SIAM J. COMPUT.
Vol. 45, No. 2, pp. 497–529

NATURAL PROOFS VERSUS DERANDOMIZATION*

R. RYAN WILLIAMS†

Abstract. We study connections between the Natural Proofs of Razborov and Rudich, derandomization, and the problem of proving "weak" circuit lower bounds such as $\mathsf{NEXP} \not\subset \mathsf{TC}^0$, which are still wide open. Natural Proofs have three properties: they are *constructive* (an efficient algorithm A is embedded in them), have *largeness* (A accepts a large fraction of strings), and are *useful* (A rejects all strings which are truth tables of small circuits). Strong circuit lower bounds that are "naturalizing" would contradict present cryptographic understanding, yet the vast majority of known circuit lower bound proofs are naturalizing. So it is imperative to understand how to pursue un-Natural Proofs. Some heuristic arguments say constructivity should be circumventable: largeness is inherent in many proof techniques, and it is probably our presently weak techniques that yield constructivity. We prove the following: (i) *Constructivity is unavoidable*, even for NEXP lower bounds. Informally, we prove for all "typical" nonuniform circuit classes $\mathcal{C}$, $\mathsf{NEXP} \not\subset \mathcal{C}$ if and only if there is a polynomial-time algorithm distinguishing *some* function from all functions computable by $\mathcal{C}$-circuits. Hence $\mathsf{NEXP} \not\subset \mathcal{C}$ is equivalent to exhibiting a constructive property useful against $\mathcal{C}$. (ii) There are no P-natural properties useful against $\mathcal{C}$ if and only if randomized exponential time can be "derandomized" using truth tables of circuits from $\mathcal{C}$ as random seeds. Therefore the task of proving there are no P-natural properties is inherently a *derandomization* problem, weaker than but implied by the existence of strong pseudorandom functions. These characterizations are applied to yield several new results, including improved ACC^0 lower bounds and new unconditional derandomizations. In general, we develop and apply several new connections between the existence of certain algorithms for analyzing truth tables, and the *nonexistence* of small circuits for problems in large classes such as NEXP.

Key words. circuit complexity, natural proofs, derandomization, randomized computation, nondeterminism, ACC

AMS subject classifications. 68Q15, 68Q17, 68Q87

DOI. 10.1137/130938219

1. Introduction.

The Natural Proofs barrier of Razborov and Rudich [31] argues that

(a) almost all known proofs of nonuniform circuit lower bounds entail efficient algorithms that can distinguish many "hard" functions from all "easy" functions (those computable with small circuits), and
(b) any efficient algorithm of this kind would break cryptographic primitives implemented with small circuits (which are believed to exist).

(A formal definition is in section 2.) Natural Proofs are self-defeating: in the course of proving a weak lower bound, they provide efficient algorithms that refute stronger lower bounds that we believe to also hold. The moral is that, in order to prove stronger circuit lower bounds, one must avoid the techniques used in proofs that entail such efficient algorithms. The argument applies even to low-level complexity classes such as TC^0 [28, 24, 25], so any major progress in the future depends on proving un-Natural

*Received by the editors September 24, 2013; accepted for publication (in revised form) March 9, 2016; published electronically April 27, 2016. A preliminary version of this paper appeared in the ACM Symposium on Theory of Computing in 2013. This work was supported in part by a David Morgenthaler II Faculty Fellowship, a Sloan Fellowship, NSF DMS-1049268 (US Junior Oberwolfach Fellow), and NSF CCF-1212372. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
http://www.siam.org/journals/sicomp/45-2/93821.html
†Computer Science Department, Stanford University, Stanford, CA 94305 (rrw@cs.stanford.edu).

lower bounds.

How should we proceed? Should we look for proofs yielding only *inefficient* algorithms, avoiding "constructivity"? Or should we look for algorithms which cannot distinguish *many* hard functions from all easy ones, avoiding "largeness"?[1] (Note there is a third criterion, "usefulness," requiring that the proof distinguishes a target function f from the circuit class $\mathcal{C}$ we are proving lower bounds against. This criterion is necessary: $f \notin \mathcal{C}$ if and only if there is a trivial property, true of only f, distinguishing f from all functions computable in $\mathcal{C}$.) In this paper, we study alternative ways to characterize Natural Proofs and their relatives as particular circuit lower bound problems and give several applications. There are multiple competing intuitions about the meaning of Natural Proofs. We wish to rigorously understand the extent to which the Razborov–Rudich framework relates to our ability to prove lower bounds in general.

NEXP lower bounds are constructive and useful. Some relationships can be easily seen. Recall EXP and NEXP are the exponential-time versions of P and NP. If $\mathsf{EXP} \not\subset \mathcal{C}$, one can obtain a polynomial-time (nonlarge) property useful against $\mathcal{C}$.[2] So, strong enough lower bounds entail constructive useful properties. However, a separation like $\mathsf{EXP} \not\subset \mathcal{C}$ is stronger than currently known, for all classes $\mathcal{C}$ containing ACC^0. Could lower bounds be proved for larger classes like NEXP, without entering constructive/useful territory? In the other direction, could one exhibit a constructive (nonlarge) property against a small circuit class like TC^0, without proving a new lower bound against that class?

The answer to both questions is *no.* Call a (nonuniform) circuit class $\mathcal{C}$ *typical* if $\mathcal{C} \in \{\mathsf{AC}^0, \mathsf{ACC}^0, \mathsf{TC}^0, \mathsf{NC}^1, \mathsf{NC}, \mathsf{P}/\text{poly}\}$.[3] For any typical $\mathcal{C}$, a property of Boolean functions $\mathcal{P}$ is said to be *useful* against $\mathcal{C}$ if, for all k, there are infinitely many n such that

- $\mathcal{P}(f)$ is true of at least one $f : \{0,1\}^n \to \{0,1\}$, and
- $\mathcal{P}(g)$ is false for all $g : \{0,1\}^n \to \{0,1\}$ having n^k-size $\mathcal{C}$-circuits.

In other words, on infinitely many input lengths n, $\mathcal{P}$ distinguishes some function from all easy functions. We prove the following.

Theorem 1.1. *For all typical $\mathcal{C}$, $\mathsf{NEXP} \not\subset \mathcal{C}$ if and only if there is a polynomial-time computable property of Boolean functions that is useful against $\mathcal{C}$ with $O(\log n)$ bits of advice.*

That is, $\mathsf{NEXP} \not\subset \mathcal{C}$ if and only if there is a language in $\mathsf{P}/O(\log n)$ defining a property of Boolean functions useful against $\mathcal{C}$.

We can remove the $O(\log n)$ bits of advice of Theorem 1.1 by relaxing the notion of a "property" of Boolean functions to hold over all strings. Boolean function properties are only defined on 2^n-length binary strings; however, *every* binary string x can be viewed as the truth table of a unique Boolean function, by simply appending zeroes to the end of x until its length is a power of 2. For brevity we shall call this longer string f_x, which is a function from $\{0,1\}^\ell$ to $\{0,1\}$ where ℓ is the smallest integer satisfying $2^\ell \geq |x|$. Informally, we define an *algorithm A* to be *useful* against $\mathcal{C}$ if, for

[1] See the webpage [1] for a discussion with many views on these questions.

[2] Define $A(T)$ to accept its 2^n-bit input T if and only if T is the truth table of a function that is complete for $\mathsf{E} = \mathsf{TIME}[2^{O(n)}]$. A can be implemented to run in poly(2^n) time and rejects all T with $\mathcal{C}$-circuits, assuming $\mathsf{EXP} \not\subset \mathcal{C}$.

[3] For simplicity, in this paper we mostly restrict ourselves to typical classes; however, it will be clear from the proofs that we only rely on a few properties of these classes, and more general statements can be made.

all k, there are infinitely many input lengths N such that

- for at least one $x \in \{0,1\}^N$, $A(x) = 1$, and
- for all $x' \in \{0,1\}^N$ such that $f_{x'} : \{0,1\}^\ell \to \{0,1\}$ has n^k-size $\mathcal{C}$-circuits, $A(x') = 0$.

THEOREM 1.2. *For all typical $\mathcal{C}$, $\mathsf{NEXP} \not\subset \mathcal{C}$ if and only if there is a polynomial-time* algorithm *that is useful against $\mathcal{C}$.*

Theorems 1.1 and 1.2 help explain *why* it is difficult to prove even NEXP circuit lower bounds: *any* NEXP lower bound must meet precisely two of the three conditions of Natural Proofs (constructivity and usefulness).[4] The above two theorems say that *every* NEXP circuit lower bound must exhibit some constructive property useful against those circuits. Polynomial-time algorithms distinguishing "some" functions from "all" easy functions look difficult to construct, even infinitely often; if one adds in largeness too, these algorithms are likely *impossible* to construct.

One can make a heuristic argument that the recent proof of $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ [36] evades Natural Proofs by being nonconstructive. Intuitively, the proof uses an ACC^0 Circuit SAT algorithm that only mildly improves over brute force, so it runs too slowly to obtain a polytime property useful against ACC^0. Theorem 1.1 shows that, in fact, constructivity is necessary. Moreover, the proofs of Theorems 1.1 and 1.2 yield an explicit property useful against ACC^0.

The techniques used in these theorems can be applied, along with several other ideas, to prove new superpolynomial lower bounds against ACC^0. First, we prove exponential-size lower bounds on the ACC^0 circuit complexity of encoding *witnesses* for NEXP languages.

THEOREM 1.3. *For all d, m there is an $\varepsilon > 0$ such that $\mathsf{NTIME}[2^{O(n)}]$ does not have 2^{n^ε}-size d-depth $\mathsf{AC}^0[m]$ witnesses.*

Formal definitions can be found in section 3; informally, Theorem 1.3 says that there are NEXP languages with verifiers that only accept witness strings of exponentially high ACC^0 circuit complexity. It is interesting that while we can prove such lower bounds for encoding NEXP witnesses, we do not yet know how to prove them for NEXP languages themselves (the best known size lower bound for NEXP is "third-exponential").

These circuit lower bounds for witnesses can also be translated into new ACC^0 lower bounds for some complexity classes. Recall that $\mathsf{NE} = \mathsf{NTIME}[2^{O(n)}]$ and io-$\mathsf{P} = io$-$\mathsf{TIME}[n^{O(1)}]$, the latter being the class of languages L such that there is an $L' \in \mathsf{P}$ where, for infinitely many n, $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$. That is, L agrees with a language in P on infinitely many input lengths. The class $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ consists of languages $L \in \mathsf{NE} \cap \mathsf{coNE}$ recognizable with "one bit of advice." That is, there are nondeterministic machines M and M' running in $2^{O(n)}$ time with the property that for all n, there are bits $y_n, z_n \in \{0,1\}$ such that for all strings x, $x \in L$ if and only if $M(x, y_n)$ accepts on all paths if and only if $M'(x, z_n)$ rejects on all paths. (In fact, in our case we may assume $y_n = z_n$ for all n.)

[4] One may also wonder whether *nonconstructive large properties* imply any new circuit lower bounds. This question does not seem to be as interesting. For one, there are already coNP-natural properties useful against P/poly (simply try all possible small circuits in parallel), and the consequences of such properties are well known. So anything coNP-constructive or worse is basically uninformative (without further information on the property). Furthermore, slightly more constructive properties, such as NP-natural ones, seem unlikely [32].

THEOREM 1.4. $\mathsf{NE} \cap io\text{-}\mathsf{P}$ *and* $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ *do not have* ACC^0 *circuits of* $n^{\log n}$ *size.*[5]

As observed by Russell Impagliazzo and an anonymous referee, it is easy to derive a circuit lower bound for the class $\mathsf{NE} \cap io\text{-}\mathsf{P}$, directly from a lower bound for NE. Let $L \in \mathsf{NE}$ be a language that does not have $\mathcal{C}$-circuits of size $s(n)$, for some circuit class $\mathcal{C}$. Define the language $L' = \{xx \mid x \in L\}$, which is in $\mathsf{NE} \cap io\text{-}\mathsf{P}$. It is clear that if L' had $\mathcal{C}$-circuits of size $s(n/2)$, then L would also have $\mathcal{C}$-circuits of size $s(n)$, a contradiction.

The lower bound for $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ is intriguing because (as far as we can tell) it must necessarily be proved differently. The known proof of $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ works for the class NEXP because there is a tight time hierarchy for nondeterminism [38]. However, the $\mathsf{NTIME} \cap \mathsf{coNTIME}$ classes are not known to have such a hierarchy. (They are among the "semantic" classes, which are generally not known to have complete languages or nice time hierarchies.) Interestingly, the proof of Theorem 1.4 crucially uses the previous lower bound framework against NEXP, and builds on it, via Theorem 1.1 and a modification of the $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ lower bound. Indeed, it follows from the arguments here (building on [37, 36]) that the lower bound consequences of nontrivial circuit SAT algorithms can be strengthened in the following sense.

THEOREM 1.5. *Let* $\mathcal{C}$ *be typical. Suppose the satisfiability problem for* $n^{O(\log^c n)}$-*size* $\mathcal{C}$-*circuits can be solved in* $O(2^n/n^{10})$ *time, for all constants* c. *Then* $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ *do not have* $n^{\log n}$-*size* $\mathcal{C}$-*circuits.*

THEOREM 1.6. *Suppose we can approximate the acceptance probability of any given* $n^{O(\log^c n)}$-*size circuit (with fan-in two and arbitrary depth) on* n *inputs to within* $1/6$, *for all* c, *in* $O(2^n/n^{10})$ *time (even nondeterministically). Then* $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ *do not have* $n^{\log n}$-*size circuits.*

Natural Proofs vs. derandomization. Given Theorem 1.1, it is natural to wonder whether full-strength natural properties are equivalent to some circuit lower bound problems. If so, such lower bounds should be considered *unlikely*. To set up the discussion, let $\mathsf{RE} = \mathsf{RTIME}[2^{O(n)}]$ and $\mathsf{ZPE} = \mathsf{ZPTIME}[2^{O(n)}]$; that is, RE is the class of languages solvable in $2^{O(n)}$ randomized time with one-sided error, and ZPE is the corresponding class with zero-error (i.e., *expected* $2^{O(n)}$ running time).

For a typical circuit class $\mathcal{C}$, we informally say that RE (resp., ZPE) *has* $\mathcal{C}$ *seeds* if, for every predicate defining a language in the respective complexity class, there are $\mathcal{C}$-circuit families succinctly encoding exponential-length "seeds" that correctly decide the predicate. (Formal definitions are given in section 5.) Having $\mathcal{C}$ seeds means that the randomized class can be derandomized very strongly: by trying all poly-size $\mathcal{C}$-circuits as random seeds, one can decide any predicate from the class in EXP.

We prove a strong correspondence between the existence of such seeds and the *nonexistence* of natural properties.

THEOREM 1.7. *Let* $\mathcal{C}$ *be typical. The following are equivalent:*

1. *There are no* P-*natural properties useful (resp., ae-useful*[6]*) against* $\mathcal{C}$.

[5] These are not the strongest size lower bounds that can be proved, but they are among the cleanest. Please note that the conference version of this paper claimed a lower bound for the class $\mathsf{NE} \cap \mathsf{coNE}$. We are grateful to Russell Impagliazzo and Igor Carboni Oliveira [30] for observing that our arguments presently only prove lower bounds for the above two classes.

[6] Here, *ae-useful* is just the "almost-everywhere useful" version, where the property is required to distinguish random functions from easy ones on almost every input length.

2. ZPE *has* $\mathcal{C}$ *seeds for almost all (resp., infinitely many) input lengths.*

Informally, Theorem 1.7 says that ruling out P-natural properties is equivalent to a strong derandomization of randomized exponential time, using small circuits to encode exponentially long random seeds. Similarly, we prove that a variant of natural properties is related to succinct "hitting sets" for RE (Theorem 5.1).

