and the satisfiability framework of Williams [22,23]. 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 $d, m, c \geqslant 1$, the class $\mathsf{ACC}^0_{d,m}(n^{(\log n)^c})$ can be non-trivially learned. Then,*

$$\mathsf{RTIME}[2^{O(n)}] \not\subseteq \mathsf{ACC}^0(n^{\log n}) \quad \textit{and} \quad \mathsf{ZPTIME}[2^{O(n)}]/1 \not\subseteq \mathsf{ACC}^0(n^{\log n}).$$

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.

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 larger 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 [14], and also [12]). 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.

We sketch in the next section the proof of Theorem 1. Due to space limitations, we refer to the full version of the paper [6] for more details about our results.

3 Proof of Theorem 1

3.1 Notation for Complexity Classes and Circuit Classes

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 $\mathsf{ACC}^0 \stackrel{\text{def}}{=} \bigcup_m \mathsf{AC}^0[m]$ (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 $\leqslant d$ and size $\leqslant s$. 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, $\mathbf{Pr}_{x\in\{0,1\}^n}[L(x) = L'(x)] \leqslant 1-\gamma(n)$. Finally, a class Γ is $\gamma(n)$-hard for $\mathcal{C}$ if there is a language in Γ that is $\gamma(n)$-hard for $\mathcal{C}$.

3.2 Background on ACC^0 Lower Bounds

We recall the following ACC^0 circuit lower bounds.

Theorem 4 ([22]). *For every $d \geqslant 1$ and $m \geqslant 1$, 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 ([23]). *There is a language in $(\mathsf{NE} \cap \mathsf{coNE})/1$ that does not admit ACC^0 circuits of size $O(n^{\log n})$.*

In order to prove Theorem 1 and its extensions, we need a strengthening of Theorem 5. We use the following technical definitions. A function $f\colon \mathbb{N} \to \mathbb{N}$ is *sub-half-exponential* if for every fixed $k \geqslant 1$, $f(f(n^k)^k) \leqslant 2^{n^{o(1)}}$. Similarly, a function $g\colon \mathbb{N} \to \mathbb{N}$ is *sub-third-exponential* if for every fixed $k \geqslant 1$, $g(g(g(n^k)^k)^k) \leqslant 2^{n^{o(1)}}$. For instance, for a fixed integer $a \geqslant 1$, $g(n) \stackrel{\text{def}}{=} 2^{(\log n)^a}$ is sub-third-exponential.

By a more careful application of William's techniques, the following result can be established. We refer to the full version of the paper [6] for details.

Theorem 6 (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.*

3.3 Tools: Error Correcting Codes and Hardness Amplification

We follow part of the terminology from [10]. The proof of Theorem 1 requires certain correcting codes that admit a uniform encoding procedure, but whose decoding can be non-uniform.

Definition 1 (Local-list-decoding in error correcting codes). *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 $\Delta(C_M(x), y) \leqslant d$, 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 [21] between fully black-box hardness amplification and binary locally-list-decodable codes.

Theorem 7 (Connection between hardness amplification and local-list-decodable codes). *If there is a* $(1/2-\epsilon, L)$*-locally list decodable error-correcting 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 [9,10]). *For every* $\exp(-\Theta(\sqrt{\log M})) \leqslant \epsilon < 1/2$*, 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.*

Observe that it is possible to get an AC^0 decoder by a standard simulation of majority gates via large AC^0 circuits.

[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 (Limited hardness amplification via constant-depth circuits of bounded size). *For every parameter* $\exp(-\Theta(\sqrt{\log M})) \leqslant \epsilon < 1/2$ *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*.*

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

3.4 The Proof of Theorem 1 and Its Extensions

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

Theorem 9 (An average-case lower bound for E^{NP}). *For every* $d \geqslant 1$ *and* $m \geqslant 1$*, 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* $2^{n^{\gamma}}$*.*

Proof. Given a sufficiently large $d \geqslant 1$ 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 $2^{n^{\delta}}$ for infinitely many values of n. For a function $\epsilon' = \epsilon'(M') \geqslant \exp(-\Theta(\sqrt{\log M'}))$ 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 $c \geqslant 1$. 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 $\leqslant N$, 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 $\epsilon'(M') \stackrel{\text{def}}{=} 1/n^{2\gamma} \gg \exp(-\Theta(\sqrt{\log M'}))$, 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 $\leqslant d$ and size $\leqslant 2^{n^\gamma}$. 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 $\leqslant d$ and size $\leqslant 2^{n^\delta}$, 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 1. $(\mathsf{NE} \cap \mathsf{coNE})/1$ *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 9, so we use the same notation and only describe the relevant differences. By Theorem 6, 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 $\leqslant N$ 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 9, 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 $\mathsf{poly}(n, \exp(1/\epsilon')) \leqslant \mathsf{poly}(n, \exp(\log t(\beta n)^{\beta})) \leqslant t(n)^{\gamma}$, 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.

Lemma 2. *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

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

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.