It is worth discussing the meaning of these results in a little more detail. Let $\mathcal{C}, \mathcal{D}$ be appropriate circuit classes. Roughly speaking, the key lesson of Natural Proofs [31, 28, 24] is that if there are $\mathcal{D}$-natural properties useful against $\mathcal{C}$, then there are no pseudorandom functions (PRFs) computable in $\mathcal{C}$ that fool $\mathcal{D}$-circuits; namely, there is a statistical test T computable in $\mathcal{D}$ such that, for every function $f(\cdot,\cdot) \in \mathcal{C}$, the test T with query access to $f(x,\cdot)$ (where x is a uniform random n-bit seed) can distinguish $f(x,\cdot)$ from a uniform random function (generated using 2^n uniform random bits). Now, if we have a PRF computable in $\mathcal{C}$ that can fool $\mathcal{D}$-circuits, this PRF can be used to obtain $\mathcal{C}$ seeds for randomized $\mathcal{D}$-circuits with one-sided error.[7] That is, the existence of PRFs implies the existence of $\mathcal{C}$ seeds, so our consequence in Theorem 1.7 (of the existence of natural properties) that "no ZPE predicate has $\mathcal{C}$ seeds" appears stronger than "there are no PRFs" (as in [31]). Moreover, this stronger consequence in Theorem 1.7 (and Theorem 5.1, proved later) yields an implication in the reverse direction: the *lack* of $\mathcal{D}$-natural properties implies strong derandomizations of randomized exponential-size $\mathcal{D}$.

Theorem 1.7 also shows that it is plausible that some derandomization problems are as hard as resolving $\mathsf{P} \neq \mathsf{NP}$. Since we believe that there are no P-natural properties useful against P/poly, then by Theorem 1.7, we must also believe that there are "canonical" derandomizations of ZPE in EXP, along the lines of item 2 in Theorem 1.7. However, *proving* that such a canonical derandomization exists would in turn imply that there are no P-natural properties useful against P/poly (again by Theorem 1.7), and hence $\mathsf{P} \neq \mathsf{NP}$.

Unconditional mild derandomizations. Understanding the relationships between the randomized complexity classes ZPP, RP, and BPP is a central problem in modern complexity theory. It is well known that

$$\mathsf{P} \subseteq \mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP} \subseteq \mathsf{RP} \subseteq \mathsf{BPP},$$

but it is not known whether any inclusion is an equality. The ideas behind Theorem 1.7 can also be applied to prove new relations between these classes. We define $\mathsf{ZPTIME}[t(n)]/d(n)$ to be the class of languages solvable in zero-error time $t(n)$ by machines of description length at most $d(n)$ (under some standard encoding of machines).[8] The "infinitely often" version io-$\mathsf{ZPTIME}[t(n)]/d(n)$ is the class of languages L solvable with machines of description length $d(n)$ running in time $t(n)$ that are zero-error for infinitely many input lengths: for infinitely many n, the machine has the zero-error property on all inputs of length n.

THEOREM 1.8. *Either*
(a) $\mathsf{RTIME}[2^{O(n)}] \subseteq \mathsf{SIZE}[n^c]$ *for some c, or*

[7] Consider any $\mathcal{D}$-circuit D that tries to use f as a source of randomness. A $\mathcal{C}$-circuit seed for D can be obtained from a circuit computing f: since f fools D, at least one n-bit seed to f will make D^f print 1.

[8] **N.B.** Although our definition is standard (see, for example, [8, 14]), it is important to note that there are other possible interpretations of the same notation. Here, we only require that the algorithm is required to be zero-error for the "correct" advice or description, but one could also require the algorithm to be zero-error *no matter what* advice is given.

(b) $\mathsf{BPP} \subseteq io\text{-}\mathsf{ZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$ *for all* $\varepsilon > 0$.

We have a win-win: either randomized exptime is very easy with nonuniform circuits, or randomized computation with two-sided error has a *zero-error* simulation (with description size n^{ε}) that dramatically avoids brute force. To appreciate the theorem statement, suppose the first case could be modified to conclude that $\mathsf{RP} \subseteq io\text{-}\mathsf{ZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$ for all $\varepsilon > 0$. Then the famous (coRP) problem of polynomial identity testing (PIT) would have a new subexponential-time algorithm, good enough to prove strong NEXP circuit lower bounds.[9] A quick corollary of Theorem 1.8 comes close to achieving this. To simplify notation, we use the SUBEXP modifier in a complexity class to abbreviate "$2^{n^{\varepsilon}}$ time for every $\varepsilon > 0$."

Corollary 1.1. *For some* $c > 0$, $\mathsf{RP} \subset io\text{-}\mathsf{ZPSUBEXP}/n^c$.

That is, the error in an RP computation can be removed in subexponential time with fixed-polynomial advice, infinitely often. We emphasize that the advice needed is *independent* of the running times of the RP and ZPSUBEXP computations: the RP computation could run in $n^{c^{c^{c^{c}}}}$ time and still need only n^c advice to be simulated in $2^{n^{1/c^{c^{c^{c}}}}}$ time. Corollary 1.1 should be compared with a theorem of Kabanets [19], who gave a simulation of RP in *pseudo*-subexponential time with zero-error. That is, his simulation is only guaranteed to succeed against efficient adversaries which try to generate bad inputs (but his simulation also does not require advice).

An analogous argument can be used to give a new simulation of Arthur–Merlin games. Informally (and following the notation outlined above), $io\text{-}\Sigma_2\mathsf{SUBEXP}/n^c$ is the class of languages which agree infinitely often with Σ_2 machines running in $2^{n^{\varepsilon}}$ time, for all $\varepsilon > 0$, with $O(n^c)$ bits of advice.

Corollary 1.2. *For some* $c > 0$, $\mathsf{AM} \subseteq io\text{-}\Sigma_2\mathsf{SUBEXP}/n^c$.

The ideas used here can also be applied to prove a new equivalence between $\mathsf{NEXP} = \mathsf{BPP}$ and nontrivial simulations of BPP. Informally, the complexity class $io\text{-}\mathsf{HeuristicZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$ is the class of languages which, for infinitely many n, agree on a $1 - 1/n$ fraction of the n-bit inputs with zero-error randomized subexponential time machines using $O(n^{\varepsilon})$ advice.

Theorem 1.9. $\mathsf{NEXP} \neq \mathsf{BPP}$ *if and only if for all* $\varepsilon > 0$, BPP *is contained in* $io\text{-}\mathsf{HeuristicZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$.

Finally, these ideas can be extended to show an *equivalence* between the existence of RP-natural properties and P-natural properties against a circuit class.

Theorem 1.10. *If there exists an* RP*-natural property* P *useful against a typical class* $\mathcal{C}$*, then there exists a* P*-natural property* P' *against* $\mathcal{C}$.

That is, given any property P with one-sided error that is sufficient for distinguishing all easy functions from many hard functions, we can obtain a deterministic property P' with analogous behavior. (Note this is not *exactly* a derandomization of property P; the property P' will in general have different input-output behavior from P, but P' does use P as a subroutine.) The key idea of the proof is to *swap the input with the randomness* in the property P.

[9] More precisely, the main result of Kabanets and Impagliazzo [21] concerning the derandomization of PIT can be extended as follows: if PIT for arithmetic circuits can be solved for infinitely many circuit sizes in nondeterministic subexponential time, then either $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ or the permanent does not have polynomial-size arithmetic circuits.

2. Preliminaries. For simplicity, all languages are over $\{0,1\}$. We fix some standard encoding of Turing machines and define the *description length* of a machine M to be the length of M under the encoding. We assume knowledge of the basics of complexity theory [4], such as advice-taking machines, and complexity classes like $\mathsf{EXP} = \mathsf{TIME}[2^{n^{O(1)}}]$, $\mathsf{NEXP} = \mathsf{NTIME}[2^{n^{O(1)}}]$, $\mathsf{AC}^0[m]$, ACC^0, and so on. We use $\mathsf{SIZE}[s(n)]$ to denote the class of languages recognized by a (nonuniform) $s(n)$-size circuit family. We also use the (standard) "subexponential-time" notation $\mathsf{SUBEXP} = \bigcap_{\varepsilon>0} \mathsf{TIME}[2^{O(n^{\varepsilon})}]$. (So, for example, $\mathsf{NSUBEXP}$ refers to the class of languages accepted in nondeterministic $2^{n^{\varepsilon}}$ time, for all $\varepsilon > 0$.) When we refer to a "typical" circuit class (AC^0, ACC^0, TC^0, NC^1, NC, or $\mathsf{P}/\text{poly}\}$), we will always assume the class is *nonuniform*, unless otherwise specified. Some familiarity with prior work connecting SAT algorithms and circuit lower bounds [37, 36] would be helpful, but this paper is mostly self-contained.

We will use *advice classes*: for a deterministic or nondeterministic class $\mathcal{C}$ and a function $a(n)$, $\mathcal{C}/a(n)$ is the class of languages L such that there is an $L' \in \mathcal{C}$ and an arbitrary function $f : \mathbb{N} \to \{0,1\}^{\star}$ with $|f(n)| \le a(n)$ for all x such that $L = \{x \mid (x, f(|x|)) \in L'\}$. That is, the arbitrary advice string $f(n)$ can be used to solve all n-bit instances within class $\mathcal{C}$.

For semantic (e.g., randomized, $\mathsf{NTIME} \cap \mathsf{coNTIME}$) classes $\mathcal{C}$, the definition of advice is technically subtle. We shall only require that the class $\mathcal{C}$ algorithm exhibits the relevant promise condition (zero-error, one-sided error, or otherwise) for the "correct" advice or description; one could also require that the algorithm satisfies the promise condition *no matter what* advice is given.

More precisely, for a randomized machine M and complexity class $\mathcal{C}$ in the set $\{\mathsf{RTIME}[t(n)], \mathsf{ZPTIME}[t(n)], \mathsf{BPTIME}[t(n)]\}$, we say that M is of type $\mathcal{C}$ on a given input x if M on x runs in time $t(|x|)$ and M satisfies the promise of one-sided/zero/two-sided error on input x. (For example, in the case of one-sided error, if $x \in L$, then M on x should accept at least $2/3$ of the computation paths; if $x \notin L$, then M on x should reject all of the computation paths. In the case of zero-error, if $x \in L$, then M on x should accept at least $2/3$ of the paths and output **?** (i.e., *don't know*) on the others; if $x \notin L$, then M on x should reject at least $2/3$ of the paths and output **?** on the others.) Then for $\mathcal{C} \in \{\mathsf{RTIME}[t(n)], \mathsf{ZPTIME}[t(n)], \mathsf{BPTIME}[t(n)]\}$, $\mathcal{C}/a(n)$ is the class of languages L recognized by a randomized machine of *description length* $a(n)$ (under some standard encoding of machines) that is of type $\mathcal{C}$ on all inputs [8]. Equivalently, $L \in \mathcal{C}/a(n)$ is in the class if there is a machine M and advice function $s : \mathbb{N} \to \{0,1\}^{a(n)}$ such that for all $x \in \{0,1\}^{\star}$, M is a machine of type $\mathcal{C}$ when executed on input $(x, a(|x|))$ (M satisfies the promise of one-sided/zero/two-sided error on that input) and $x \in L$ if and only if $M(x, a(|x|))$ accepts [14].

We also use *infinitely often classes*: for a deterministic or nondeterministic complexity class $\mathcal{C}$, *io*-$\mathcal{C}$ is the class of languages L such that there is an $L' \in \mathcal{C}$ where, for infinitely many n, $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$. For randomized classes $\mathcal{C} \in \{\mathsf{RTIME}[t(n)], \mathsf{ZPTIME}[t(n)], \mathsf{BPTIME}[t(n)]\}$, as well as "semantic" complexity classes such as $(\mathsf{NTIME} \cap \mathsf{coNTIME})[t(n)]$, *io*-$\mathcal{C}$ is the class of languages L recognized by a machine M such that, for infinitely many input lengths n, M is of type $\mathcal{C}$ on all inputs of length n (and need not be of type $\mathcal{C}$ on other input lengths).

Some particular notation and conventions will be useful. For any circuit C viewed as a function $C(x_1, x_2, \ldots, x_n)$, $i < j$, and $a_1, \ldots, a_n \in \{0,1\}$, the notation $C(a_1, \ldots, a_i, \cdot, a_j, \ldots, a_n)$ represents the circuit with $j - i - 1$ inputs obtained by assigning the input x_q to a_q, for all $q \in [1, i] \cup [j, n]$. In general, the symbol $\cdot$ is used to denote free unassigned inputs to the circuit.

2.1. Truth tables and their circuit complexity. In this paper, we study the circuit complexities of all strings, even those which are not of length equal to a power of two. To make the discussion precise, we carefully develop the concepts in this section.

Let $y_1, \ldots, y_{2^k} \in \{0,1\}^k$ be the list of k-bit strings in lex order. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the truth table of f is defined to be

$$tt(f) := f(y_1)f(y_2)\cdots f(y_{2^n}),$$

and the truth table of a circuit is simply the truth table of the function it defines. For binary strings with lengths that are not powers of two, we use the following encoding convention. Let T be a binary string, and let $k = \lceil \log_2 |T| \rceil$. The *Boolean function encoded by* T or the *function corresponding to* T, denoted by f_T, is the function satisfying $tt(f_T) = T0^{2^k - |T|}$.

The *size* of a circuit is its number of gates. The circuit complexity of an arbitrary string (and, hence, a function) takes some care to properly define, based on the circuit model. For the unrestricted model, the *circuit complexity of* T, denoted as $CC(T)$, is simply the minimum size of any circuit computing f_T. For a depth-bounded circuit model, where a depth function must be specified prior to giving the circuit family, the appropriate measure is the *depth-d circuit complexity of* T, denoted as $CC_d(T)$, which is the minimum size of any depth-d circuit computing f_T. (Note that, even for circuit classes like NC^1, we have to specify a depth upper bound $c \log n$ for some constant c.) For the class ACC^0, we must specify a modulus m for the MOD gates, as well as a depth bound, so when considering ACC^0 circuit complexity, we look at the *depth-d mod-m circuit complexity of* T, $CC_{d,m}(T)$, for fixed d and m.

A simple fact about the circuit complexities of truth tables and their substrings will be very useful.

PROPOSITION 1. *Suppose $T = T_1 \cdots T_{2^k}$ is a string of length $2^{k+\ell}$ such that $T_1, \ldots, T_{2^k}$ each have length 2^ℓ. Then $CC(T_i) \le CC(T)$, $CC_d(T_i) \le CC_d(T)$, and $CC_{d,m}(T_i) \le CC_{d,m}(T)$.*

Proof. Given a circuit C of size s for f_T, a circuit for f_{T_i} is obtained by substituting values for the first k inputs of C. This yields a circuit of size at most s. ∎

Sometimes we will require a more general claim: for any string T, the circuit complexity of an arbitrary substring of T can be bounded via the circuit complexity of T.

LEMMA 2.1. *There is a universal $c \ge 1$ such that the following holds. Let T be a binary string, and let S be any substring of T. Then for all d and m, $CC(f_S) \le CC(f_T) + (c \log |T|)$, $CC_d(f_S) \le CC_{d+c}(f_T) + (c \log |T|)^{1+o(1)}$, and $CC_{d,m}(f_S) \le CC_{d+c,m}(f_T) + (c \log |T|)^{1+o(1)}$.*

Proof. Let c' be sufficiently large in the following. Let k be the minimum integer satisfying $2^k \ge |T|$ so that the Boolean function f_T representing T has truth table $T0^{2^k - |T|}$. Suppose C is a size-s depth-d circuit for f_T. Let S be a substring of $T = t_1 \cdots t_{2^k} \in \{0,1\}^{2^k}$, and let $A, B \in \{1, \ldots, 2^k\}$ be such that $S = t_A \cdots t_B$. Let $\ell \le k$ be a minimum integer which satisfies $2^\ell \ge B - A$. Our goal is to construct a small circuit D with ℓ inputs and truth table $S0^{2^\ell - (B-A)}$.

Let $x_1, \ldots, x_{2^\ell}$ be the ℓ-bit strings in lex order. The desired circuit D on input x_i can be implemented as follows: Compute $i + A$. If $(i + A) \le B$, then output $C(x_{i+A})$;

otherwise output 0. To bound the size of D, first note there are depth-c' circuits of at most $c' \cdot n \log^\star n$ size for addition of two n-bit numbers [11], and there are also well-known $O(n)$-size (unrestricted depth) circuits for addition.

Therefore in depth-c' and size at most $c' \cdot k \log^\star k$ we can, given input x_i of length ℓ, output $i + A$. Determining whether $i \leq B - A$ can be done with $(c' \cdot \ell)$-size depth-c' circuits. Therefore D can either be implemented as a circuit of size at most $s + c'((k \log^\star k) + \ell + 1)$ and depth $2c' + d$, or as an (unrestricted depth) circuit of size at most $s + c'(k + \ell + 1)$. To complete the proof, let $c \geq 3c'$. ∎

We will use the following strong construction of pseudorandom generators from hard functions.

THEOREM 2.1 (Umans [35]). *There is a universal constant g and a function $G : \{0,1\}^\star \times \{0,1\}^\star \to \{0,1\}^\star$ such that, for all s and Y satisfying $CC(Y) \geq s^g$, and for all circuits C of size s,*

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

Furthermore, G is computable in poly$(|Y|)$ *time.*

Natural Proofs. A *property of Boolean functions* $\mathcal{P}$ is a subset of the set of all Boolean functions. Let Γ be a complexity class, and let $\mathcal{C}$ be a circuit class (typically, $\Gamma = \mathsf{P}$ and $\mathcal{C} = \mathsf{P}/\text{poly}$). A Γ-*natural property useful against* $\mathcal{C}$ is a property of Boolean functions $\mathcal{P}$ that satisfies the following axioms:

- (Constructivity) $\mathcal{P}$ is decidable in Γ.
- (Largeness) For all n, $\mathcal{P}$ contains a $1/2^{O(n)}$ fraction of all 2^n-bit strings.
- (Usefulness) Let $f = \{f_n\}$ be a sequence of functions $\{f_n\}$ such that $f_n \in \mathcal{P}$ for all n. Then for all k and infinitely many n, f_n does not have n^k-size $\mathcal{C}$-circuits.[10]

Let $f = \{f_n : \{0,1\}^n \to \{0,1\}\}$ be a sequence of Boolean functions. A Γ-*natural proof* that $f \notin \mathcal{C}$ establishes the existence of a Γ-natural property $\mathcal{P}$ useful against $\mathcal{C}$ such that $\mathcal{P}(f_n) = 1$ for all n. Razborov and Rudich proved that any P/poly-natural property useful against P/poly could break all strong pseudorandom generator candidates in P/poly. More generally, P/poly-natural properties useful against typical $\mathcal{C} \subset \mathsf{P}/\text{poly}$ imply there are no strong PRFs in $\mathcal{C}$ (but such functions are believed to exist, even when $\mathcal{C} = \mathsf{TC}^0$ [28]).

The natural property framework (as originally defined) only applies to strings encoding Boolean functions, with lengths always equal to a power of two. In this paper, we also consider the obvious extension of the natural property concept to *arbitrary* length strings. We call such objects *natural algorithms*, to emphasize that they are best viewed as algorithms operating on inputs of arbitrary length. For a string x of length n, let ℓ be the smallest integer such that $2^\ell \geq n$. Recall we defined the *Boolean function corresponding to x* to be $f_x : \{0,1\}^\ell \to \{0,1\}$ with truth table $x0^{2^\ell - n}$.

DEFINITION 2.1. *A* Γ-natural algorithm A useful against $\mathcal{C}$ *satisfies the following axioms:*

- *(Constructivity) $L(A)$ is in Γ.*
- *(Largeness) For all n, A accepts at least a $1/n^{O(1)}$ fraction of all n-bit strings.*

[10] Note that some papers, including Razborov and Rudich [31], replace "infinitely many" with "almost every"; in this paper, we call that version *ae-usefulness*.

- *(Usefulness) There are infinitely many n such that*
  - (a) *A accepts at least one string x of length n, and*
  - (b) *for all y of length n accepted by A, the function f_y does not have n^k-size $\mathcal{C}$-circuits.*

The above definition of natural algorithm does not radically change the notion of usefulness (due to Lemma 2.1 in section 2.1); that is, padding a modest number of zeroes onto a string does not significantly alter the circuit complexity of the function represented by the string. However, the generalization to arbitrary input lengths is very useful for connecting the ideas of natural proofs to derandomization and circuit lower bounds.

2.2. Related work.

Equivalences between algorithms and lower bounds. Some of our results are equivalences between algorithm design problems and circuit lower bounds. Equivalences between derandomization hypotheses and circuit lower bounds have been known for some time, and recently there has been an increase in results of this form. Nisan and Wigderson [29] famously proved an equivalence between "approximate" circuit lower bounds and the existence of pseudorandom generators. Impagliazzo and Wigderson [17] prove that $\mathsf{BPP} \neq \mathsf{EXP}$ implies deterministic subexponential-time *heuristic* algorithms for BPP (the simulation succeeds on most inputs drawn from an efficiently samplable distribution, for infinitely many input lengths). As the opposite direction can be shown to hold, this is actually an equivalence. (Impagliazzo, Kabanets, and Wigderson [16] proved another such equivalence, which we discuss below.) Two more recent examples are Jansen and Santhanam [18], who give an equivalence between nontrivial algorithms for polynomial identity testing and lower bounds for the algebraic version of NEXP, and Aydinlioglu and van Melkebeek [5], who give an equivalence between Σ_2-simulations of Arthur–Merlin games and circuit lower bounds for $\Sigma_2\mathsf{EXP}$.

Almost-Natural Proofs. Strongly philosophically related to the present work, Chow [13] showed that if strong pseudorandom generators do exist, then there *is* a proof of $\mathsf{NP} \not\subset \mathsf{P}/\text{poly}$ that is *almost-natural*, where the fraction of inputs in the largeness condition is relaxed from $1/2^{O(n)}$ to $1/2^{n^{\text{poly}(\log n)}}$. Hence the Natural Proofs barrier was already known to be sensitive to relaxations of largeness. To compare, we show that removing the largeness condition entirely results in a direct *equivalence* between the existence of "almost-natural" properties and circuit lower bounds against NEXP. Chow also proved relevant unconditional results: for example, there exists a $\mathsf{SIZE}[O(n)]$-natural property that is $1/2^{n^{(\log n)^{\omega(1)}}}$-large and useful against P/poly. Theorem 1.1 shows that if $\mathsf{SIZE}[O(n)]$ could be replaced with P, then $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$ follows.

The work of Impagliazzo, Kabanets, and Wigderson. Impagliazzo, Kabanets, and Wigderson [16] proved a theorem similar to one direction of Theorem 1.1, showing that an NP-natural property (without largeness) useful against P/poly implies $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$. Allender [2] proved that there is a (nonlarge) property computable in NP useful against P/poly if and only if there is such a property in uniform AC^0. Hence his equivalence implies, at least for $\mathcal{C} = \mathsf{P}/\text{poly}$, that the "polynomial-time" guarantee of Theorem 1.1 can be relaxed to "AC^0."

Impagliazzo, Kabanets, and Wigderson [16] also give an equivalence between NEXP lower bounds and an algorithmic problem: $\mathsf{NEXP} \not\subset \mathsf{P}/\text{poly}$ if and only if the acceptance probability of any circuit can be approximated, for infinitely many circuit sizes, in nondeterministic subexponential time with subpolynomial advice. The major differences between their equivalence and Theorems 1.1 and 1.2 are in the underlying computational problems and the algorithmic guarantees: they study subexponential-time algorithms for approximating acceptance probabilities of circuits, while we study algorithms which estimate the circuit complexities of given functions. Moreover, their equivalence is less general with respect to circuit classes; for example, it is not known how to prove an analogue of their equivalence for ACC^0.

Since they proved that the existence of NP-natural properties useful against P/poly imply that $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$, Impagliazzo, Kabanets, and Wigderson posed the following interesting open problem:

Does the existence of a P*-natural property useful against* P/poly *imply* $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$*?*

Our work shows that the *absence* of a P-natural property useful against P/poly implies new lower bounds.

COROLLARY 2.1. *If there is no* P*-natural property useful against* P/poly*, then* $\mathsf{NP} \neq \mathsf{ZPP}$*.*

Proof. We prove the contrapositive. If $\mathsf{NP} = \mathsf{ZPP}$, then there is a ZPP-natural property useful against P/poly (since there are trivially coNP-natural properties). Theorem 1.10 implies there is also a P-natural property useful against P/poly. ∎

Therefore, an affirmative answer to the problem of Impagliazzo, Kabanets, and Wigderson would prove that $\mathsf{EXP} \neq \mathsf{ZPP}$.

THEOREM 2.2. *If (*P*-natural properties useful against* $\mathsf{P}/\mathrm{poly} \Rightarrow \mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$*) is true, then* $\mathsf{EXP} \neq \mathsf{ZPP}$ *unconditionally.*

Proof. We have

$$\begin{aligned}\mathsf{EXP} = \mathsf{ZPP} &\Rightarrow \mathsf{NP} = \mathsf{ZPP}\\ &\Rightarrow \text{there are } \mathsf{P}\text{-natural properties useful against } \mathsf{P}/\mathrm{poly}, \text{ by Corollary 2.1}\\ &\Rightarrow \mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}, \text{ by assumption}\\ &\Rightarrow \mathsf{EXP} \neq \mathsf{ZPP}.\end{aligned}$$

Thus $\mathsf{EXP} \neq \mathsf{ZPP}$. ∎

3. NEXP lower bounds and useful properties. In this section, we prove equivalences between NEXP circuit lower bounds and some relaxations of natural properties.

REMINDER OF THEOREM 1.1. *For all typical* $\mathcal{C}$*,* $\mathsf{NEXP} \not\subset \mathcal{C}$ *if and only if there is a polynomial-time computable property of Boolean functions that is useful against* $\mathcal{C}$ *with* $O(\log n)$ *bits of advice.*

REMINDER OF THEOREM 1.2. *For all typical* $\mathcal{C}$*,* $\mathsf{NEXP} \not\subset \mathcal{C}$ *if and only if there is a polynomial-time algorithm that is useful against* $\mathcal{C}$*.*

Our proofs of these theorems take several steps (they could be shortened, as in Oliveira's survey [30], but the overall proofs would be less informative). First, we give an equivalence between the existence of small circuits for NEXP and the existence of small circuits encoding *witnesses* to NEXP languages (Theorem 3.1), strengthening the results of Impagliazzo, Kabanets, and Wigderson [16] (who essentially proved one direction of the equivalence). Second, we prove an equivalence between the *nonexistence* of size-$s(O(n))$ witness circuits for NEXP and the existence of a P-constructive property P_s useful against size $s(O(n))$ circuits (Theorem 3.2), for all circuit sizes $s(n)$. For each polynomial $s(n) = n^k$, this yields a (potentially different) useful property P_s; to get a single property that works for all polynomial circuit sizes, we show that there exists a "universal" P-constructive property $P^\star$: if for every circuit size s there is *some* P-constructive useful property P_s, this particular property $P^\star$ is useful for all s (Theorem 3.3).

We first need a definition of what it means for a language (and a complexity class) to have small circuits encoding witnesses. We restrict ourselves to "good" verifiers which examine witnesses of length equal to a power of two so that witnesses can be viewed as truth tables of Boolean functions.

DEFINITION 3.1. *Let $L \in \mathsf{NTIME}[t(n)]$, where $t(n) \geq n$ is constructible, and let $\mathcal{C}$ be a circuit class. An algorithm $V(x, y)$ is a* good predicate *for L if*

- *V runs in time $O(\text{poly}(|y| + t(|x|)))$, and*
- *for all $x \in \{0,1\}^\star$, $x \in L$ if and only if there is a string y such that $|y| = 2^\ell \leq O(t(|x|))$ for some ℓ (a* witness *for x) such that $V(x, y)$ accepts.*

Let $L(V)$ denote the language accepted by V.

For every $L \in \mathsf{NTIME}[t(n)]$, basic complexity arguments show that there is at least one good predicate V such that $L = L(V)$. Furthermore, for every reasonable verifier V used to define an NEXP language L, there is an equivalent good predicate V' (with possibly slightly longer witness lengths). Now we define what it means for a verifier to have small-circuit witnesses.

DEFINITION 3.2. *Let V be a good predicate. V* has $\mathcal{C}$ witnesses of size $s(n)$ *if for all strings x, if $x \in L$, then there is a $\mathcal{C}$-circuit C_x of size at most $s(n)$ such that $V(x, tt(C_x(\cdot)))$ accepts.*

L has $\mathcal{C}$ witnesses of $s(n)$ size *if for all good predicates V for L, V has $\mathcal{C}$ witnesses of size at most $s(n)$.*[11]

The class $\mathsf{NTIME}[t(n)]$ has $\mathcal{C}$ witnesses of size $s(n)$ *if for every language $L \in \mathsf{NTIME}[t(n)]$, L has $\mathcal{C}$ witnesses of at most $s(n)$ size. The meaning of NEXP having $\mathcal{C}$ witnesses is defined analogously.*

The above definition of circuit witnesses allows, for every x, a different circuit C_x encoding a witness for x. We will also consider a stronger notion of *oblivious* witnesses, where a single circuit C_n encodes witnesses for all $x \in L$ of length n.

DEFINITION 3.3. *Let $L \in \mathsf{NTIME}[t(n)]$, and let $\mathcal{C}$ be a circuit class. L* has oblivious $\mathcal{C}$ witnesses of size $s(n)$ *if for every good predicate V for L, there is a $\mathcal{C}$-circuit family $\{C_n\}$ of size $s(n)$ such that for all $x \in \{0,1\}^\star$, if $x \in L$, then $V(x, tt(C_{|x|}(x, \cdot)))$ accepts.*[12]

$\mathsf{NTIME}[t(n)]$ has oblivious $\mathcal{C}$ witnesses *if every $L \in \mathsf{NTIME}[t(n)]$ has oblivious $\mathcal{C}$ witnesses. The meaning of NEXP having $\mathcal{C}$ witnesses is defined analogously.*

We establish an equivalence between the existence of small circuits for NEXP and small circuits for NEXP witnesses in both the oblivious and normal senses.

THEOREM 3.1. *Let $\mathcal{C}$ be a typical polynomial-size circuit class. The following are equivalent:*

(1) $\mathsf{NEXP} \subset \mathcal{C}$.

[11] **N.B.** For circuit classes $\mathcal{C}$ where the depth d and/or modulus m may be bounded, we also quantify this d and m simultaneously with the size parameter $s(n)$. That is, the depth, size, and modulus parameters are chosen prior to choosing an input, as usual.

[12] That is, the truth table of $C_{|x|}$ with x hard-coded is a valid witness for x.

(2) NEXP *has* $\mathcal{C}$ *witnesses.*
(3) NEXP *has oblivious* $\mathcal{C}$ *witnesses.*

Proof. (1) $\Rightarrow$ (2) Impagliazzo, Kabanets, and Wigderson [16] proved this direction for $\mathcal{C} = \mathsf{P}/\text{poly}$. The other cases of $\mathcal{C}$ were observed in prior work [37, 36].

(2) $\Rightarrow$ (3) Assume NEXP has $\mathcal{C}$ witnesses (implicitly, they are of polynomial size). Let $V(x, y)$ be a good predicate for an NEXP problem that (without loss of generality) accepts witnesses y of length exactly $2^{p(|x|)}$, for some polynomial $p(n)$. We will construct a $\mathcal{C}$-circuit family $\{C_n\}$ such that $x \in L$ if and only if $V(x, tt(C_{|x|}(x, \cdot)))$ accepts (recall $tt(C_{|x|}(x, \cdot))$ is the truth table of the circuit $C_{|x|}$ with x hard-coded and the remaining inputs free). The idea is to construct a new verifier that "merges" witnesses for all inputs of a given length into a single witness. (This theme will reappear throughout the paper.)

Let $x_1, \ldots, x_{2^n}$ be the list of strings of length n in lexicographical order. We define a new good predicate V' which takes a pair (x, q) where $x \in \{0,1\}^n$ and $q = 0, \ldots, 2^n$, along with y of length $2^{n+p(n)}$:

> $V'((x, q), y)$: *Accept* if and only if the following are all true:
> 1. $y = b_1 z_1 \cdots b_{2^{|x|}} z_{2^{|x|}}$, where for all $i = 1, \ldots, 2^{|x|}$, $b_i \in \{0,1\}$ and $z_i \in \{0,1\}^{2^{p(|x|)}}$;
> 2. exactly q of the b_i's are 1;
> 3. for all i's such that $b_i = 1$, $V(x_i, z_i)$ accepts;
> 4. for all i's such that $b_i = 0$, $z_i = 0^{2^{p(|x|)}}$.

V' runs in time exponential in $|x|$; by assumption, V' has $\mathcal{C}$ witnesses of polynomial size. Observe that the computation of V' does not depend on the input x, only the length $|x|$.

To obtain oblivious $\mathcal{C}$ witnesses for V, let q_n be the actual number of x of length n such that $x \in L(V)$. Then for every y'' such that $V'((x', q_n), y'')$ accepts, the string y'' must encode a valid witnesses z_i for every $x_i \in L(V)$. By assumption, there is a circuit $C_{(x', q_n)}$ such that $C_{(x', q_n)}(i)$ outputs the ith bit of y''. This circuit $C_{(x', q_n)}$ yields the desired witness circuit: indeed, the circuit $D_n(x, j) := C_{(x', q_n)}(x \circ j)$ (where $x \circ j$ denotes the concatenation of x and j as binary strings) prints the jth bit of a valid witness for x (or it prints 0 if $x \notin L(V)$).