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

Theorem 10 (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.*

Acknowledgements. We would like to thank Marco Carmosino for posing the question of proving average-case hardness against ACC^0, and for useful discussions. 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? https://cstheory.stackexchange.com/q/37232. Accessed 27 Sept 2017
2. Ajtai, M.: Σ_1^1-formulae on finite structures. Ann. Pure Appl. Logic **24**(1), 1–48 (1983). https://doi.org/10.1016/0168-0072(83)90038-6
3. Arora, S., Barak, B.: Complexity Theory: A Modern Approach. Cambridge University Press, Cambridge (2009)
4. Buresh-Oppenheim, J., Kabanets, V., Santhanam, R.: Uniform hardness amplification in NP via monotone codes. In: Electronic Colloquium on Computational Complexity (ECCC) TR06-154 (2006). https://eccc.weizmann.ac.il/eccc-reports/2006/TR06-154/
5. Carmosino, M.L., Impagliazzo, R., Kabanets, V., Kolokolova, A.: Learning algorithms from natural proofs. In: Conference on Computational Complexity (CCC), pp. 10:1–10:24 (2016). https://doi.org/10.4230/LIPIcs.CCC.2016.10
6. Chen, R., Oliveira, I.C., Santhanam, R.: An average-case lower bound against ACC^0. In: Electronic Colloquium on Computational Complexity (ECCC) TR17-173 (2017). https://eccc.weizmann.ac.il/report/2017/173/
7. Fefferman, B., Shaltiel, R., Umans, C., Viola, E.: On beating the hybrid argument. Theory Comput. **9**, 809–843 (2013). https://doi.org/10.4086/toc.2013.v009a026
8. Furst, M., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Math. Syst. Theory **17**(1), 13–27 (1984). https://doi.org/10.1007/BF01744431
9. Goldwasser, S., Gutfreund, D., Healy, A., Kaufman, T., Rothblum, G.N.: Verifying and decoding in constant depth. In: Symposium on Theory of Computing (STOC), pp. 440–449 (2007). https://doi.org/10.1145/1250790.1250855
10. Gutfreund, D., Rothblum, G.N.: The complexity of local list decoding. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX/RANDOM -2008. LNCS, vol. 5171, pp. 455–468. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85363-3_36
11. Håstad, J.: Almost optimal lower bounds for small depth circuits. In: Symposium on Theory of Computing (STOC), pp. 6–20 (1986). https://doi.org/10.1145/12130.12132
12. Impagliazzo, R., Kabanets, V., Volkovich, I.: The power of natural properties as oracles. In: Electronic Colloquium on Computational Complexity (ECCC) TR17-023 (2017). https://eccc.weizmann.ac.il/report/2017/023/
13. Nisan, N., Wigderson, A.: Hardness vs randomness. J. Comput. Syst. Sci. **49**(2), 149–167 (1994). https://doi.org/10.1016/S0022-0000(05)80043-1

14. Oliveira, I.C., Santhanam, R.: Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In: Computational Complexity Conference (CCC), pp. 18:1–18:49 (2017). https://doi.org/10.4230/LIPIcs.CCC.2017.18
15. Oliveira, I.C., Santhanam, R.: Pseudodeterministic constructions in subexponential time. In: Symposium on Theory of Computing (STOC), pp. 665–677 (2017). https://doi.org/10.1145/3055399.3055500
16. Razborov, A.A.: Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. Math. Notes Acad. Sci. USSR **41**(4), 333–338 (1987)
17. Servedio, R., Tan, L.Y.: What circuit classes can be learned with non-trivial savings? In: Innovations in Theoretical Computer Science Conference (ITCS), pp. 1–23 (2017)
18. Shaltiel, R., Viola, E.: Hardness amplification proofs require majority. SIAM J. Comput. **39**(7), 3122–3154 (2010). https://doi.org/10.1137/080735096
19. Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: Symposium on Theory of Computing (STOC), pp. 77–82 (1987). https://doi.org/10.1145/28395.28404
20. Srinivasan, S.: On improved degree lower bounds for polynomial approximation. In: Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), pp. 201–212 (2013). https://doi.org/10.4230/LIPIcs.FSTTCS.2013.201
21. Sudan, M., Trevisan, L., Vadhan, S.P.: Pseudorandom generators without the XOR lemma. J. Comput. Syst. Sci. **62**(2), 236–266 (2001). https://doi.org/10.1006/jcss.2000.1730
22. Williams, R.: Nonuniform ACC circuit lower bounds. J. ACM **61**(1), 2:1–2:32 (2014). https://doi.org/10.1145/2559903
23. Williams, R.: Natural proofs versus derandomization. SIAM J. Comput. **45**(2), 497–529 (2016). https://doi.org/10.1137/130938219
24. Yao, A.C.: Separating the polynomial-time hierarchy by oracles (preliminary version). In: Symposium on Foundations of Computer Science (FOCS), pp. 1–10 (1985). https://doi.org/10.1109/SFCS.1985.49