(3) $\Rightarrow$ (1) Assume NEXP has oblivious $\mathcal{C}$ witnesses. Let M be a nondeterministic exponential-time machine. We want to give a $\mathcal{C}$-circuit family recognizing $L(M)$. First, we define a good predicate V_k:

> $V_k(x, y)$: For all circuits C of size $|x|^k + k$,
> If $tt(C)$ encodes an accepting computation history of $M(x)$, then
> *accept* if and only if the first bit of y is 1.
> End for
> *Accept* if and only if the first bit of y is 0.

By assumption, there is a k such that accepting computation histories of M on all length n inputs can be encoded with a single $\mathcal{C}$-circuit family of size at most $n^k + k$. For such a k, V_k will run in $2^{O(n^k)}$ time and will always find a circuit C encoding an accepting computation history of $M(x)$, when $x \in L(M)$. Therefore, $V_k(x, y)$ accepts if and only if

$$[(\text{first bit of } y = 1) \wedge (x \in L(M))] \vee [(\text{first bit of } y = 0) \wedge (x \notin L(M))].$$

Now, because V_k is a good predicate for the NEXP language $L(M)$, we can apply the assumption again to V_k itself, meaning there is a $\mathcal{C}$-circuit family $\{C_n\}$ encoding witnesses for V_k obliviously. This family can be easily used to compute $L(M)$: define the circuit D_n for n-bit instances of $L(M)$ to output the first bit of the witness encoded by $C_n(x, \cdot)$. ∎

Next, we prove a tight relation between witnesses for NE computations and constructive useful properties. (This equivalence will be useful for proving new consequences later.) Here, the typical circuit class $\mathcal{C}$ does not have to be polynomial-size bounded, and the size function $s(n)$ quantified below can be any reasonable function in the range $[n^2, 2^n/(2n)]$ (for example). We have two versions of the relation: one for constructive properties of Boolean functions (defined only on 2^n-bit strings) and one for polynomial-time algorithms (running on strings of all possible lengths).

THEOREM 3.2. *For all circuit-size functions $s(n) \in [n^2, 2^n/(2n)]$, the following are equivalent:*

1. *There is a $c \in (0,1]$ such that $\mathsf{NTIME}[2^{O(n)}]$ does not have $s(cn)$-size witness circuits from $\mathcal{C}$.*
2. *There are a $c \in (0,1]$ and a $\mathsf{P}/(\log n)$-computable property of Boolean functions that are useful against $\mathcal{C}$-circuits of size at most $s(cn)$.*[13]
3. *There are a $c \in (0,1]$ and a polynomial-time algorithm that are useful against $\mathcal{C}$-circuits of size at most $s(cn)$.*

Proof. (1) ⇒ (2) Suppose $\mathsf{NTIME}[2^{O(n)}]$ does not have $s(c \cdot n)$-size witness $\mathcal{C}$-circuits for some $c \in (0,1]$. Then there must be a good predicate V running in $\mathsf{TIME}[2^{dn}]$ for some $d \geq 1$ that does not have $s(c \cdot n)$-size witnesses. Hence there is an infinite subsequence of "bad" inputs $\{x'_i\}$ such that for all i, $x'_i \in L(V)$, but for every y such that $V(x'_i, y)$ accepts, y requires $s(c \cdot |x'_i|)$-size $\mathcal{C}$-circuits to encode.

To give a $\mathsf{P}/(\log n)$-computable property of Boolean functions $\mathcal{P}$ that is useful against $\mathcal{C}$-circuits, simply define $\mathcal{P}(f)$ with advice x'_i to be true if and only if $f : \{0,1\}^{d|x'_i|} \to \{0,1\}$ and $V(x'_i, f)$ accepts (when f is construed as a $2^{d|x_i|}$-bit string). The property $\mathcal{P}$ is clearly implementable in $\mathsf{P}/(\log n)$ (the advice can be anything when no appropriate x'_i exists), and for infinitely many input lengths ℓ, there is a string $x'_i \in L(V)$ of length ℓ such that every string y of length $2^{d\ell}$ accepted by $V(x'_i, y)$ requires $s(c \cdot \ell)$-size $\mathcal{C}$-circuits as a Boolean function. Hence for infinitely many ℓ, the property $\mathcal{P}$ is true of at least one Boolean function on $d\ell$ bits, and is false for all functions on $d\ell$ bits with $s(c \cdot \ell)$-size $\mathcal{C}$-circuits, for some fixed d.

(2) ⇒ (3) Let $\mathcal{P}$ be a property of Boolean functions with $\log n$ bits of advice, implemented by a polynomial-time algorithm $B(\cdot, \cdot)$, which is useful against $\mathcal{C}$-circuits of size $s(cn)$. We give a polynomial-time *algorithm* A with no advice that is useful against $\mathcal{C}$-circuits of size at most $s(cn)$. Again, let $x_1, \ldots, x_{2^\ell}$ be the ℓ-bit strings in lexicographical order in the following.

[13] For circuit classes $\mathcal{C}$ with depth bound d, this d will be universally quantified after c. So, for example, there is a c such that for all constant d, $\mathsf{NTIME}[2^{O(n)}]$ does not have $s(cn)$-size depth-d $\mathsf{AC}^0[6]$ witnesses if and only if there is a c such that for all d, there is a polynomial-time algorithm useful against depth-d $\mathsf{AC}^0[6]$ circuits of size $s(cn)$.

> $A(y)$: If y does not have the form $z01^k$, with $|z| = 2^\ell$, for some $k = 0, \dots, 2^\ell - 1$ and ℓ, then *reject*. Otherwise, compute k by counting the trailing 1's at the end of y, and *accept* if and only if $B(z, x_k)$ accepts.

Let ℓ be an integer such that the property $\mathcal{P}$, with the appropriate advice x_k of length $d\ell$, is useful for functions on ℓ bits. Then for every (z, x_k) pair accepted by the algorithm B, the Boolean function defined by z of length 2^ℓ is not computable with $s(c \cdot \ell)$-size $\mathcal{C}$-circuits.

Observe that, for each ℓ, and every possible $k = 0, \dots, 2^\ell - 1$, there is *exactly* one input length, namely $n = 2^\ell + k + 1$, for which the input x_k of length ℓ will be considered, along with all possible z's of length 2^ℓ. Therefore, on those infinitely many input lengths n for which the corresponding input x_k of length ℓ equals some bad input x'_j, A is useful against size-$s(c \cdot \ell)$ circuits from $\mathcal{C}$.

(3) $\Rightarrow$ (1) Let A be a poly(n)-time algorithm that is useful against $s(c \cdot n)$-size $\mathcal{C}$-circuits for some fixed constant c. In the following, let x_k be the kth string in the lexicographical ordering of strings of length $|x_k|$. Define a machine:

> $M(x_k, T)$: If $|T| \neq 2^{|x_k|}$, *reject.* If $k > |T|/2$, *reject.*
> Otherwise, strip the last $k-1$ bits from T,
> obtaining a string T' of length $2^{|x_k|} - (k-1)$.
> *Accept* if and only if $A(T')$ accepts.

Now define $L = \{x \mid (\exists\ T : |T| = 2^{|x|})[M(x, T) \text{ accepts}]\}$. Note that $L \in \mathsf{NTIME}[2^{O(n)}]$, and that M is a good verifier for L. By our assumption that A is a polytime useful algorithm, there are infinitely many integers ℓ such that

(1) A accepts at least one string y_ℓ of length ℓ, and
(2) if A accepts y_ℓ of length ℓ, then the Boolean function corresponding to y_ℓ (possibly obtained by padding zeroes to the end of y_ℓ) has circuit complexity greater than $s(c \cdot \ell)$.

For each such ℓ, let j_ℓ be the smallest integer such that $2^{j_\ell} \geq \ell$. Define $i_\ell := 2^{j_\ell} - \ell$; that is, $i_\ell \in \{0, 1, \dots, 2^{j_\ell - 1} - 1\}$ equals the number of zeroes needed to pad y_ℓ so that the length becomes a power of two. In the following, let $x_1, \dots, x_{2^{j_\ell}}$ be the list of all j_ℓ-bit strings in lexicographical order.

Then, $M(x_{i_\ell}, T)$ accepts if and only if $|T| = 2^{j_\ell}$, $T = y_\ell z$ for some z with $|z| = i_\ell$, and $A(y_\ell)$ accepts. For infinitely many ℓ, each such y_ℓ has the property that $y_\ell 0^{i_\ell}$ has circuit complexity greater than $s(c \cdot j_\ell)$; therefore each of the strings T such that $M(x_{i_\ell}, T)$ accepts must have circuit complexity greater than $s(c \cdot j_\ell) - j_\ell^{1+o(1)}$ as well, by Proposition 1. So there is an infinite sequence of inputs $\{x'_\ell\}$ such that all strings x'_ℓ are in L, and all witnesses of x'_ℓ have circuit complexity greater than $s(c \cdot |x'_\ell|) - |x'_\ell|^{1+o(1)}$. Hence L is a language in $\mathsf{NTIME}[2^{O(n)}]$ that does not have $(s(c \cdot n) - n^{1+o(1)})$-size witnesses. Since $s(n) \geq n^2$, we have completed the proof of this direction. ∎

Using complete languages for NEXP, one can obtain an explicit property in P that is useful against $\mathcal{C}$-circuits if there is *any* constructive useful property. This universality means that if there are multiple constructive properties that are useful against various circuit-size functions, then there is one constructive property useful against all these size functions.

THEOREM 3.3. *Let $\{s_k(n)\}$ be an infinite family of functions such that for all k, there is a polynomial-time algorithm P_k (or polynomial-time property of Boolean functions with $\log n$ bits of advice) that is useful against all $\mathcal{C}$-circuits of $s_k(n)$ size. Then there is a single P-computable algorithm $P^\star$ such that, for all k, there is a $c > 0$ such that $P^\star$ is useful against all $\mathcal{C}$-circuits of $s_k(cn)$ size.*[14]

Proof. Let $b(n)$ denote the nth string of $\{0,1\}^\star$ in lexicographical order. The SUCCINCT HALTING problem consists of all triples $\langle M, x, b(n)\rangle$ such that the nondeterministic TM M accepts x within at most n steps. Define the following algorithm:

> HISTORY(y): Compute $z = b(|y|)$. If z does not have the form $\langle M, x, b(n)\rangle$, reject. *Accept* if and only if there is a prefix y' of y with length equal to a power of two such that y' encodes an accepting computation history to $z \in$ SUCCINCTHALTING.

Observe that HISTORY is implementable in polynomial time. The theorem follows from the next claim.

CLAIM 3.1. HISTORY *is useful against $\mathcal{C}$-circuits of size $s(cn)$ for some $c > 0$ if and only if there is some polynomial-time algorithm (possibly with $\log n$ bits of advice) that is useful against $\mathcal{C}$-circuits of size $s(n)$.*

To see why Theorem 3.3 follows, observe that if we have infinitely many properties P_k, each of which is useful against $\mathcal{C}$-circuits of $s_k(n)$ size, then for every k, HISTORY will be useful against $s_k(n)$-size $\mathcal{C}$-circuits.

One direction of the claim is obvious. For the other, suppose there is a polynomial-time property with $\log n$ bits of advice (or a polynomial-time algorithm) useful against $\mathcal{C}$-circuits of size $s(n)$. By Theorem 3.2, $\mathsf{NTIME}[2^{O(n)}]$ does not have $s(dn)$-size witnesses from $\mathcal{C}$ for some constant d. Let V be a good predicate running in time 2^{kn} that does not have $s(dn)$-size $\mathcal{C}$ witnesses, and let M be the corresponding nondeterministic machine which, on x, guesses a y and accepts if and only if $V(x, y)$ accepts. It follows that there are infinitely many instances of SUCCINCTHALTING of the form $\langle M, x, b(2^{k|x|})\rangle$ that do not have $\mathcal{C}$ witnesses of size $s(cn)$ for some constant c. Therefore, there are infinitely many $z_i = \langle M_i, x_i, n_i\rangle$ in SUCCINCTHALTING, where every accepting computation history y' of $M_i(x_i)$ has greater than $s(cn)$-size $\mathcal{C}$-circuit complexity. Then for all n such that $z_i = b(n)$ for some i, there is a y of length n such that HISTORY(y) accepts but for all y'' which encode functions with $\mathcal{C}$-circuits of $s(cn)$ size, HISTORY(y'') rejects (by Proposition 1; note that y'' has length equal to a power of two). Hence HISTORY is useful against $\mathcal{C}$-circuits of size $s(cn)$. This concludes the proof of the theorem. ∎

Putting everything together, we obtain Theorems 1.1 and 1.2.

Proof of Theorems 1.1 *and* 1.2. We prove Theorem 1.2; the proof of Theorem 1.1 is analogous, and we add parenthetical remarks below about how to prove it. Let $\mathcal{C}$ be a typical class (of polynomial-size circuits). By Theorem 3.1, we have $\mathsf{NEXP} \not\subset \mathcal{C}$ if and only if for every k, NEXP does not have $\mathcal{C}$ witnesses of n^k size.

Setting $s(n) = n^k$ for arbitrary k in Theorem 3.2, we infer that for every k, we have the equivalence: NEXP does not have $\mathcal{C}$ witnesses of n^k size if and only if there are $c > 0$ and a polynomial-time algorithm that is useful against all $\mathcal{C}$-circuits of size

[14] For depth-bounded/modulus-bounded circuit classes $\mathcal{C}$, an analogous statement holds where we quantify not only over k but also the depth d and modulus m.

at most $(cn)^k$. (Note that Theorem 3.2 also implies an equivalence between the above two conditions and the existence of a $\mathsf{P}/(\log n)$-computable property useful against $\mathcal{C}$-circuits of size $(cn)^k$.)

Applying Theorem 3.3, we conclude that $\mathsf{NEXP} \not\subset \mathcal{C}$ if and only if there is a polynomial-time algorithm such that, for all k, it is useful against all $\mathcal{C}$-circuits of size at most n^k. ∎

4. New ACC lower bounds. In this section, we prove new lower bounds against ACC^0. Our approach uses a new nondeterministic simulation of randomized computation (assuming small circuits for ACC^0). The simulation itself uses several ingredients. First, we prove an exponential-size lower bound on the sizes of ACC^0 circuits encoding *witnesses* for $\mathsf{NTIME}[2^{O(n)}]$. (Recall that, for NEXP, the best known ACC^0-size lower bounds are only "third-exponential" [36].) Second, we use the connection between witness-size lower bounds and constructive useful properties of Theorem 3.2. The third ingredient is a well-known hardness-randomness connection: from a constructive useful property, we can nondeterministically guess a hard function, verify its hardness using the property, and then use the hard function to construct a pseudorandom generator. (Here, we will need to make an assumption like $\mathsf{P} \subset \mathsf{ACC}^0$, as it is not known how to convert hardness into pseudorandomness in the ACC^0 setting [34].)

4.1. Exponential lower bounds for encoding NEXP witnesses.

REMINDER OF THEOREM 1.3. *For all d, m there is an $\varepsilon = 1/m^{\Theta(d)}$ such that $\mathsf{NTIME}[2^{O(n)}]$ does not have 2^{n^ε}-size d-depth $\mathsf{AC}^0[m]$ witnesses.*[15]

The proof is quite related in structure to the $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$ proof, so we will merely sketch how it is different.

Proof (sketch). Assume $\mathsf{NTIME}[2^{O(n)}]$ has 2^{n^ε}-size ACC^0 witnesses, for all $\varepsilon > 0$. We will show that the earlier framework [36] can be adapted to still establish a contradiction. First, observe the assumption implies that $\mathsf{TIME}[2^{O(n)}]$ has 2^{n^ε}-size ACC^0 circuits. (The proof is similar to the proof of Theorem 1.1: for any given exponential-time algorithm A, one can set up a good predicate that only accepts its input of length n if the witness is a truth table for the 2^n-bit function computed by A on n-bit inputs. Then, a witness circuit for this x is a circuit for the entire function on n bits.) Therefore (by Lemma 3.1 in [36]) there is a nondeterministic 2^{n-n^δ} time algorithm A (where δ depends on the depth and modulus of ACC^0 circuits for circuit evaluation) that, given any circuit C of size $n^{O(1)}$ and n inputs, generates an equivalent ACC^0 circuit C' of 2^{n^ε} size, for all $\varepsilon > 0$. (More precisely, there is some computation path on which A generates such a circuit, and on every path, it either prints such a circuit or outputs *fail*.)

The rest of the proof is analogous to prior NEXP lower bounds [36]; we sketch the details for completeness. Our goal is to simulate every $L \in \mathsf{NTIME}[2^n]$ in nondeterministic time 2^{n-n^δ}, which will contradict the nondeterministic time hierarchy of Žák [38]. Given an instance x of L, we first reduce L to the NEXP-complete SUCCINCT 3SAT problem using an efficient polynomial-time reduction. This yields an unrestricted circuit D of size $n^{O(1)}$ and $n + O(\log n)$ inputs with truth table equal to a formula F, such that F is satisfiable if and only if $x \in L$. We run algorithm A on D to obtain an equivalent 2^{n^ε}-size ACC^0 circuit D'. Then we guess a 2^{n^ε}-size ACC^0 circuit E with truth table equal to a satisfying assignment for F. (If $x \in L$, then such

[15]The $m^{\Theta(d)}$ factor arises from the ACC-SAT algorithm in [36], which in turn comes from Beigel and Tarui's simulation of ACC^0 in SYM-AND [9].

a circuit exists, by assumption.) By combining copies of D' and copies of E, we can obtain a single ACC^0 circuit C with $n + O(\log n)$ inputs which is unsatisfiable if and only if E encodes a satisfying assignment for F. By calling a nontrivial satisfiability algorithm for ACC, we get a nondeterministic $2^{n-n^{\delta}}$ time simulation for every L, a contradiction. ∎

Applying Theorem 3.2 and its corollary to the lower bound of Theorem 1.3, we can conclude the following.

Corollary 4.1. *For all d, m, there are an $\varepsilon = 1/m^{\Theta(d)}$ and a P-computable property that is useful against all depth-d $\mathsf{AC}^0[m]$ circuits of size at most $2^{n^{\varepsilon}}$.*

Hence there is an efficient way of distinguishing some functions from all functions computable with subexponential-size ACC^0 circuits. Let CAPP be the problem: *given a circuit C, output $p \in [0, 1]$ satisfying*

$$|Pr_x[C(x) = 1] - p| < 1/6.$$

That is, we wish to approximate the acceptance probability of C to within $1/6$. We can give a quasi-polynomial time nondeterministic algorithm for CAPP, assuming P is in quasi-polynomial-size ACC^0.

Theorem 4.1. *Suppose P has ACC^0 circuits of size $n^{\log n}$. Then there is a constant c such that for infinitely many sizes s, CAPP for size s circuits is computable in nondeterministic $2^{(\log s)^c}$ time.*

Theorem 4.1 is a surprisingly strong consequence: given that $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$, one would expect only a $2^{O(n^{\varepsilon})}$-time algorithm for CAPP, with n^{ε} bits of advice. (Indeed, from the results of Impagliazzo, Kabanets, and Wigderson [16] one can derive such an algorithm, assuming $\mathsf{P} \subseteq \mathsf{ACC}^0$.)

Before proving Theorem 4.1, we first extend Theorem 1.3 a little bit. Recall a *unary language* is a subset of $\{1^n \mid n \in \mathbb{N}\} \subseteq \{0,1\}^{\star}$. The proof of Theorem 1.3 also has the following consequence.

Corollary 4.2. *If P has ACC^0 circuits of $n^{\log n}$ size, then for all d, m there is an ε such that there are unary languages in $\mathsf{NTIME}[2^n]$ without $2^{n^{\varepsilon}}$-size d-depth $\mathsf{AC}^0[m]$ witnesses.*

Proof. The tight nondeterministic time hierarchy of Žák [38] holds also for unary languages. That is, there is a unary $L \in \mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n^{10}]$. So assume (for a contradiction to this hierarchy) that all unary languages in $\mathsf{NTIME}[2^n]$ have $2^{n^{\varepsilon}}$-size witnesses, for every $\varepsilon > 0$. This says that, for every good predicate V for every unary language $L \in \mathsf{NTIME}[2^n]$, every $1^n \in L$ has a witness y with $2^{n^{\varepsilon}}$-size circuit complexity. Choose a predicate V that reduces a given unary L to a Succinct3SAT instance and then checks that its witness is a SAT assignment to the instance; by assumption, such SAT assignments must have circuit complexity at most $2^{n^{\varepsilon}}$, for almost all n. By guessing such a circuit and assuming P has $n^{\log n}$-size ACC^0 circuits, the remainder of the proof of Theorem 1.3 goes through: the simulation of arbitrary L in $\mathsf{NTIME}[2^{n-n^{\delta}}]$ works and yields the contradiction. ∎

Corollary 4.2 allows us to strengthen Corollary 4.1, to yield a "nondeterministically constructive" and useful property against ACC^0. Informally, having a *unary* language without small witness circuits allows us to obtain a derandomization *without* advice, as there is no need to store a "hard" input for a given input length. In particular, the unconditional lower bound of Corollary 4.2 can be used to build an efficient "hardness test" for ACC^0 circuit complexity, which is then used with a pseudorandom generator to solve CAPP by guessing a hard function and verifying it with the test. This basic idea seems to have originated with [20, 19].

Proof of Theorem 4.1. First we claim that if P has $n^{\log n}$-size ACC^0 circuits, then there is a $d^\star$ and $m^\star$ such that every Boolean function f with unrestricted circuits of size S has depth-$d^\star$ $\mathsf{AC}^0[m^\star]$ circuits of size at most $S^{\log S}$. To see this, consider the Circuit Evaluation problem: *given a circuit C and an input x, does $C(x) = 1$?* Assuming P is in $n^{\log n}$ ACC^0, this problem has a depth-$d^\star$ $AC^0[m^\star]$ circuit family $\{D_n\}$ of $n^{\log n}$ size, for some fixed $d^\star$ and $m^\star$. Therefore, by plugging in the description of any circuit C of size S into the input of the appropriate ACC^0 circuit $D_{O(S)}$, we get an ACC^0 circuit of fixed modulus and depth that is equivalent to C and has size $O(S^{\log S})$.

By Corollary 4.2, there are an ε and a unary L in $\mathsf{NTIME}[2^n]$ that does not have 2^{n^ε}-size $\mathsf{AC}^0[m^\star]$ witnesses of depth $d^\star$. By the previous paragraph (and assuming P is in $n^{\log n}$-size ACC^0), it follows that L does not have witnesses encoded with $2^{n^{\varepsilon/2}}$-size unrestricted circuits. (Letting $S^{\log S} = 2^{n^\varepsilon}$, we find that $S = 2^{n^{\varepsilon/2}}$.) Let V be a good predicate for L that lacks such witnesses, and let g be the constant in the pseudorandom generator of Theorem 2.1. Consider the nondeterministic algorithm P which, on input 1^s, sets $n = (g \log s)^{2/\varepsilon}$, guesses a string Y of 2^n length, and outputs Y if $V(1^n, Y)$ accepts (otherwise, P outputs *reject*). For infinitely many s, $P(1^s)$ nondeterministically generates strings Y of $2^{(g \log s)^{2/\varepsilon}}$ length that do not have $s^g = 2^{n^{\varepsilon/2}}$-size circuits: as there is an infinite set of $\{n_i\}$ such that all witnesses to 1^{n_i} have circuit complexity at least $2^{(n_i)^{\varepsilon/2}}$, there is an infinite set $\{s_i\}$ such that $P(1^{s_i})$ computes $n_i = (g \log s_i)^{2/\varepsilon}$ and generates Y which does not have $(s_i)^g = 2^{(n_i)^{\varepsilon/2}}$-size circuits.

Given a circuit C of size s, our nondeterministic simulation runs P to generate Y. (If P rejects, the simulation *rejects*.) Applying Theorem 2.1, Y can be used to construct a poly($|Y|$)-time PRG $G(Y, \cdot) : \{0,1\}^{g \log |Y|} \to \{0,1\}^s$ which fools circuits of size s. By trying all $|Y|^g \leq 2^{O((\log s)^{2/\varepsilon})}$ inputs to G_Y, we can approximate the acceptance probability of a size-s circuit in $2^{O((\log s)^{2/\varepsilon})}$ time. As ε depended only on $d^\star$ and $m^\star$, which are both constants, we can set $c = 3/\varepsilon$ to complete the proof. ∎

4.2. Slightly stronger ACC lower bounds. Now we turn to proving lower bounds for the classes $\mathsf{NE} \cap io\text{-}\mathsf{P}$ and $\mathsf{NE}/1 \cap \mathsf{coNE}/1$. We will need an implication between circuits and Merlin–Arthur simulations that extends Babai et al. [7].

Theorem 4.2 (Lemma 8 in [26]). *Let $g(n) > 2^n$ and $s(n) \geq n$ be increasing and time constructible. There is a constant $c > 1$ such that* $\mathsf{TIME}[2^{O(n)}] \subseteq \mathsf{SIZE}[s(n)] \Longrightarrow \mathsf{TIME}[g(n)] \subseteq \mathsf{MATIME}[s(3 \log g(n))^c]$.

That is, if we assume exponential time has $s(n)$-size circuits, we can simulate even larger time bounds with Merlin–Arthur games. This follows from the proof of $\mathsf{EXP} \subset \mathsf{P}/\mathrm{poly} \Longrightarrow \mathsf{EXP} = \mathsf{MA}$ [7] combined with a padding argument.

Reminder of Theorem 1.4. $\mathsf{NE} \cap io\text{-}\mathsf{P}$ *and* $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ *do not have* ACC^0 *circuits of $n^{\log n}$ size.*

Proof. The easy proof for $\mathsf{NE} \cap io\text{-}\mathsf{P}$ was already given in the introduction.

Suppose $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ has $n^{\log n}$-size ACC^0 circuits. We wish to derive a contradiction. Of course the assumption implies that $\mathsf{TIME}[2^{O(n)}]$ has $n^{\log n}$-size circuits as well. Applying Theorem 4.2 with $g(n) = 2^{n^{2\log n}}$ and $s(n) = n^{\log n}$, we have

$$\mathsf{TIME}[2^{n^{2\log n}}] \subseteq \mathsf{MATIME}[n^{O(\log^3 n)}].$$

Let L be an arbitrary language in $O(2^{n^{2\log n}})$ time. As $\mathsf{TIME}[2^{n^{2\log n}}]$ is closed under complement, an analogous argument (applied to any machine accepting the complement of a given $\mathsf{TIME}[2^{n^{2\log n}}]$ language) implies

$$\mathsf{TIME}[2^{n^{2\log n}}] \subseteq \mathsf{coMATIME}[n^{O(\log^3 n)}].$$

Thus, both L and $\overline{L}$ have Merlin–Arthur games running in $n^{O(\log^3 n)}$ time.

By Theorem 4.1 and assuming that P has ACC^0 circuits of size $n^{\log n}$, there is a constant c and a pseudorandom generator (PRG) with the following properties: for infinitely many circuit sizes s, the generator nondeterministically guesses a string Y of length $2^{(\log s)^c}$, verifies Y in poly($|Y|$) deterministic time with a useful property P, and then uses Y to construct a PRG that runs in poly($|Y|$) time deterministically over poly($|Y|$) different seeds. The poly($|Y|$) outputs of length s can then be used to correctly approximate the acceptance probability of any size s circuit.

We can use this generator to fool Merlin–Arthur games, as well as co-Merlin–Arthur games, as follows.

For the language L, take an $n^{O(\log^3 n)}$-size circuit C encoding the predicate in a given Merlin–Arthur game of that length: C takes an input x, Merlin's string of length $n^{O(\log^3 n)}$, and Arthur's string of length $n^{O(\log^3 n)}$ and outputs a bit. Our nondeterministic simulation first guesses Merlin's string m and then runs the PRG, which guesses a Y and verifies that Y is a hard function; if the verification fails, we *reject*.

Next, the simulation uses the PRG on $C(x, m, \cdot)$ to simulate Arthur's string and the final outcome, accepting if and only if the majority of strings generated by the PRG lead to acceptance. On infinitely many input lengths, the simulation of the Merlin–Arthur game will be "faithful" in the sense that the PRG simulating Arthur will work as intended.

By adding a bit of advice $y_n \in \{0,1\}$ to encode whether or not the PRG is successful for a given input length n, we can simulate an arbitrary Merlin–Arthur game for L (and its complement $\overline{L}$) running in $n^{O(\log^3 n)}$ time infinitely often, in

$$\mathsf{NTIME}[n^{\log^d n}]/1 \cap \mathsf{coNTIME}[n^{\log^d n}]/1$$

for some constant d.

In particular, we define two nondeterministic machines N and N' which take one advice bit as follows. If the advice bit is 0, both simulations *reject*. Otherwise, N attempts to run the Merlin–Arthur simulation of C, and N' attempts to Merlin–Arthur simulate the complement language $\overline{L}$, as described above. When the advice bits are assigned appropriately on all input lengths, the nondeterministic N accepts a language $L' \in \mathsf{NTIME}[n^{\log^d n}]/1$ which agrees with L on infinitely many input lengths $n_1, n_2, \ldots$ and is empty on the other input lengths. For the complement language $\overline{L}$, the machine N' accepts a language $L'' \in \mathsf{NTIME}[n^{\log^d n}]/1$ which agrees with $\overline{L}$ on the *same* list of input lengths $n_1, n_2, \ldots$ as above and is empty on the other input lengths. It follows that $\overline{L''} = L'$, and therefore we have

$$L' \in \mathsf{NTIME}[n^{\log^d n}]/1 \cap \mathsf{coNTIME}[n^{\log^d n}]/1$$

such that for all possible input lengths n, either $L' \cap \{0,1\}^n = \varnothing$ (for those input lengths where the advice is set to 0) or $L' \cap \{0,1\}^n = L \cap \{0,1\}^n$ (for infinitely many n).

In summary, given a language L recognizable in $O(2^{n^{2\log n}})$ time, there is a language L' in the class $\mathsf{NTIME}[n^{\log^d n}]/1 \cap \mathsf{coNTIME}[n^{\log^d n}]/1$ which agrees with L on infinitely many input lengths. Therefore

$$\mathsf{TIME}[2^{n^{2\log n}}] \subseteq io\text{-}(\mathsf{NTIME}[n^{\log^d n}]/1 \cap \mathsf{coNTIME}[n^{\log^d n}]/1).$$

Assuming every language in $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ has circuits of size $n^{\log n}$, it follows that every language in the class $io\text{-}(\mathsf{NE}/1 \cap \mathsf{coNE}/1)$ has circuits of size $n^{\log n}$, for infinitely many input lengths n. Therefore

$$\mathsf{TIME}[2^{n^{2\log n}}] \subset io\text{-}\mathsf{SIZE}[n^{\log n}].$$

But this implies a contradiction: for almost *every* n, by simply enumerating all $n^{\log n}$-size circuits and computing their 2^n-bit truth tables, we can determine the lexicographically first Boolean function on n bits which does not have $n^{\log n}$-size circuits, in $O(2^{n^{2\log n}})$ time. ∎

We conclude the section by sketching how the above argument can be recast in a more generic form, as a connection between SAT algorithms and circuit lower bounds.

Reminder of Theorem 1.5. *Let $\mathcal{C}$ be typical. Suppose the satisfiability problem for $n^{O(\log^c n)}$-size $\mathcal{C}$-circuits can be solved in $O(2^n/n^{10})$ time, for all constants c. Then* $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ *does not have $n^{\log n}$-size $\mathcal{C}$-circuits.*

Proof (sketch). Suppose that satisfiability for $\mathcal{C}$-circuits of $n^{O(\log^c n)}$ size is in $O(2^n/n^{10})$ time (for all c), and that $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ has $n^{\log n}$-size circuits.

By the proof of Theorem 4.1, assuming P has $n^{\log n}$-size $\mathcal{C}$ circuits, for all $\varepsilon > 0$, we obtain a nondeterministic algorithm N running in $2^{2^{O(\log^\varepsilon s)}}$ time on all circuits of size s (for infinitely many s) and outputs a good approximation to the given circuit's acceptance probability. In particular, from the assumptions we can derive a unary language computable in $\mathsf{NTIME}[2^n]$ that does not have witness circuits of $n^{\log^c n}$ size, for every c. This can be used to obtain a nondeterministic algorithm N as in Theorem 4.1, by setting $s = n^{O(\log^c n)}$, solving for $n = 2^{O((\log s)^{1/(c+1)})}$, and then running the nondeterministic algorithm N in $2^{O(n)} \le 2^{2^{O(\log^\varepsilon s)}}$ time, where $\varepsilon \le 1/(c+1)$.

By the same argument as in the proof of Theorem 1.4, we obtain

$$\mathsf{TIME}[2^{n^{2\log n}}] \subseteq (\mathsf{MATIME} \cap \mathsf{coMATIME})[n^{O(\log^3 n)}].$$

By applying algorithm N to circuits of size $s = n^{O(\log^3 n)}$, setting $\varepsilon \ll 1/4$, and using a bit of advice to encode when the PRG works, we obtain

$$(\mathsf{MATIME} \cap \mathsf{coMATIME})[n^{O(\log^3 n)}] \subseteq io\text{-}(\mathsf{NTIME}[2^{O(n)}]/1 \cap \mathsf{coNTIME}[2^{O(n)}]/1).$$

But the latter class is in $io\text{-}\mathsf{SIZE}[n^{\log n}]$ by assumption, so we conclude a contradiction as in Theorem 1.4. ∎

Reminder of Theorem 1.6. *Suppose we can approximate the acceptance probability of any given $n^{O(\log^c n)}$-size circuit (with fan-in two and arbitrary depth) on n inputs to within $1/6$, for all c, in $O(2^n/n^{10})$ time (even nondeterministically). Then* $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ *does not have $n^{\log n}$-size circuits.*

Proof (sketch). For the lower bound arguments given in this section, an algorithm which can approximate the acceptance probability of a given $n^{O(\log^c n)}$-size circuit can be applied in place of a faster SAT algorithm [37, 36, 33]. That is, from the hypothesis of the theorem we can derive exponential-size witness circuit lower bounds for NEXP (as in Theorem 1.3) and infinitely often correct PRGs against general circuits (as in Theorem 4.1). Therefore the proofs of Theorem 1.4 and consequently Theorem 1.5 also carry over under the hypothesis of the theorem. ∎

5. Natural properties and derandomization. In this section, we characterize (the nonexistence of) natural properties as a particular sort of derandomization problem and exhibit several consequences.

Let $\mathsf{ZPE} = \mathsf{ZPTIME}[2^{O(n)}]$, i.e., the class of languages solvable in $2^{O(n)}$ time with randomness and no error (the machine can output **?**, or *don't know*). $\mathsf{RE} = \mathsf{RTIME}[2^{O(n)}]$ is its one-sided-error equivalent. Analogously to Definition 3.1, we define a witness notion for ZPE as follows.

Definition 5.1. *Let $L \in \mathsf{ZPE}$. A* ZPE predicate *for L is a procedure $M(x, y)$ that runs in time $2^{O(|x|)}$ on inputs y of length $2^{c|x|}$ for some constant c, such that for every x and y, the following hold:*

- *The output of $M(x, y)$ is in the set $\{1, 0, \mathbf{?}\}$.*
- *$x \in L \implies \Pr_{y \in \{0,1\}^{2^{c|x|}}}[M(x, y)$ outputs $1] \geq 2/3$, and for all y of length $2^{c|x|}$, $M(x, y) \in \{1, \mathbf{?}\}$.*
- *$x \notin L \implies \Pr_{y \in \{0,1\}^{2^{c|x|}}}[M(x, y)$ outputs $0] \geq 2/3$, and for all y of length $2^{c|x|}$, $M(x, y) \in \{0, \mathbf{?}\}$.*

ZPE has $\mathcal{C}$ seeds *if for every* ZPE *predicate M, there is a k such that for all x, there is a $\mathcal{C}$-circuit C_x of size at most $|x|^k + k$ such that $M(x, tt(C_x)) \neq \mathbf{?}$.*[16]

ZPE has $\mathcal{C}$ seeds for infinitely many input lengths *if for every* ZPE *predicate M, there is a k such that for infinitely many n and for all x of length n, there is a $\mathcal{C}$-circuit C_x of size at most $n^k + k$ such that $M(x, tt(C_x)) \neq \mathbf{?}$.*

That is, $\mathcal{C}$ seeds for ZPE are succinct encodings of strings that lead to a decision by the algorithm. Analogously, we can define RE *predicates* and the notion of RE *having $\mathcal{C}$ seeds*: RE predicates will accept with probability at least 2/3 when $x \in L$, but reject with probability 1 when $x \notin L$. Hence, when RE has $\mathcal{C}$ seeds, we only require $x \in L$ to have small circuits C_x encoding witnesses.

Succinct seeds for zero-error computation are closely related to uniform natural properties as follows.

Reminder of Theorem 1.7. *Let $\mathcal{C}$ be a typical polynomial-size circuit class. The following are equivalent:*

1. *There are no* P*-natural properties useful (resp., ae-useful*[17]*) against $\mathcal{C}$.*
2. ZPE *has $\mathcal{C}$ seeds for almost all (resp., infinitely many) input lengths.*

The intuition is that, given a P-natural useful property, its probability of acceptance can be amplified (at a mild cost to usefulness), yielding a ZPE predicate which accepts random strings with decent probability but still lacks small seeds. In the other direction, suppose a ZPE predicate has "bad" inputs that can't be decided using small

[16] For circuit classes where the depth d and/or modulus m may be bounded, we also quantify this d and m simultaneously with the size parameter k. That is, the depth, size, and modulus parameters are chosen prior to choosing the circuit family, as usual.

[17] Here, *ae-useful* is just the "almost-everywhere useful" version, where the property is required to distinguish random functions from easy ones on almost every input length.

circuits encoding seeds. This implies that a "hitting set" of exponential-length strings, sufficient for deciding all inputs of a given length, must have high circuit complexity—otherwise, all strings in the set would have low circuit complexity (by Lemma 2.1), but at least one such string decides even a bad input. Checking for a hitting set is then a P-natural, useful property.

Proof of Theorem 1.7. ($\neg(1) \Rightarrow \neg(2)$) Suppose there is a P-natural property which is ae-useful (resp., useful) against $\mathcal{C}$. For some $c, d \geq 1$, this is an n^c-time algorithm A such that, for almost all n (resp., infinitely many n), A accepts at least a $1/2^{d \log n} = 1/n^d$ fraction of n-bit inputs, for $n = 2^\ell$, and for almost all $n = 2^\ell$ (resp., for infinitely many n) and all c, A rejects all n-bit inputs representing truth tables of $(\log n)^c$-size $\mathcal{C}$-circuits.

Let $\varepsilon > 0$ be sufficiently small. Define an algorithm V:

> $V(x, z)$: Let $k = |x|$. If $k = 0$, then output $\mathbf{1}$. If $|z| \neq 2^{(d+2)k}$, then output $\mathbf{0}$.
> Partition z into $t = 2^{(d+1)k}$ strings $z_1, \ldots, z_t$ each of length 2^k.
> If $A(z_i)$ accepts for some i, then output 1; else, output $\mathbf{?}$.

We claim that V is a ZPE predicate for the language $L = \{0,1\}^\star$ with constant $c = d + 2$. Let x be an arbitrary bit string of length k where $k \geq 1$, and consider a bit string z of length $2^{(d+2)k}$ chosen uniformly at random. All z_i of length 2^k are independent random strings, and by assumption, A accepts at least $1/|z_i|^d = 1/2^{dk}$ strings of that length. The probability that all z_i are among the $(1 - 1/2^{dk})$ fraction of strings of length 2^k rejected by A is at most $(1 - 1/2^{dk})^{2^{(d+1)k}} \leq \exp(-2^k) < 1/3$ (for $k \geq 1$). Therefore the probability V accepts a random z of the appropriate length is at least $2/3$.

By construction, V accepts (x, z) precisely when $|z| = 2^{(d+2)|x|}$ and some z_i of length $2^{d|x|}$ is accepted by A. Hence for almost every k (resp., infinitely many k), when $V(x, z)$ accepts on z of length $2^{(d+2)k}$, some substring z_i of length 2^k has $\mathcal{C}$-circuit complexity at least $(\log 2^k)^c \geq \Omega(\log^c |z|)$. Therefore by Lemma 2.1, z itself has $\mathcal{C}$-circuit complexity at least $\Omega((\log |z|)^c - (\log |z|)^{1+o(1)})$. As this holds for every c, the predicate V does not have $\mathcal{C}$ seeds infinitely often (resp., almost everywhere).

($\neg(2) \Rightarrow \neg(1)$) Suppose there is a ZPE predicate V that does not have $\mathcal{C}$ seeds almost everywhere (resp, infinitely often). This means that, for all k and for infinitely many (resp., almost all) input lengths n_i, there is some input x of length n_i such that, for every string r of length 2^{cn_i} satisfying $V(x, r) \neq \mathbf{?}$, the $\mathcal{C}$-circuit complexity of r is at least $(cn_i)^k$. (Note that the constant c depends only on V.) Define a new predicate V' as follows, intended to be executed on the inputs x with lengths in $\{n_i\}$:

> $V'(x, r)$: If $|r| \neq 2^{\ell + c|x|}$, where ℓ is the smallest integer such that $2|x| \leq 2^\ell$, then *reject*.
> Partition r into 2^ℓ strings $\{r_i\}$ of length $2^{c|x|}$ each.
> *Accept* if and only if $V(x, r_i) \neq \mathbf{?}$ for some i.

For those inputs x of length n_i, any r accepted by $V'(x, r)$ does not have circuits of size n_i^k, due to Proposition 1 and the fact that such an r contains a substring r_i such that $V(x, r_i)$ accepts; hence r_i has circuit complexity at least n_i^k. By standard probabilistic arguments and our choice of ℓ, it is likely that the string r encodes a hitting set for all inputs of length n_i, i.e.,

$$\Pr_{r \in \{0,1\}^{2^{cn_i+\ell}}} \left[(\exists x \in \{0,1\}^{n_i})(\forall\ i = 1, \ldots, 2^{\ell})[V(x, r_i) = \mathbf{?}]\right] < 1/3.$$

Therefore, a randomly chosen r of length $2^{cn_i+\ell}$ is accepted by V', with probability at least $2/3$. Equipped with this knowledge, we now give an algorithm A that defines a P-natural property of Boolean functions:

> $A(f)$: Given a Boolean function $f : \{0,1\}^{\ell'} \to \{0,1\}$, compute the largest $n \in \mathbb{Z}$ such that $\ell' \geq \ell + cn$, where $\ell = O(\log n)$ is the smallest integer satisfying $2n \leq 2^{\ell}$. Set r to be the first $2^{\ell+cn}$ bits of f. Search over all strings in $\{0,1\}^n$ for an x such that $V'(x, r) \neq \mathbf{?}$ for some r_i. Output *accept* if such an x is found; otherwise *reject*.

The algorithm A runs in $\text{poly}(2^{\ell'})$ time and accepts at least $1/2$ of its inputs. Furthermore, when the integer n computed by A is in the sequence $\{n_i\}$, A rejects all f with $\mathcal{C}$-circuit complexity at most $n_i^k = \Theta((\ell')^k)$: if f had such circuits, then all substrings r_i of f would as well, by Proposition 1. As this is true for every constant k, A is a P-natural property useful against polynomial-size $\mathcal{C}$-circuits. ∎

To prove a related result for RE predicates, we first need a little more notation. Let V be an $\mathsf{RTIME}[2^{kn}]$ predicate accepting a language L. For a given input length n, a set $S_n \subseteq \{0,1\}^{2^{kn}}$ is a *hitting set for V on n* if, for all $x \in L$ of length n, there is a $y \in S_n$ such that $V(x_n, y)$ accepts. For a string T of length $m \cdot 2^{kn}$, T *encodes a hitting set for V on n* if, breaking T into m strings $y_1, \ldots, y_m$ of equal length, the set $\{y_1, \ldots, y_m\}$ is a hitting set for V on n.

We consider yet another relaxation of naturalness. For a typical circuit class $\mathcal{C}$, we say that a polynomial-time algorithm A *is io-P-natural against $\mathcal{C}$* provided that, for every k and infinitely many integers n,

- A accepts at least a $1/\text{poly}(n)$ fraction of n-bit inputs, and
- A rejects all n-bit inputs x such that the corresponding Boolean function f_x has $((\log n)^k + k)$-size $\mathcal{C}$-circuits.[18]

(Compare with Definition 2.1.) In the usual notion of natural properties, we are restricted to inputs with length equal to a power of two, and largeness holds almost everywhere; here, neither condition is required.

We can relate succinctly encoded hitting sets to natural algorithms as follows.

THEOREM 5.1. *Suppose, for all c, $\mathsf{RTIME}[2^{O(n)}]$ does not have n^2-size hitting sets encoded by n^c-size circuits. Then, for all c, there is an io-P-natural algorithm useful against n^c-size circuits.*

Proof. The hypothesis says that for every c, there is an $\mathsf{RTIME}[2^{O(n)}]$ predicate V_c accepting some language L with the following property: for every n^c-size circuit family $\{C_n\}$, there are infinitely many n where $tt(C_n)$ does not encode an n^2-size hitting set for V_c on n.

We may obtain an *io*-natural algorithm running in $\text{poly}(N)$ time with $O(\log N)$ bits of advice (where N is the length of the input) as follows.

[18] As usual, if $\mathcal{C}$ is also characterized by a depth d or modulus constraint m, those d and m are quantified alongside k.

> $A(Y,a)$: Given Y of length $N = 2^{kn+2\log n}$, view the $O(\log N)$-bit advice string a as the number of inputs of length n in $L(V_c)$.
>
> Partition Y into $y_1, \ldots, y_{2^{2\log n}}$ of length 2^{kn}.
>
> Let b be the number of x of length $n \leq (\log N)/k$ such that $V_c(x, y_i)$ accepts for some i. If $(a = b)$, then *accept*; else *reject*.

For infinitely many N, this procedure (with the appropriate advice string a) accepts a random string with high probability, because a random collection of n^2 strings is a hitting set with high probability. On those same input lengths N, the procedure A also rejects strings encoded by n^c-size circuit families, by assumption. Therefore A defines *io*-$\mathsf{P}/(\log n)$-natural algorithm A useful against n^c-size circuits, running on strings Y with length equal to a power of two.

We can use A to design an *io*-P-natural algorithm A' that runs on arbitrary length strings, analogously to one direction of Theorem 3.2. For every $n \in \mathbb{N}$, we associate the interval $I_n = [n^2, (n+1)^2 - 1]$; note that the collection of I_n is a partition of $\mathbb{N}$. Our algorithm A' runs as follows:

> $A'(X)$: On input X of length m, determine n such that $m \in I_n$.
>
> If n is not a power of two, *reject*.
>
> Compute $a = m - n^2$, and treat a as a binary string of length $O(\log n)$.
>
> Let Y be the first n bits of X.
>
> Run $A(Y, a)$ and output the answer.

Observe that $A'(X)$ runs in $\mathrm{poly}(m)$ time. Since a as defined in A' is contained in $\{0, \ldots, 2n\}$, the number a can be treated as an advice string of length $(\log n)$ for n-bit inputs.

For infinitely many input lengths n_i, the original algorithm A (equipped with the appropriate advice a_i) satisfies largeness and usefulness against n^c-size circuits. For each such n_i, there is a slightly larger input length m_i such that the number of n_i-bit inputs in $L(V_c)$ is exactly $a_i = m_i - n_i^2$.

On these integers m_i, the algorithm $A'(\cdot)$ also satisfies largeness and usefulness, since it is essentially equivalent to running $A(\cdot, a_i)$ on inputs of length n_i. More precisely, since the input length has increased by a square ($m_i = \Theta(n_i^2)$), the strings of length m_i define functions on only twice as many input bits as n_i. Therefore, when $A(x, a_i)$ accepts (hence x has circuit complexity at least $(\log n_i)^c$), by Lemma 2.1 we may conclude that the original input X to A' defines a Boolean function on at most $2\log m_i \leq 4 \log n_i$ bits, with circuit complexity at least $(\log n_i)^c - (\log n_i)^{1+o(1)}$. Therefore the new algorithm A' is useful against circuits of size up to $(n/4)^c$. As this condition holds for every constant c, the theorem follows. ∎

The other direction (from *io*-P-natural algorithms to $\mathsf{RTIME}[2^{O(n)}]$) seems difficult to satisfy: it could be that, for infinitely many n, the natural algorithm does not obey any nice promise conditions on the number of accepted inputs of length n.

5.1. Unconditional mild derandomizations. We are now prepared to give some unconditionally true derandomization results. The first one is the following.

REMINDER OF THEOREM 1.8. *Either* $\mathsf{RTIME}[2^{O(n)}] \subset \mathsf{SIZE}[n^c]$ *for some* c*, or* $\mathsf{BPP} \subset$ *io*-$\mathsf{ZPTIME}[2^{n^\varepsilon}]/n^\varepsilon$ *for all* $\varepsilon > 0$.

To give intuition for the proof, we compare with the "easy witness" method of Kabanets [19], which shows that RP can be *pseudo*-simulated in *io*-$\mathsf{ZPTIME}[2^{n^\varepsilon}]$ (no efficient adversary can generate an input on which the simulation fails, almost everywhere). That simulation works as follows: for all $\varepsilon > 0$, given an RP predicate, try all n^ε-size circuits and check whether any encode a good seed for the predicate. If this always happens (against all efficient adversaries), then we can simulate RP in subexponential time. Otherwise, some efficient algorithm can generate, infinitely often, inputs on which this simulation fails. This algorithm generates the truth table of a function that does not have n^ε-size circuits; this hard function can be used to derandomize BPP.

In order to get a nontrivial simulation that works on all inputs for many lengths, we consider *easy hitting sets*: sets of strings (as in Theorem 5.1) that contain seeds for *all* inputs of a given length, encoded by n^c-size circuits (where c does not have to be tiny, but rather a fixed constant). When such seeds exist for some c, we can use $\tilde{O}(n^c)$ bits of advice to simulate RP deterministically. Otherwise, we apply Theorem 5.1 to obtain an *io*-P-natural algorithm which can be used (by randomly guessing a hard function) to simulate BPP in subexponential time. This allows us to avoid explicit enumeration of all small circuits; instead, we let the circuit size exceed the input length, and we enumerate over (short) inputs in our natural property.

Proof of Theorem 1.8. First, suppose there exists a $c \geq 2$ so that for every $\mathsf{RTIME}[2^{O(n)}]$ predicate V accepting a language L, there is an n^{c-1}-size circuit family $\{C_n\}$ such that for almost all n, C_n has $O(n)$ inputs and its truth table encodes a hitting set for V on n with $2^{2\log n}$ strings. That is, the truth table of C_n is a string Y of length $\ell = 2^{2\log n} \cdot 2^{kn}$ for a constant k, with the property that when we break Y into $O(n^2)$ equal length strings $y_1, \ldots, y_{2^{2\log n}}$, the set $\{y_i\}$ is a hitting set for V on n. Then it follows immediately that $\mathsf{RTIME}[2^{O(n)}] \subset \mathsf{TIME}[2^{O(n)}]/n^c$, because for almost all lengths n, we can provide the appropriate n^{c-1}-size circuit C_n as $O(n^c)$ bits of advice, and recognize L on any n-bit input x by evaluating C on all its possible inputs, testing the resulting hitting set of $O(n^2)$ size with x. (We will show later how to strengthen this case.)

If the above supposition is false, that means for every c, there is an $\mathsf{RTIME}[2^{O(n)}]$ predicate V_c accepting some language L with the following property: for every n^c-size circuit family $\{C_n\}$, there are infinitely many n such that the truth table of C_n does not encode a hitting set for V on n. Theorem 5.1 says that for all c, we can extract an *io*-P-natural algorithm A_c useful against n^c-size circuits, for all c. In particular, the proof of Theorem 5.1 shows that for all c there are infinitely many n and $m \in [2^{n/3}, 2^{3n}]$ such that A_c is useful and large on its inputs of length m. So if we want a function $f : \{0,1\}^{O(n)} \to \{0,1\}$ that does not have n^k-size circuits, then by setting $c = k$, providing the number m as $O(n)$ bits of advice, and randomly selecting Y of m bits, we can generate an f that has guaranteed high circuit complexity, with zero-error.

For every k, we can simulate any language in $\mathsf{BPTIME}[O(n^k)]$ (two-sided randomized n^k time) as follows. Given any k and $\varepsilon > 0$, set $c = gk/\varepsilon$ (where g is the constant in Theorem 2.1). On input x of length n, our ZP simulation will have hard-coded advice of length $O(n^\varepsilon)$, specifying an input length $m = 2^{\Theta(n^\varepsilon)}$. Then it chooses a random string Y of length m and computes $A_c(Y)$. If $A_c(Y)$ rejects, then the simulation outputs *don't know.* (For the proper advice m and the proper input lengths, this case will happen with low probability.) Otherwise, for infinitely many n, Y is an $m = 2^{\Theta(n^\varepsilon)}$ bit string with circuit complexity at least $(n^\varepsilon)^c \geq n^{gk}$.

Applying Theorem 2.1, Y can be used to construct a PRG $G_Y : \{0,1\}^{g \log |Y|} \to \{0,1\}^{n^{3k}}$ which fools circuits of size n^{3k}, where d is a universal constant (independent of ε and k). Each call to G_Y takes $\text{poly}(|Y|) \leq 2^{O(n^{\varepsilon})}$ time. Trying all $|Y|^g \leq 2^{O(n^{\varepsilon})}$ seeds to G_Y, we can approximate the acceptance probability of an n^{3k}-size circuit simulating any $\mathsf{BPTIME}[O(n^k)]$ language on n-bit inputs, thereby determining acceptance/rejection of any n-bit input.

Now we have either (1) $\mathsf{RTIME}[2^{O(n)}] \subset \mathsf{TIME}[2^{O(n)}]/n^c$ for some c, or (2) $\mathsf{BPP} \subset io\text{-}\mathsf{ZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$ for all $\varepsilon > 0$. To complete the proof, we recall that Babai et al. [7] proved that if $\mathsf{BPP} \not\subset io\text{-}\mathsf{SUBEXP}$, then $\mathsf{EXP} \subset \mathsf{P}/\text{poly}$. Therefore, if case (2) does not hold, the first case can be improved: using a complete language for E, we infer from $\mathsf{EXP} \subset \mathsf{P}/\text{poly}$ that $\mathsf{TIME}[2^{O(n)}] \subset \mathsf{SIZE}[n^c]$ for some c, so $\mathsf{RTIME}[2^{O(n)}] \subset \mathsf{SIZE}[n^c]$ for some constant c. ∎

Reminder of Corollary 1.1. *For some constant c,* $\mathsf{RP} \subseteq io\text{-}\mathsf{ZPSUBEXP}/n^c$.

Proof. By Theorem 1.8, there are two cases: (1) $\mathsf{RTIME}[2^{O(n)}] \subset \mathsf{SIZE}[n^c]$ for some c, or (2) $\mathsf{BPP} \subset io\text{-}\mathsf{ZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$ for all ε. In case (1), $\mathsf{RP} \subseteq \mathsf{RTIME}[2^{O(n)}] \subseteq \mathsf{TIME}[n^c]/n^c$. In case (2), $\mathsf{RP} \subseteq \mathsf{BPP} \subseteq io\text{-}\mathsf{ZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$. ∎

The simulation can be ported over to Arthur–Merlin games. Recall that a language L is in AM if and only if there is a k and deterministic algorithm $V(x,y,z)$ running in time $|x|^k$ with the following properties:

- If $x \in L$, then $\Pr_{y \in \{0,1\}^{|x|^k}}[\exists z \in \{0,1\}^{|x|^k}\ V(x,y,z) \text{ accepts}] = 1$.
- If $x \notin L$, then $\Pr_{y \in \{0,1\}^{|x|^k}}[\forall z \in \{0,1\}^{|x|^k}\ V(x,y,z) \text{ rejects}] > 2/3$.

An AM computation corresponds to an interaction between a randomized verifier (Arthur) that sends random string y and a prover (Merlin) that nondeterministically guesses a string z.

Reminder of Corollary 1.2. *For some $c \geq 1$,* $\mathsf{AM} \subseteq io\text{-}\Sigma_2\mathsf{SUBEXP}/n^c$.

Note that $\mathsf{AM} \subseteq \Pi_2\mathsf{P}$ [6], but it has been open for some time to find an interesting relationship between AM and $\Sigma_2\mathsf{P}$; see, for example, [15, 5].

Proof (sketch). The proof is roughly analogous to relativizing Theorem 1.8 with an NP oracle; for completeness, we include some of the details. Instead of hitting sets for RP computations, we consider hitting sets for AM computations: a poly(n)-size set S of n^k-bit strings that can replace the role of y (Arthur) in the AM computation. (Such hitting sets always exist, by a probabilistic argument.) That is, on all strings x of length n, computing the probability of $(\exists z)[V(x,y,z)]$ over all $y \in S$ allows us to approximate the probability over *all* n^k-bit strings. Instead of considering hitting sets that are succinctly encoded by typical circuits, we consider AM hitting sets that are succinctly encoded by circuits with oracle gates that compute SAT. There are two possible cases:

1. *There is a c such that for all languages $L \in \mathsf{AM}$ and verifiers V_c for L, there is an n^c-size SAT-oracle circuit family encoding hitting sets for V_c, on almost all input lengths n.* In this case, we can put AM in the class $\mathsf{P}^{\mathsf{NP}}/\tilde{O}(n^c)$: we can use $\tilde{O}(n^c)$ advice to store a circuit encoding a hitting set for each input length n, evaluate this circuit on $n^{O(1)}$ inputs in P^{NP}, producing the hitting set, and then use the hitting set and the NP oracle to simulate the AM computation.

2. *For all c, there is some verifier V of some AM language such that, for infinitely many input lengths n, every hitting set for V over all inputs of length n has SAT-oracle circuit complexity greater than n^c.* First we show how to use this case to check that a given string Y has high SAT-oracle circuit complexity for infinitely many input

lengths; the argument is similar to prior ones. Given a string Y, let $k \geq 1$ be a parameter, let $\varepsilon > 0$ be sufficiently small, and consider the verifier $V_{10k/\varepsilon}$ on all inputs of length $n = m^{\varepsilon}$ (where n is one of the infinitely many input lengths which are "good"). We can verify that the string Y encodes a hitting set for $V_{10k/\varepsilon}$ on inputs of length n as follows. First we guess those 2^n strings of length n which are accepted, and those which are rejected (comparing our guesses against the $O(n)$ bits of advice, which will encode the total number of accepted inputs of length n). For each string that is guessed to be accepted, we use the set S and nondeterminism to simulate Arthur and Merlin's acceptance in $2^n \cdot \text{poly}(n)$ time. Then, for each string that is guessed to be rejected, we use the string Y and universal guessing to confirm that Arthur and Merlin reject in $2^n \cdot \text{poly}(n)$ time. This is a Σ_2 computation running in time $2^{O(n)} \leq 2^{O(m^{\varepsilon})}$, which (when given the appropriate advice of length $O(m^{\varepsilon})$) correctly determines that at least some string Y has SAT-oracle circuit complexity at least $(m^{\varepsilon})^{10k/\varepsilon} \geq n^{10k}$, on infinitely many input lengths.

Now suppose we want to simulate an AM computation on inputs of length m running in time m^k. That AM computation can be simulated in $io\text{-}\Sigma_2\mathsf{TIME}[2^{n^{\varepsilon}}]/O(n^{\varepsilon})$ as follows: we guess a string Y with high SAT-oracle circuit complexity, and we apply known relativizing results in derandomization (in particular, Theorems 3.2 and 3.3 from [22]) that use the string Y to simulate AM computations in $\mathsf{NSUBEXP}$. Then we apply the aforementioned Σ_2 procedure to verify that the Y guessed has high SAT-oracle circuit complexity. We accept if and only if the simulation of AM accepts and the verification of Y accepts. ∎

It looks plausible that Corollary 1.2 could be combined with other results (for example, the work on lower bounds against fixed-polynomial advice of Buhrman, Fortnow, and Santhanam [10]) to prove new separations.

Another application of Theorem 1.8 is an unexpected equivalence between the infamous separation problem $\mathsf{NEXP} \neq \mathsf{BPP}$ and zero-error simulations of BPP. We need one more definition: $\mathsf{Heuristic}\,\mathcal{C}$ is the class of languages L such that there is an $L' \in \mathcal{C}$ whereby, for almost every n, the symmetric difference $(L \cap \{0,1\}^n)\Delta(L' \cap \{0,1\}^n)$ has cardinality less than $2^n/n$.[19] (That is, there is a language in $\mathcal{C}$ that "agrees" with L on at least a $1 - 1/n$ fraction of inputs.) The infinitely often version $io\text{-}\mathsf{Heuristic}\,\mathcal{C}$ is defined analogously.

Reminder of Theorem 1.9. $\mathsf{NEXP} \neq \mathsf{BPP}$ *if and only if for all* $\varepsilon > 0$, $\mathsf{BPP} \subseteq io\text{-}\mathsf{HeuristicZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$.

This extends an amazing result of Impagliazzo and Wigderson [17] that $\mathsf{EXP} \neq \mathsf{BPP}$ if and only if for all $\varepsilon > 0$, $\mathsf{BPP} \subseteq io\text{-}\mathsf{HeuristicTIME}[2^{n^{\varepsilon}}]$. It is interesting that NEXP versus BPP, a problem concerning the power of nondeterminism, is equivalent to a statement about derandomization of BPP *without* nondeterminism. Theorem 1.9 should also be contrasted with the NEXP vs. P/poly equivalence of Impagliazzo, Kabanets, and Wigderson[16]: $\mathsf{NEXP} \not\subset \mathsf{P}/\mathsf{poly}$ if and only if $\mathsf{MA} \subseteq io\text{-}\mathsf{NTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$ for all $\varepsilon > 0$.

Proof of Theorem 1.9. First, assume BPP is not in $io\text{-}\mathsf{HeuristicZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$ for some ε. Then $\mathsf{BPP} \not\subseteq io\text{-}\mathsf{ZPTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$, so by Theorem 1.8 we have that $\mathsf{RTIME}[2^{O(n)}]$ has size-n^c seeds, which implies $\mathsf{REXP} = \mathsf{EXP}$. The hypothesis also implies that BPP is not in $io\text{-}\mathsf{HeuristicTIME}[2^{n^{\varepsilon}}]$, so by Impagliazzo and Wigderson [17] we have $\mathsf{EXP} = \mathsf{BPP}$. Therefore $\mathsf{REXP} = \mathsf{BPP}$. But this implies $\mathsf{NP} \subseteq \mathsf{BPP}$, so by

[19] **N.B.** This is a weaker definition than usually stated, but it will suffice for our purposes.

Ko's theorem [23] we have $\mathsf{NP} = \mathsf{RP}$. Finally, by padding, $\mathsf{NEXP} = \mathsf{REXP} = \mathsf{BPP}$.

For the other direction, assume $\mathsf{NEXP} = \mathsf{BPP}$ and that for all $\varepsilon > 0$ we have $\mathsf{BPP} \subseteq io\text{-}\mathsf{HeuristicZPTIME}[2^{n^\varepsilon}]/n^\varepsilon$. We wish to prove a contradiction. The two assumptions together say that $\mathsf{NEXP} \subseteq io\text{-}\mathsf{HeuristicNTIME}[2^{n^\varepsilon}]/n^\varepsilon$ for all $\varepsilon > 0$. $\mathsf{NEXP} = \mathsf{BPP}$ implies $\mathsf{NEXP} = \mathsf{EXP}$, and since NE has a linear-time complete language, we have $\mathsf{NTIME}[2^{O(n)}] \subseteq \mathsf{TIME}[2^{O(n^c)}]$ for some constant c. (More precisely, the SUCCINCTHALTING problem from Theorem 1.1 can be solved in $2^{O(n^c)}$ time for some c, and every language in $\mathsf{NTIME}[2^{O(n)}]$ can be reduced in linear time to SUCCINCTHALTING.) As a consequence, we derive

(1)
$$\mathsf{EXP} = \mathsf{NEXP} \subseteq \bigcap_{\varepsilon>0} io\text{-}\mathsf{HeuristicNTIME}[2^{n^\varepsilon}]/n^\varepsilon \subseteq \bigcap_{\varepsilon>0} io\text{-}\mathsf{HeuristicTIME}[2^{O(n^c)}]/n^\varepsilon.$$

The last inclusion in (1) can be proved as follows: let L be an arbitrary language in $\bigcap_{\varepsilon>0} io\text{-}\mathsf{HeuristicNTIME}[2^{n^\varepsilon}]/n^\varepsilon$, and let $L' \in \bigcap_{\varepsilon>0} \mathsf{NTIME}[2^{n^\varepsilon}]/n^\varepsilon$ be such that $(L \cap \{0,1\}^n)\Delta(L' \cap \{0,1\}^n) \le 2^n/n$ on infinitely many n. This means that, for any ε, L' can be solved using a collection of nondeterministic machines $\{M_n\}$ running in 2^{n^ε} time such that M_n solves all instances on n bits and the description of M_n can be encoded in $O(n^\varepsilon)$ bits. To get a collection of equivalent deterministic machines, let M_n be the advice for inputs of length n; on any input x of length n, call the $2^{O(n^c)}$ time algorithm for SUCCINCTHALTING on the input $\langle M_n, x, b(2^{n^\varepsilon})\rangle$, where $b(m)$ is the binary encoding of m. Using standard encodings, this instance has $n + O(n^\varepsilon)$ length; hence it is solved deterministically in $2^{O(n^c)}$ time.

Finally, we prove that the above inclusion (1) is false, by direct diagonalization. That is, we can find an $L \in \mathsf{EXP}$ such that $L \notin io\text{-}\mathsf{HeuristicTIME}[2^{O(n^c)}]/n^{1/2}$. Let $\{M_i\}$ be a list of all 2^{n^c} time machines. We will give a $2^{n^{c+1}}$-time M diagonalizing (even heuristically) against all $\{M_i\}$ with $n^{1/2}$ advice. For every n, M divides up its n-bit inputs into blocks of length $B = 1 + n^{1/2} + \log n$, with $2^n/B$ blocks in total. On input x of length n, M identifies the block containing x, letting $x_1, \ldots, x_B$ be the strings in that block. Let $\{a_j\}$ be the set of all possible advice strings of length $n^{1/2}$. The following loop is performed.

Let $S_0 = \{(j,k) \mid j = 1, \ldots, n,\ k = 1, \ldots, 2^{n^{1/2}}\}$. For $i = 1, \ldots, B$, decide that M accepts x_i if and only if the majority of $M_j(x_i, a_k)$ reject over all $(j,k) \in S_{i-1}$. Set S_i to be the subset of S_{i-1} containing those (M_j, a_k) which agree with M on x_i. If $x_i = x$, then output the decision.

Observe that M runs in $B \cdot n \cdot 2^{O(n^c)} \le O(2^{n^{c+1}})$ time. For every block and every i, we have $|S_i| \le |S_{i-1}|/2$. Since $|S_0| = 2^{n^{1/2}} \cdot n$, this implies that $|S_B| = 0$. So for every block, every pair (M_j, a_k) disagrees with M on at least one input. Therefore every pair (M_j, a_k) disagrees with M on at least $2^n/B > 2^n/n$ inputs, one from each block, and this happens for almost all input lengths n. Summing up, for almost every n we have that M disagrees with every M_i and its $n^{1/2}$ bits of advice, on greater than a $1/n$ fraction of n-bit inputs. That is, $L(M) \in \mathsf{EXP}$ but $L(M) \notin io\text{-}\mathsf{HeuristicTIME}[2^{O(n^c)}]/n^{1/2}$. ∎

Remark 1. An anonymous reviewer observed that the previous proof, very slightly modified, also shows $\mathsf{NEXP} \neq \mathsf{BPP}$ if and only if $\mathsf{BPP} \subseteq io\text{-}\mathsf{HeuristicNTIME}[2^{n^\varepsilon}]/n^\varepsilon$ for all $\varepsilon > 0$. That is, separating NEXP from BPP is *equivalent* to obtaining a nontrivial simulation of BPP with nondeterminism.

6. Unconditional derandomization of natural properties. In this last section, we show how one can use similar ideas to generically "derandomize" natural properties, in the sense that RP-natural properties entail P-natural ones. The formal claim is the following.

Reminder of Theorem 1.10. *If there exists an* RP*-natural property P useful against a typical class $\mathcal{C}$, then there exists a* P*-natural property P' useful against $\mathcal{C}$.*

That is, suppose there is a randomized algorithm that can distinguish hard functions from easy functions with one-sided error—the algorithm may err on some hard functions but never on any easy functions. Then we can obtain a deterministic algorithm with essentially the same functionality. The idea behind P' is directly inspired by other arguments in the paper (such as the proof of Theorem 1.7): we split the input string T into small substrings, and we feed the substrings as inputs to P while the whole input string T is used as randomness to P.

Proof. Suppose A is a randomized polytime algorithm taking n bits of input and n^{k-2} bits of randomness (for some $k \geq 3$), deciding a large and useful property against n^c-size circuits for every c. For concreteness, let us say that A accepts some $1/n^b$-fraction of n-bit inputs with probability at least $2/3$, and rejects all n-bit truth tables of $(\log n)^c$-size circuits, where $b \geq k$ (making b larger is only a weaker guarantee). Standard amplification techniques show that, by increasing the randomness from n^{k-2} to n^k, we can boost the success probability of A to greater than $1 - 1/4^n$.

Our deterministic algorithm A' will, on n-bit input T, partition T into substrings $T_1, \ldots, T_{n^{1-1/k}}$ of length at most $n^{1/k}$ each and will *accept* if and only if $A(T_i, T)$ accepts for some i.

First, we show that A' satisfies largeness. Consider the set R of n-bit strings T such that for all $n^{1/k}$-bit strings x, $A(x, T)$ accepts if and only if $A(x, T')$ accepts for some n-bit T'. As there are only $2^{n^{1/k}}$ strings on $n^{1/k}$ bits, and the probability that a random n-bit T works for a given $n^{1/k}$-bit string is at least $1 - 1/4^{n^{1/k}}$, we have (by a union bound) that $|R| \geq 2^n \cdot (1 - 2^{n^{1/k}}/4^{n^{1/k}}) \geq 2^n \cdot (1 - 1/2^{n^{1/k}})$.

Now consider the set S of all n-bit strings $T = T_1 \cdots T_{n^{1-1/k}}$ (where for all i, $|T_i| = n^{1/k}$) such that $A(T_i, T')$ accepts for some i and some n-bit T'. Since there are at least $t = 2^{n^{1/k}}/n^{b/k}$ such strings T_i of length $n^{1/k}$ (by largeness of A), the cardinality of S is at least

$$n^{1-1/k} \cdot t \cdot \left(2^{n^{1/k}} - t\right)^{n^{1-1/k}-1} = n^{1-1/k} \cdot \frac{2^{n^{1/k}}}{n^{b/k}} \cdot \left(2^{n-n^{1/k}}\right) \cdot \left(1 - 1/n^{b/k}\right)^{n^{1-1/k}-1},$$

as this expression just counts the number of strings T with exactly one T_i from the t strings accepted by A. Since $b \geq k$, $(1 - 1/n^{b/k})^{n^{1-1/k}-1} \geq 1/e$, and the above expression simplifies to $\Omega(2^n/n^{1/k-1+b/k})$. Therefore, there is a constant $e = b/k + 1/k - 1$ such that $|S| \geq \Omega(2^n/n^e)$.

Observe that if $T \in S \cap R$, then $A(T_i, T)$ accepts for some i (where T_i is defined as above). Applying the inequality $|S \cap R| \geq |S| + |R| - 2^n$, there are at least $2^n(1/n^e - 1/2^{n^{1/k}})$ strings such that $A(T_i, T)$ accepts for some i. This is at least $2^n/n^{e+1}$ for sufficiently large n, so A' satisfies largeness.

Second, we show that A' is useful. Suppose for a contradiction that $A'(T)$ accepts for some T with $(\log |T|)^c$-size circuits, where c is an arbitrarily large (but fixed) constant. Then $A(T_i, T)$ must accept for some i. Because A is useful against n^d-size circuits for all d, it must be that T_i cannot have $(\log |T_i|)^{c+1}$-size circuits. However, recall that if a string T has $(\log |T|)^c$-size circuits, then by Lemma 2.1, every $|T|^{1/k}$-length substring T_i of T has circuit complexity at most $(\log |T|)^c + (\log |T|)^{1+o(1)} \leq$

$2 \cdot (k \cdot \log |T_i|)^c$. As k is a fixed constant, this quantity is less than $(\log |T_i|)^{c+1}$ when $|T_i|$ is sufficiently large, a contradiction. $\square$

7. Conclusion. Ketan Mulmuley has recently suggested that "$\mathsf{P} \neq \mathsf{NP}$ because P is big, not because P is small" [27]. That is to say, the power of efficient computation is the true reason we can prove lower bounds. The equivalence in Theorems 1.1 and 1.2 between NEXP lower bounds and constructive useful properties can be viewed as one rigorous formalization of this intuition. We conclude with some open questions of interest.

- *Do* NEXP *problems have witnesses that are* average-case hard *for* ACC^0*?* More precisely, are there NEXP predicates with the property that, for almost all valid witnesses of length $2^{O(n)}$, their corresponding Boolean functions on $O(n)$ variables are such that that no ACC^0 circuit of polynomial size agrees with these functions on $1/2 + 1/\text{poly}(n)$ of the inputs? Such predicates could be used to yield unconditional derandomized simulations of ACC^0 circuits (using nondeterminism). The primary technical impediment seems to be that we do not think ACC^0 can compute the majority function, which appears to be necessary for hardness amplification (see [34]). But this should make it *easier* to prove lower bounds against ACC^0, not harder!
- *Equivalences for nonuniform natural properties?* In this paper, we have mainly studied natural properties decidable by algorithms with $\log n$ bits of advice or less; however, the more general notion of P/poly-natural proofs has also been considered. Are there reasonable equivalences that can be derived between the existence of such properties and lower bounds?
- *What algorithms follow from stronger lower bound assumptions?* There is an interesting tension between the assumptions "$\mathsf{NEXP} \not\subset \mathsf{P}$/poly" and "integer factorization is not in subexponential time." The first asserts nontrivial efficient algorithms for recognizing some hard Boolean functions (as seen in Theorems 1.1 and 1.2); the second denies efficient algorithms for recognizing a nonnegligible fraction of hard Boolean functions [20, 3]. An equivalence involving $\mathsf{NP} \not\subset \mathsf{P}$/poly could yield more powerful algorithms for recognizing hardness. In recent work addressing this problem, Chapman and Williams [12] prove that $\mathsf{NP} \not\subset \mathsf{P}$/poly is equivalent to the existence of natural properties which are true of SAT but are useful against all polynomial-size "SAT-solving" circuits.

Acknowledgments. I thank Amir Abboud, Marco Carmosino, Russell Impagliazzo, Igor Carboni Oliveira, Steven Rudich, Rahul Santhanam, and the anonymous reviewers of SICOMP and STOC for useful comments and discussions. I also thank Emanuele Viola for a pointer to his paper with Eric Miles.

REFERENCES

[1] S. Aaronson, *Shtetl-Optimized*, http://www.scottaaronson.com/blog/?p=240 (May 2007).

[2] E. Allender, *When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity*, in FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Comput. Sci. 2245, Springer, Berlin, 2001, pp. 1–15.

[3] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger, *Power from random strings*, SIAM J. Comput., 35 (2006), pp. 1467–1493, doi:10.1137/050628994.

[4] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, Cambridge, UK, 2009.

[5] B. Aydinlioglu and D. van Melkebeek, *Nondeterministic circuit lower bounds from mildly de-randomizing Arthur-Merlin games*, in Proceedings of the IEEE Conference on Compu-

tational Complexity, 2012, pp. 269–279.

[6] L. Babai, *Trading group theory for randomness*, in Proceedings of the 17th Annual ACM Symposium on Theory of Computing, 1985, pp. 421–429.

[7] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, *BPP has subexponential time simulations unless EXPTIME has publishable proofs*, Comput. Complexity, 3 (1993), pp. 307–318.

[8] B. Barak, *A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms*, in Randomization and Approximation Techniques in Computer Science, Lecture Notes in Comput. Sci. 2483, Springer, Berlin, 2002, pp. 194–208.

[9] R. Beigel and J. Tarui, *On ACC*, Comput. Complexity, 4 (1994), pp. 350–366.

[10] H. Buhrman, L. Fortnow, and R. Santhanam, *Unconditional lower bounds against advice*, in Automata, Languages and Programming. Part I, Springer, Berlin, 2009, pp. 195–209.

[11] A. K. Chandra, S. Fortune, and R. J. Lipton, *Unbounded fan-in circuits and associative functions*, J. Comput. System Sci., 30 (1985), pp. 222–234.

[12] B. Chapman and R. Williams, *The circuit-input game, natural proofs, and testing circuits with data*, in Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS 2015), Rehovot, Israel, 2015, pp. 263–270.

[13] T. Y. Chow, *Almost-natural proofs*, J. Comput. System Sci., 77 (2011), pp. 728–737.

[14] L. Fortnow, R. Santhanam, and L. Trevisan, *Hierarchies for semantic classes*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 2005, pp. 348–355.

[15] D. Gutfreund, R. Shaltiel, and A. Ta-Shma, *Uniform hardness versus randomness tradeoffs for Arthur-Merlin games*, Comput. Complexity, 12 (2003), pp. 85–130.

[16] R. Impagliazzo, V. Kabanets, and A. Wigderson, *In search of an easy witness: Exponential time vs. probabilistic polynomial time*, J. Comput. System Sci., 65 (2002), pp. 672–694.

[17] R. Impagliazzo and A. Wigderson, *Randomness vs time: Derandomization under a uniform assumption*, J. Comput. System Sci., 63 (2001), pp. 672–688.

[18] M. J. Jansen and R. Santhanam, *Stronger lower bounds and randomness-hardness tradeoffs using associated algebraic complexity classes*, in Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science, 2012, pp. 519–530.

[19] V. Kabanets, *Easiness assumptions and hardness tests: Trading time for zero error*, J. Comput. System Sci., 63 (2001), pp. 236–252.

[20] V. Kabanets and J.-Y. Cai, *Circuit minimization problem*, in Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, 2000, pp. 73–79.

[21] V. Kabanets and R. Impagliazzo, *Derandomizing polynomial identity tests means proving circuit lower bounds*, Comput. Complexity, 13 (2004), pp. 1–46.

[22] A. R. Klivans and D. van Melkebeek, *Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses*, SIAM J. Comput., 31 (2002), pp. 1501–1526, doi:10.1137/S0097539700389652.

[23] K.-I. Ko, *Some observations on the probabilistic algorithms and NP-hard problems*, Inform. Process. Lett., 14 (1982), pp. 39–43.

[24] M. Krause and S. Lucks, *Pseudorandom functions in $TC0$ and cryptographic limitations to proving lower bounds*, Comput. Complexity, 10 (2001), pp. 297–313.

[25] E. Miles and E. Viola, *Substitution-permutation networks, pseudorandom functions, and natural proofs*, in Advances in Cryptology—CRYPTO 2012, Lecture Notes in Comput. Sci. 7417, Springer, Heidelberg, 2012, pp. 68–85.

[26] P. B. Miltersen, N. V. Vinodchandran, and O. Watanabe, *Super-polynomial versus half-exponential circuit size in the exponential hierarchy*, in Computing and Combinatorics (Tokyo, 1999), Lecture Notes in Comput. Sci. 1627, Springer, Berlin, 1999, pp. 210–220.

[27] K. Mulmuley, *private communication*, 2011.

[28] M. Naor and O. Reingold, *Number-theoretic constructions of efficient pseudo-random functions*, J. ACM, 51 (2004), pp. 231–262.

[29] N. Nisan and A. Wigderson, *Hardness vs randomness*, J. Comput. System Sci., 49 (1994), pp. 149–167.

[30] I. Oliveira, *Algorithms versus Circuit Lower Bounds*, Tech. Report TR13-117, Electronic Colloquium on Computational Complexity (ECCC), 2013.

[31] A. Razborov and S. Rudich, *Natural proofs*, J. Comput. System Sci., 55 (1997), pp. 24–35.

[32] S. Rudich, *Super-bits, demi-bits, and $N\tilde{P}$/qpoly-natural proofs*, in Randomization and Approximation Techniques in Computer Science (Bologna, 1997), Lecture Notes in Comput. Sci. 1269, Springer, Berlin, 1997, pp. 85–93.

[33] R. Santhanam and R. Williams, *On medium-uniformity and circuit lower bounds*, in Proceedings of the IEEE Conference on Computational Complexity, 2013, pp. 15–23.

[34] R. Shaltiel and E. Viola, *Hardness amplification proofs require majority*, SIAM J. Comput., 39 (2010), pp. 3122–3154, doi:10.1137/080735096.

[35] C. Umans, *Pseudo-random generators for all hardnesses*, J. Comput. System Sci., 67 (2003), pp. 419–440.
[36] R. Williams, *Non-uniform ACC circuit lower bounds*, in Proceedings of the IEEE Conference on Computational Complexity, 2011, pp. 115–125.
[37] R. Williams, *Improving exhaustive search implies superpolynomial lower bounds*, SIAM J. Comput., 42 (2013), pp. 1218–1244, doi:10.1137/10080703X.
[38] S. Žák, *A Turing machine time hierarchy*, Theoret. Comput. Sci., 26 (1983), pp. 327–333.